Sebastian Rudolph
Frithjof Dau
Sergei O. Kuznetsov (Eds.)

LNAI 5662

Conceptual Structures: Leveraging Semantic Technologies

17th International Conference
on Conceptual Structures, ICCS 2009
Moscow, Russia, July 2009, Proceedings

Springer

Sebastian Rudolph   Frithjof Dau
Sergei O. Kuznetsov (Eds.)

Conceptual Structures: Leveraging Semantic Technologies

17th International Conference
on Conceptual Structures, ICCS 2009
Moscow, Russia, July 26-31, 2009
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Sebastian Rudolph
Universität Karlsruhe (TH)
Institut AIFB
76128 Karlsruhe, Germany
E-mail: Sebastian.Rudolph@kit.edu

Frithjof Dau
SAP Research Center CEC Dresden
Chemnitzer Strasse 48, 01187 Dresden, Germany
E-mail: frithjof.dau@sap.com

Sergei O. Kuznetsov
State University
Higher School of Economics
Department of Applied Mathematics and Information Science
Kirpichnaya 33/5, 105679 Moscow, Russia
E-mail: skuznetsov@hse.ru

Library of Congress Control Number: 2009930107

CR Subject Classification (1998): I.2, G.2.2, F.4.1, F.2.1, H.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-642-03078-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-03078-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12718344 06/3180 5 4 3 2 1 0

Preface

The nature of conceptual thinking constitutes a central topic in a variety of scientific disciplines. Since 1993, the International Conference on Conceptual Structures (ICCS) has served as a platform that brings together researchers and practioners in information and computer sciences as well as social science to explore novel ways of representing and analyzing conceptual knowledge. Originally centered around research on knowledge representation and reasoning with conceptual graphs, over the years ICCS has broadened its scope to include innovations from a wider range of theories and related practices, among them other forms of graph-based formalisms like RDF or existential graphs, formal concept analysis, Semantic Web technologies, ontologies, concept mapping and more. Today, ICCS draws inspiration from areas as diverse as artificial intelligence, knowledge representation and reasoning, applied mathematics and lattice theory, computational linguistics, conceptual modeling and design, diagrammatic reasoning and logic, intelligent systems and knowledge management.

In addition to vivid conferences, the vibrancy of the field is documented by two recently published books (Hitzler, Schärfe (Eds): *Conceptual Structures in Practice* and Chein, Mugnier: *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*) as well as by an ISO standard ("Common Logic", ISO/ IEC 24707) which orginated in this community.

This volume contains the proceedings of ICCS 2009, the 17th International Conference on Conceptual Structures (ICCS) held in Moscow. The theme of ICCS 2009, "leveraging semantic technologies," hints at the large overlap of the research fields of semantic technologies and conceptual structures, and emphasizes the goal of closer connecting these two areas. We are confident that fostering the exchange of ideas will lead to cross-fertilization and mutual benefit. We are proud that we could welcome seven renowned researchers who elaborated on the relationship between conceptual structures and semantic technologies from different perspectives. Five of the seven speakers submitted accompanying papers which were peer reviewed and included in this volume.

Roughly 50 papers were submitted to ICCS 2009 for peer review. All submissions were assessed by at least three referees one of whom was an Editorial Board member, who managed any neccessary revisions. The top-ranked 18 papers were selected for this volume, amounting to an acceptance rate of about 35%. Another nine papers were published in a supplementary volume as CEUR workshop proceedings. The thorough selection process would not have been possible without the help of the numerous reviewers to whom we express our thanks.

Last but not least, we would also like to thank the local organizing team and the administration of the State University Higher School of Economics (Moscow) who – with genuine Russian hospitality – took care of all the arrangements to make this conference pleasant and enjoyable.

July 2009

Sebastian Rudolph
Frithjof Dau
Sergei O. Kuznetsov

Organization

ICCS Executive

ICCS Administrative

ICCS Editorial Board

ICCS Program Committee

Radim Belohlavek (USA)
Tru Cao (Vietnam)
Dan Corbett (USA)
Olivier Corby (France)
Madalina Croitoru (France)
Juliette Dibie-Barthlemy (France)
Pavlin Dobrev (Bulgaria)
Udo Hebisch (Germany)
Joachim Hereth (Germany)
Nathalie Hernandez (France)
Wolfgang Hesse (Germany)
Richard Hill (UK)
Jan Hladik (Germany)
Andreas Hotho (UK)
Adil Kabbaj (Morocco)
Markus Krötzsch (Germany)
Rob Kremer (Canada)
Leonhard Kwuida (Switzerland)
Michel Leclere (France)
Robert Levinson (USA)
Philippe Martin (Australia)
Claudio Masolo (Italia)
Sergei Obiedkov (Russia)
Daniel Oberle (Germany)
John Old (UK)
Anne-Marie Rassinoux (Switzerland)
Gary Richmond (USA)
Olivier Ridoux (France)
Eric Salvat (France)
Ulrik Sandborg-Petersen (Denmark)
Jeffrey Schiffel (USA)
York Sure (Germany)
Rallou Thomopoulos (USA)
Denny Vrandecic (Germany)
GQ Zhang (USA)

ICCS Additional Referees

Folke Eisterlehner
Robert Jäschke

ICCS Sponsoring Institutions

ABBYY, Moscow
Microsoft Russia, Moscow
Poncelet Laboratory (UMI 2615 du CNRS), Moscow
Russian Foundation for Basic Research, Moscow
State University Higher School of Economics, Moscow

Table of Contents

Invited Papers

The Maturing Semantic Web: Lessons in Web-Scale Knowledge Representation (Abstract) 1
Mark Greaves

Concept Formation in Linguistic Ontologies 2
Natalia Loukachevitch

Conceptual Graph Rules and Equivalent Rules: A Synthesis 23
Marie-Laure Mugnier

Two Paradigms Are Better Than One, and Multiple Paradigms Are Even Better 32
Arun K. Majumdar and John F. Sowa

Semantic Search – Using Graph-Structured Semantic Models for Supporting the Search Process 48
Thanh Tran, Peter Haase, and Rudi Studer

Human Being and Mathematics Logical and Mathematical Thinking . . . 66
Rudolf Wille

Accepted Papers

Default Conceptual Graph Rules: Preliminary Results for an Agronomy Application 86
Jean-François Baget, Madalina Croitoru, Jérôme Fortin, and Rallou Thomopoulos

Towards Extraction of Conceptual Structures from Electronic Health Records 100
Svetla Boytcheva and Galia Angelova

Algorithm Design Using Traversals of the Covering Relation 114
Andrew Burrow

Representing and Reasoning about Different Viewpoints: An Agronomy Application 128
Madalina Croitoru and Rallou Thomopoulos

Access Policy Design Supported by FCA Methods 141
Frithjof Dau and Martin Knechtel

Using VRML Technology for Visualization of Relations between the Main Concepts of an Educational Course 155
Evgeny Eremin

Efficient Browsing and Update of Complex Data Based on the Decomposition of Contexts 159
Sébastien Ferré

In Search of Semantic Compositionality in Vector Spaces 173
Eugenie Giesbrecht

Frequent Itemset Mining for Clustering Near Duplicate Web Documents 185
Dmitry I. Ignatov and Sergei O. Kuznetsov

System Consequence 201
Robert E. Kent

Fusion of Claude Bernard's Experiments for Scientific Discovery Reasoning 219
Claire Laudy, Bassel Habib, and Jean-Gabriel Ganascia

Distinguishing Answers in Conceptual Graph Knowledge Bases 233
Nicolas Moreau, Michel Leclère, and Madalina Croitoru

A Practical Exploration of Ontology Interoperability 247
Simon Polovina, James Cooke, and Jeremy Loke

Relation Algebra Operations on Formal Contexts 257
Uta Priss

Conceptual Graphs and Datatypes 270
Thomas Raimbault, David Genest, and Stéphane Loiseau

Towards the Complexity of Recognizing Pseudo-intents 284
Barış Sertkaya

Another Reason Why Conceptual Graphs Need Actors 293
B.J. Smith and Harry Delugach

Relational Scaling in Relational Semantic Systems 307
Karl Erich Wolff

Author Index 321

The Maturing Semantic Web: Lessons in Web-Scale Knowledge Representation

Mark Greaves

Vulcan Inc.
MarkG@vulcan.com

Abstract. This paper is an extended abstract of the talk given at ICCS'09. Rules have long been considered as an essential component of knowledge-based systems. We focus here on conceptual graph rules and on the semantically equivalent knowledge constructs in logic and databases, namely rules with existential variables and tuple-generating dependencies. The aim of this presentation is to synthesize main decidability, complexity and algorithmic results obtained on this kind of rules. We emphasize the fact that the graph vision of rules has led to new results.

Concept Formation in Linguistic Ontologies

Natalia Loukachevitch

Research Computing Center of M.V. Lomonosov Moscow State University
(NIVC MGU)
Leninskiye Gory 1, building 4, NIVC MGU, Moscow 119991, Russia
louk@mail.cir.ru

Abstract. Problems of conceptualization in linguistic ontologies are discussed We show that it is necessary to form concepts of a linguistic ontology as close as possible to the meanings of linguistic units, because excessive generalization and clustering of meanings necessarily lead to distortions in the system of relations, excessive problems in a specific subject field, or an application. At the same time it is important to ensure that concepts can be distinguished from superconcepts and sibling concepts. The usage of really existing multiword expressions helps us mitigate these contradictory requirements. The introduction of concepts on the basis of multiword expressions does not change the essence of a linguistic ontology, but also makes the distinction between the concepts much clearer.

Keywords: thesaurus, linguistic ontology, conceptualization.

1 Introduction

An ontology is often considered to be independent of a natural language [1,2,3]. D. Lenat [4] emphasizes that taking the meaning of words into account can only confuse ("words are often red herrings"), the meanings of words divide the world ambiguously, and the division lines come from a variety of reasons: historical, physiological, etc.

From another point of view an ontology can not be fully independent of natural language. Names of the concepts in ontologies are often formulated in natural language – this is a standard practice that has been used in knowledge representation systems in artificial intelligence [2,5]. Breuster et.al [6] stress that people manipulate concepts through words. In all known ontologies the words are used to represent concepts. Therefore, phenomena that are not verbalized, can not be modelled. The Breuster et.al characterize this phenomenon as the Ontological Whorf-Sapir hypothesis, i.e. "that which can not be captured by words cannot be represented in an ontology." Y. Wilks [2,5] asserts that the symbols in representation languages are fundamentally based on the natural language, that a representation language is a means of human communication with the inherent dynamics, polysemy and possibility of extended interpretation.

Moreover, in ontological resources developed for specific domains most ontology concepts are related to the meanings of domain terms. So J. Tsujii and S. Ananiadou [7] stress that "in many fields of application, knowledge to be shared and integrated is presented mostly in text". Many ontologies in biology, such as GO (Gene Ontology) [8], actually represent the information-retrieval thesauri (controlled vocabularies), and they by nature differ from ontologies, required within the formal ontological approach to knowledge description.

As a result a paradoxical notion of *linguistic ontology* emerges, i.e. an ontology, concepts of which are considerably related to the meanings of linguistic units, the terms of the subject field [9, 10]. Linguistic ontologies cover most of the words of the language or the subject field, and at the same time they have an ontological structure represented in relations between the concepts. Therefore, a linguistic ontology can be considered as a special kind of a lexical database and a special type of an ontology. Linguistic ontologies are relatively weakly formalized, i.e. they belong to the "terminological" ontologies according to J. Sowa [11]. The role of the "linguistic ontologies" increases greatly in applications related to natural language processing.

Examples of linguistic ontologies are the Princeton WordNet [12] and wordnets of other languages. Information retrieval thesauri can also be considered as linguistic ontologies.One of the serious problems in the linguistic ontology development consists in formulation of the principles for the concepts formation, since the relations between the concepts and lexical meanings are quite complex. Understanding these issues is important for developers of any types of ontologies, because creation of any ontology deals more or less with lexical or terminological meanings.

In this paper we will describe problems of concept formation in linguistic ontologies. We will consider principles for introducing a new concept in such linguistic ontologies as Princeton WordNet, MikroKosmos ontology [2], information-retrieval thesauri. Finally, we present our approach to description of concepts in Thesaurus of Russian language - RuThes, which we also consider to be a linguistic ontology [13].

RuThes is used in information-retrieval applications, such as conceptual indexing; automatic text categorization, document clustering, automatic text summarization, question-answering. At present it includes more than 50 thousand concepts and more than 140 thousand Russian words and multiword expressions. It was translated into English and comprises almost 130 thousand English words and expressions.

2 Principles of Concept Introduction and Lexical Senses

The general recommendations on the ontology concepts formation are usually described as follows [3,14,15]. One needs to distinguish the concept and its name, i.e. synonyms of the same concept do not represent different classes, synonyms are just different names of the concepts.

A child concept should be distinctly different from the parent one. This difference can be expressed in the form of a distinctive property that the child concept bears, in limitations on slot fillers that are distinct from limitations of other classes; or in the existence of additional relationships with other concepts. A concept must be clearly distinguished from the concepts at the same level (sibling concepts).

These recommendations are not easy to realize if the developed ontology is based on the existing linguistic meanings. First of all, it is not easy to distinguish a concept and its names, working with linguistic meanings. Secondly, a critical challenge for resources designed for the natural language processing is the presentation of ambiguous words, especially if the meanings are closely related to each other. Moreover, a serious problem is caused by the words with similar meanings, or near-synonyms, the meanings of which can differ in several features (conceptual content, speaker's attitude, collocations, etc.), and be dependent on the context.

In the following sections we will discuss the way these problems are solved in specific linguistic ontologies. We will use the following notation: *CONCEPT, term* or *word, 'meaning',* 'transcription of Russian words*'*.

3 Confusion of a Concept and Its Name in Linguistic Ontologies

3.1 Confusion of a Concept and Its Name in WordNet

Initially, WordNet was considered to be a lexical rather than an ontological resource. However, over time, the growing importance of the ontological research, as well as the similarity of the WordNet nouns hierarchy with an ontology became apparent [16]. At the same time there exist a lot of deficiencies of WordNet descriptions from ontological point of view [17].

Numerous examples of confusion between concept and its name can be found in WordNet. This is due to the fact that the basic relation in WordNet is the synonymy. Sets of synonyms – synsets – are the main structural elements of WordNet. Definitions of synonyms in synsets are based on the principle of substitution of one for another in sentences [12]. This basic principle of the WordNet construction leads to the situation when different synsets are introduced for different ways of naming same entities.

There are several types of confusion of concepts and their names in wordnet-like resources.

First of all, the confusion of concepts and their names shows itself in the support of different hierarchies for different parts of speech. Indeed, if, for example, the *PRIVATIZATION* concept (*privatize, privatization*) is mentioned in a text using whatever parts of speech – it is always a reference to the same concept with the help of different lexical means, the parts of speech changing should not affect the relations between this concept and other concepts.

The first Wordnet followers (EuroWordNet project) considered the integration of all parts of speech–derivatives in a single synset, since such division is contrary to the principles of the development of ontological resources. However, the decision to connect parts of speech in the same hierarchy was not made [18].

The second type of confusion of a concept and its name in WordNet is the usage of different synsets to describe the old and new names, the names of concepts in different dialects of the language, in different text genres, etc.

According to ontological principles of the distinction of an entity and its name, all these different names should not produce new units of representation or concepts; they should remain to be textual expressions of the same concept. To differentiate the features of their linguistic use, they can be provided with extra labeling.

Thus, in the Princeton WordNet, one can find numerous examples of synsets, which appeared due to different characteristics of the use of words. For example, for describing slang synonyms of the word "nose", a special synset is introduced. This synset is presented as hyponym of *nose* synset.

beak, honker, hooter, nozzle, snoot, snout, schnozzle, schnoz *-- (informal terms for the nose)*

Informal words related to money are also collected in a separate synset:

boodle, bread, cabbage, clams, dinero, dough, gelt, kale, lettuce, lolly, lucre, loot, moolah, pelf, scratch, shekels, simoleons, sugar, wampum *-- (informal terms for money)*

Several synsets fix specific features of English dialects, as a special synset for domestic ass in British English:

Moke 1 *-- (British informal)*

domestic ass, donkey, Equus asinus *-- (domestic beast of burden descended from the African wild ass; patient but stubborn)*

Next example of confusion of a concept and its names consists in description of monetary units with the same name used in different countries such as franc or centime:

franc *-- (the basic monetary unit in many countries; equal to 100 centimes)*

centime *-- (a fractional monetary unit of several countries: France and Algeria and Belgium and Burkina Faso and Burundi and Cameroon and Chad and the Congo and Gabon and Haiti and the Ivory Coast and Luxembourg and Mali and Morocco and Niger and Rwanda and Senegal and Switzerland and Togo)*

From the ontological point of view such synsets are not valid, because similarity between different monetary units is only their names, they are different in value . Therefore concepts should be introduced for such entities as Swiss franc, French franc, American dollar, Canadian dollar, and so on.

3.2 Differentiation of Ontology and Lexicon in the MikroKosmos Ontology

The authors of the MikroKosmos [3] ontology make a clear distinction between an ontology and a lexicon. The concepts of the ontology are described as frames – sets of slots. The system's lexicon describes the meaning of words and phrases, by establishing links from them to the ontology concepts. This division mainly prevents confusion of a concept and its name.

The MikroKosmos ontology is relatively small; it contains about 6 thousand concepts. The lexicon contains tens of thousand linguistic expressions. A lexicon entry can have a simple structure - a reference to an ontology concept, or a rather complex structure, containing both a reference to an ontology concept and features of a particular lexical unit. The names of a concept in the ontology can look like English words or phrases, but their semantics is expressed by a set of well-defined relations between concepts.

The authors declare the independence of ontology from a specific natural language that manifests itself in two aspects:

1) the ontology contains no units specific to a language such as English or Spanish, although the names of concepts are given in English for the sake of convenience.
2) the concepts of ontology do not have one-to-one mapping to word senses in natural languages. Many concepts may not be mapped to any word in the language; other concepts may correspond to several words in the same language and vice versa.

The main stages of the ontology development include:

- assessment whether the meaning of a word gives a sufficient ground for the introduction of a new concept;
- location of the concept in the ontology, identification of existing concepts which can be described as generic concepts for the new one;
- description of the new concept features that should differ from the properties of superconcepts and subconcepts. These features are given not just by slots filling, but also in a more informative way, e.g., in the availability of other properties or relations to other concepts.

Thus, the proclaimed linguistic independence should not be misleading. At its core, the MikroKosmos ontology is certainly a linguistic ontology, because, the basic principle, which justifies the introduction of new concepts, is the existence of words with the same meaning in many languages.

At the same time, the principle of linguistic independence of this ontology stresses that in the construction of a linguistic ontology it is not necessary to follow the system of meanings of a specific language. A linguistic ontology can take into account the system of meanings of a particular language or an aggregate of languages, and in doing so adhere to the ontological principles of the concepts introduction.

3.3 Concepts and Terms in Information Retrieval Thesauri

Information retrieval thesauri are usually considered as a kind of ontological resources [19]. In addition, thesauri are based on the terms of a subject field, so they can also be considered as linguistic ontologies. Conventional information-retrieval thesauri regulated by national and international standards [20, 21] are intended to be used in manual indexing by human indexers.

The basic thesauri units are terms that are divided into descriptors (= authorized terms) and non-descriptors (= ascriptors). Most standards for information-retrieval thesauri highlight the connection between the terms and concepts of a subject field. The American standard points out that a term is one or more words referring to a concept. The ISO standard [20] emphasizes that an indexing term is a concept presentation, preferably in the form of a noun or a noun phrase. A concept is considered as a unit of a thought, mentally formed to reflect some or all of the properties of a concrete or an abstract, real or mental object. Concepts exist as abstract entities, regardless of the terms that express them.

It should be noted that not all thesauri developers distinguished concepts and terms. Thus, developers of AGROVOC thesaurus characterize their resource as term-oriented; it is manifested in the fact that a term cannot be complemented with synonyms. This feature of the thesaurus is considered by the authors as a disadvantage that must be corrected [22].

An important property of a descriptor is that it should be formulated explicitly, its implied meaning in the thesaurus should be clear to the user. If an unambiguous and clear descriptor cannot be found, the term, taken as a descriptor is supplied with a “relator” (a brief note) or a comment.

American standard Z39.19 [21] recommends to use relators for descriptors’ names, even when a descriptor sounds uniquely within a given subject field, but has different meanings in the general language, or other domains. This makes it easier to search through multiple databases and to compare descriptors of various subject fields. For example, it is proposed to introduce *Shells (structures*) descriptor for the engineering subject field, since the word *shell* has a lot of meanings in English.

4 Similar Meanings of Ambiguous Words

The existence of closely related meanings of ambiguous words presents major difficulties for developers of linguistic ontologies. The difficulty of automatic disambiguation of ambiguous words requires to formulate principles for the description of such sets of related meanings as concepts of a linguistic ontology.

4.1 Similar Meanings of Ambiguous Words in WordNet

Many authors admit that the differences of meanings in WordNet are too fine-grained for such applications as machine translation, information retrieval, text classification, question-answering systems, etc. In [23] it was indicated that the average number of senses in WordNet is larger than in traditional lexicographical dictionaries.

The number of senses of certain lexical items may vary in different lexical resources, dictionaries. However, a large number of meanings in WordNet causes difficulties in applications related to natural language processing and brings about the question of how and what senses can be combined (“clustered”) [24, 25] to use them in the applications.

Gonzalo [26] pointed out that the experiments on sense clustering led to a conclusion that the typology of the relations between different senses of ambiguous words is more useful than the formation of sense clusters, because sense proximity depends on the application. For example, metaphoric senses belong to different semantic fields therefore distinction of such senses is very important for information retrieval applications and question-answering systems. However, for machine translation applications, this distinction may be unimportant, as the metaphorical transfer may be similar in different languages.

Fellbaum and Miller [27] review attempts to cluster senses of WordNet. They emphasize that the sense clustering can be based on a variety of alternative criteria (semantic, syntactical, domain-oriented), which apparently confirms the significance of different sense clusters for natural language applications.The problem of automatic selection of WordNet senses in practical applications can be mitigated through the use of semantically-marked, according to WordNet meanings, corpus SemCor [28].

The OntoNotes project proposed its own way of integrating lexical meanings of an ambiguous word and concepts formation [29], which is based on the consideration of the use of an ambiguous word in a corpus, representing the majority of its senses. All examples of use are analyzed and divided into groups of senses most distant from each other, a branch node in the tree is created, and then for each node the process should be iterated.

The development of the ontology of word senses depends on the explanatory need or application requirements. A standard termination condition for the ontology development process is the absence of an obvious way of splitting the remaining group of senses into subgroups, or the existence of equally reasonable ways of splitting it into subgroups according to different reasons. The work also highlights the usefulness of a multilingual consideration for the appropriate separation of the meanings and concepts.

Consider an example of the verb *drive*, for which WordNet provides 22 individual senses. In OntoNotes project two independent experts defined 7 most important groups of meanings of this verb. The most frequent group of senses comprises seven senses from WordNet and can be called "Operating or traveling by means of a vehicle":

WN1: Can you ***drive*** *a truck?*

WN2: ***drive*** *to school,*

WN3: ***drive*** *her to school,*

WN12: this truck ***drives*** *well,*

WN13: He ***drives*** *taxi,*

WN14: The car ***drove*** *around the corner,*

WN16: ***Drive*** *the turnpike to work.*

It should be stressed that, e.g. from the viewpoint of a Russian native speaker this group of senses is not very evident because these senses correspond to 5 different (non-synonymous) Russian words: 'vodit' (WN1, WN13), 'ehat' (WN2, WN12), 'vezti' (WN3), 'povernut' (WN14), 'proehat' (WN16).

4.2 Similar Meanings of Ambiguous Words in the MikroKosmos Ontology

The basic rule, proclaimed for dealing with similar meanings of ambiguous words in the MikroKosmos ontology, is the reduction of polysemy rule [3]: it is necessary to decide how many dictionary meanings a particular lexical sense can represent, and to unite as many meanings as possible, so there will be as few different senses as possible.

The principles of distinguishing meanings are as follows:

- A candidate meaning must be clearly distinguishable from the already described meanings.
- It is necessary to check whether the meaning needs further clarification, if it is used in a short sentence. If one needs additional context to figure out what meaning is used, the meaning should not be introduced, but must be attributed to one of the existing meanings.

- It is necessary to check whether there is a property in the description of the meaning, which is filled with too small number of fillers. If so, the meaning must also be assigned to one of the more common meanings, or described as a multiword expression.

Seemingly, this procedure should reduce the problem of sense disambiguation, but on the other hand, it leads to violations of the ontology structure. Thus, N. Guarino [17] criticizes several existing ontologies, including the MikroKosmos ontology, for the polysemy of ontological nodes, e.g., for the treatment of the *WINDOW* concept as an artifact and as a place at the same time.

The problem resides in the fact that *window* in different contexts may denote an opening (*A man looked out of the window*) and an artifact (*The workers mounted a window*), and these entities *WINDOW (OPENING)* and *WINDOW (ARTIFACT)* are very closely linked to each other. This criticism relates to the fact that, in Guarino opinion, the polysemy in ontological nodes should not be permitted in any form. To conform to the principle of forbidding polysemy nodes, the ontology should have different nodes at different locations for such concepts as *WINDOW (ARTIFACT)* and *WINDOW (OPENING)*.

To reply to N. Guarino, the authors of the ontology [3, p. 129] explain that the fact of the English word *window* having two meanings is not crucial for the ontology development, since it is not considered that the relations between the meanings of the natural language (or, more precisely, the meanings of all known languages) and the ontology concepts should have a unique correspondence.

As a justification for their position the authors argue that they do not know such a natural language, in which the word for the *WINDOW* concept does not realize two meanings: the meaning of an opening and the meaning of an artifact. This semantic genericity is the strongest argument in favour of the fact that people can combine these two concepts. The authors of the ontology also emphasize that "an effort to split ontological concepts into ever smaller unambiguous units leads to a sharp increase in polysemy and, therefore, makes the task of disambiguation so much more difficult... So if the ontology is made less ambiguous, it only means that the ambiguity will have to be treated increasingly elsewhere" [3, p. 132].

However, we need to agree with N. Guarino that violations of the ontological structure is also a serious problem because, if it is required to use the described relationships between the concepts for the logical inference, then at first it will be necessary to determine if these relationships could be applied in current context, which means that the problem of choosing the right meaning of an ambiguous word simply shifted to another stage of text processing.

4.3 Lack of Similar Meanings of Ambiguous Words in Information Retrieval Thesauri

Recently terms were considered to be unambiguous, context-independent linguistic units, now it is known that the terms have many features of the common language units, in particular, the terms can also be ambiguous, and their meanings can be very similar, for example, *industrial production* (the process and the outcome).

However, as the traditional information retrieval thesauri are not intended for the automatic processing of texts, usually only one of related term meanings is presented in an information-retrieval thesaurus.

5 Near-Synonyms in Linguistic Ontologies

The problem with near-synonyms, i.e. different words with similar meanings, is that they can differ in many features: denotative content, language register, evaluation, dialect, collocations, etc. This justifies the existence of a special genre of "dictionaries of synonyms", which explain in detail the specificity of using synonyms.

For many of these sets of near-synonyms it is extremely difficult to establish a unique correspondence in other languages, because in another language the corresponding set of near-synonyms is characterized by its own system of parametric differences and, accordingly, its own specificities.

Although the linguistic ontology takes into account the existing lexical meanings, nevertheless it should remain to be an ontology. According to general principles of ontological hierarchy (see Section 2) its main elements – the concepts -- should have clear, context-independent differences from the related concepts.

G. Hirst [30] explains that for the description of the words with similar meanings in the linguistic ontologies, it is necessary to implement one more level of representation, a conceptual-semantic level. This level should specify a relatively coarse conceptual hierarchical system, which is based on denotative, context-independent properties of words. Each concept is linked to a set of near-synonyms, and their features (stylistic, evaluation, connotations, etc.) are described in additional intra-conceptual structures.

However, the denotative component of the meaning is often very difficult to separate from other components. For example, consider the problem of determining the optimal number of concepts (and principles of it) to be associated with the following set of words with the meaning '*error*': e*rror, fault, omission, oversight, blunder, mistake, miss, screw-up, dereliction, defect.*

The authors of work [31] point out that it is often very difficult to determine the words with similar meaning that should better be described as a part of the internal structure of concept, and which should belong to different concepts. On the one hand, linguist's intuition can help. On the other hand, a look at the conceptual structure in terms of another language can really help to better delimit the boundaries of the concepts.

5.1 Near-Synonyms in WordNet

To describe the relationship between meanings according to the principles of the possibility of synonymous substitutions in same sentences, as it was made during the creation of Princeton WordNet, means that near-synonyms should be classified on several grounds, as the synonymous substitution of the word must take into account conceptual, stylistic, attitudinal and other components of the meaning. It is clear that the development of a hierarchy on such grounds is impossible; the whole construction becomes very volatile during the transition from language to language.

Therefore, WordNet has a large number of synsets that are difficult to distinguish from one another; this violates the ontological principles of descriptions of the concepts. For example, there are four different synsets denoting *likeness, similarity*, each next synset is a hyponym of the previous, which is hardly distinguishable from its hyperonym:

sameness *-- (the quality of being alike)*

similarity *-- (the quality of being similar)*

likeness, alikeness, similitude *-- (similarity in appearance or character or nature between persons or things)*

resemblance *-- (similarity in appearance or external or superficial details)*

5.2 Near-Synonyms in MikroKosmos Ontology

In the MikroKosmos ontology large sets of near-synonyms are related to the same concept of ontology, their specific features are described in the lexicon [3].

The authors provide an example: all the '*change*' verbs are assigned to the same concept of *CHANGE-EVENT*. The features of the words are described in lexicon entries, for example, for the verb *to increase* it is pointed out that the *THEME* semantic role of the verb should be presented by a *SCALAR_VALUE* (for example, price or height) and the value of this quantity is changed to a larger one. The meaning of the word *Zionist* is represented in the dictionary as a *POLITICAL_ROLE*, which is an *AGENT_OF* a *SUPPORT_EVEN*T, the theme of which is Israel. The meaning of the word *to asphalt* is described as a COVER_EVENT, an instrument of which is the *ASPHALT* concept.

Using the Web site of the ontology (http://ilit.umbc.edu), we can see that the situation with the implementation of the principles is quite complicated. The concept *CHANGE_EVENT* is associated to a long list of words in the lexicon. In the list there are such words as *acclimatization, commerzialization, contamination, damage, deteriorate, improve,* and many others – there are no separate concepts for these words.

At the same time, the following concepts can be seen lower in the hierarchy: *ADJUST, CORRECT-EVENT, DIVIDE, INTEGRATE, RESTRUCTURE,* etc. It is not clear why separate concepts were introduced for some of the words, but were not introduced for the others. Why the meaning of word *acclimatization* does not deserve an independent concept, although there are important relations to climate, biological processes, but the meaning of *adjust* has received a concept?

In addition to questions of consistency/inconsistency in the description, there are clear consequences for the natural language processing applications. Thus, it is difficult to establish what words from a larger list of lexical entries to the concept of *CHANGE-EVENT* can be regarded as synonyms, and what are the relations between other words. Besides, one cannot specify the relations between, e.g., the *asphalting* and *road works*.

In addition, in domain-specific applications the relatively small size of the ontology leads to the introduction of additional concepts even for words that are already included in the lexicon.

Thus, we believe that in MikroKosmos ontology the problem of near-synonyms is being solved by overgeneralization, which can lead to problems in real subject domains. In our opinion, it is necessary to implement an additional level of concepts, which would help to divide the words more clearly, not dumping them into large chunks.

5.3 Near-Synonyms in Information Retrieval Thesauri

In information-retrieval thesauri, each descriptor, which most often corresponds to a concept, combines several ascriptors that are considered as equivalent. Ascriptors are of three sub-types [20,21]:

- actual synonyms;
- lexical variations,
- quasi-synonyms.

Lexical variants differ from synonyms in that they represent some modification of the same expression, for example, different spelling, abbreviations, etc.

Quasi-synonyms are terms, the meanings of which generally differ, but are regarded as equivalents for the purposes of the thesaurus, for example, antonyms are often regarded as quasi-synonyms (*nuclear danger – nuclear safety*). Other frequent type of quasi-synonyms is the case where some integrating type is considered as a descriptor, and its subspecies are described as ascriptors.

For example, in the thesaurus LIV of Research Service of the U.S. Congress [32] descriptor *Transplantation of organs, tissues etc*. contains such ascriptors as *medical transplantation*, *organ transplantation, skin grafting, surgical transplantation, tissue transplantation,* some of which could be considered as subordinate concepts (Skin grafting).

In the same thesaurus, the term *deflation* is included as an ascriptor in the thesaurus entry of the descriptor *inflation*, because the developers believe that these are different manifestations of a more general concept. Typically, thesauri authors prefer introducing more quasi-synonyms for concepts, regarded as peripheral to the basic domain of the developed thesaurus.

In addition, standards and guidelines for the creation of information-retrieval thesauri often recommend not to include some kinds of terms into a thesaurus.

Relatively low frequency terms can be removed from the list of candidate terms, or represented as ascriptors for more common or more frequent concepts. Too specific terms may also be excluded from the list, since it is believed that if a thesaurus contains too many hierarchy levels, it is difficult to manage. In particular indexing subjectivity increases, since indexers can use descriptors of different levels for documents indexing [21]. If several terms with similar meanings were revealed, it is necessary to choose the most representative one; the remaining terms can be partially removed and transferred to ascriptors.

Thus, in traditional information retrieval thesauri some near-synonyms are excluded from consideration, other near-synonyms are introduced as ascriptors. The exclusion of near-synonyms from a thesaurus, made for the sake of convenience and reduction of the subjectivity of manual indexing, results in decline of search quality if the information retrieval thesaurus is used in automatic modes of document processing.

Table 1 summarizes above-mentioned distinctions in concept formation approaches.

Table 1. Specific features of concept formation approaches in linguistic ontologies

Problems of concept formation	**WordNet**	**MikroKosmos**	**Information-retrieval thesauri**
Confusion of concepts and their names	Concept and names are often confused	Concept and names are rarely confused	Concept and names are rarely confused
Representation of related senses of ambiguous words	Very detailed description of senses	Related senses of ambiguous words are rarely described	No related senses
Relations between related senses	There are no relations between related senses	Related senses are generalized to the same concept	-
Near-synonyms	Sets of near-synonyms are arbitrarily split to synsets	Near-synonyms are generalized to the same concept	Near-synonyms are absent or presented as ascriptors to the same descriptor

6 Concepts and Senses in Thesaurus of Russian Language RuThes

RuThes Thesaurus of Russian language [13] can be called a linguistic ontology for the automatic text processing, i.e. an ontology, where the majority of concepts are introduced on the basis of actual linguistic expressions.

RuThes is a hierarchical network of concepts. Each concept has a name, relations with other concepts, a set of linguistic expressions, i.e., text entries (words, phrases, terms), the meanings of which correspond to the concept.

In construction of the Thesaurus we combined three different methodologies [12, 20, 21, 15, 17]:

- the methods of construction of information-retrieval thesauri (information-retrieval context, analysis of terminology, terminology-based concepts, a small set of relation types)
- the development of wordnets for various languages (word-based concepts, detailed sets of synonyms, description of ambiguous text expressions)
- ontology and formal ontology research (concepts as main units, strictness of relations description, necessity of many-step inference).

The main types of relations are taxonomic relations and a specific set of conceptual relations based on ontological dependence relations [33, 34]. This set of relations was experimentally confirmed to be effective in information-retrieval applications [35, 36].

In the following sections we will present the principles of concept description in RuThes.

6.1 Scope of Concepts in Thesaurus RuThes

Most concepts in RuThes are associated with the meanings of linguistic expressions.

Linguistic expressions that may give rise to an independent concept in the Thesaurus RuThes belong not only to the general vocabulary, but also can be terms of specific subject domains within the scope of social life (economy, law, international relations, politics), and of the infrastructure (transport, banks, etc.), so-called socio-political domain (Fig.1). This is due to the fact that many professional concepts, terms, and slang of these domains penetrate easily into the general language, and can become widely discussed in mass media [37].

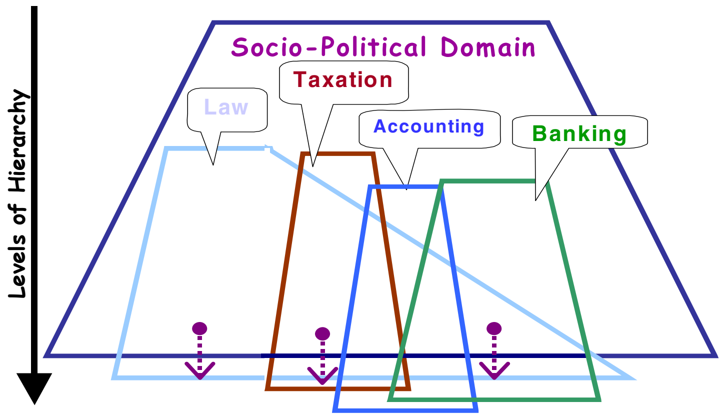


Fig. 1. Specific domains vs. Socio-political domain

Multiword expressions are also actively used as concept sources in RuThes. The basic principle of introducing this kind of concepts is the need to record some additional information that cannot be described on the basis of component word concepts.

6.2 Concept of Ontology Is Not a Synset

In RuThes, a unit is presented not by a set of similar words or terms, as it is done in the WordNet thesaurus, but by a concept – as a unit of thought, which can be associated with several synonymic language expressions.

Words and phrases, the meanings of which are represented as references to the same concepts of the thesaurus, are called text entries. Text entries of a concept can be:

- words that belong to different parts of speech (*stabilization, stabilize, stabilized*);
- linguistic expressions relating to different linguistic styles, genres;
- single words, idioms, free multiword expressions, the meanings of which correspond to this concept.

Concepts often have more than 10 text entries including single nouns, verbs, adjectives and noun or verb groups. For example, a set of English text entries of concept *JUDICIAL COURT* looks as follows:

`court, court authorities, court instance, court of judiciary, court of jurisdiction, court of justice, court of law, judicature, judicial bodies, judicial court, judicial organ, judicial tribunal, law court, tribunal.`

NATURE PROTECTION concept comprises such English expressions as:

`conservancy, conservation of nature, to conserve nature, to conserve natural environment, defense of nature, maintenance of nature, nature conservation, nature conservative, to protect nature, protection of nature`

and others.

6.3 Concept Name

To work with concepts, to analyse the results of automatic text processing, each concept should have a clear, univocal and concise name. From this point of view, the synonymic sets are not very convenient to use as concept names. Moreover, a synset can consist only of a single ambiguous word and needs additional explication.

Therefore, in the RuThes ontology, each concept has an assigned name.

Name of a concept can be:

- one of unambiguous synonyms;
- an unambiguous multiword expression;
- a pair of synonyms that uniquely identifies the concept;
- an ambiguous word with a relator similar to those used in traditional information retrieval thesauri.

If necessary, a concept may have a comment, which is not a part of the concept name. This is the usual practice in the development of traditional information retrieval thesauri.

6.4 Closely Related Senses in RuThes

The problem of similar meanings of ambiguous words, which may be very hard for compilers of explanatory dictionaries, often become even more complicated for developers of computational linguistic resources. It is often assumed that the integration of similar senses would help reducing the complexity of this problem in computational vocabularies.

However, such clustering may cause other problems.

First of all, as it was already noted, the integration of similar senses may be different depending on an intended application, e.g., machine translation, information extraction, or information retrieval.

Secondly, if we look at the example with word *window*, the problem of confusion of the name of the concept and the actual concept itself reappears. For example, *WINDOW (OPENING)* and *WINDOW (ARTIFACT)* are distinctly different entities, which arose and existed for some time independent of each other. Integration of different entities because of their similar names presents an example of such confusion.

As we have stated in Section 4.2 the integration of two different entities results in confusion of their relations with other concepts, which eventually may affect the logical inference.

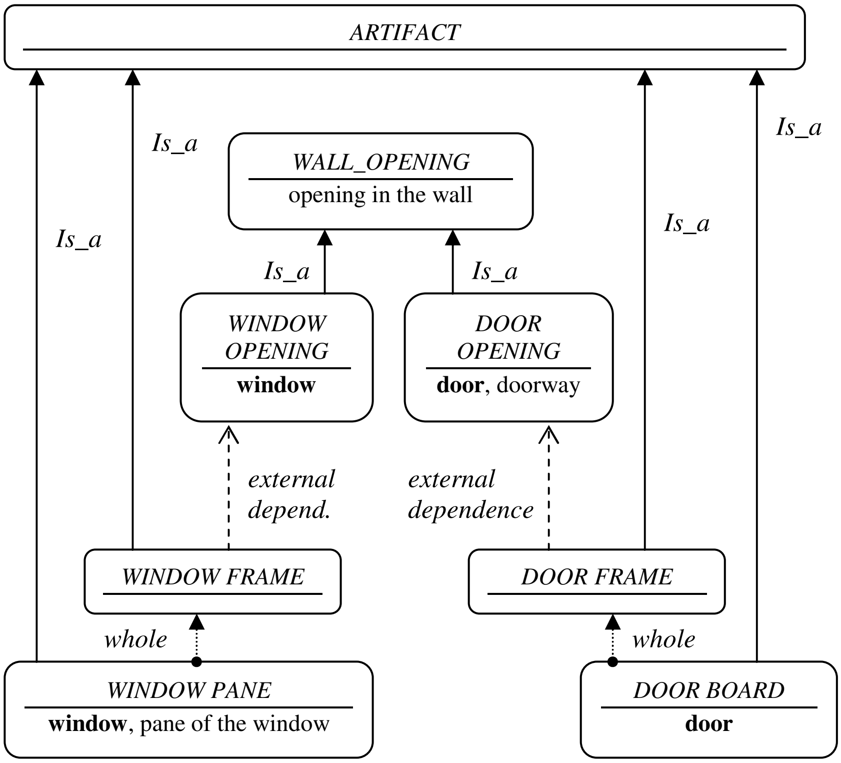


Fig. 2. Set of concepts corresponding to ambiguous words *window* and *door*

Finally, each of these independent entities can be expressed precisely and unambiguously: *WINDOW (OPENING)* can be expressed as *window opening* (351,000 pages in Google), a *WINDOW (ARTIFACT)* can be expressed as *window pane* (697,000 pages in Google). As a result of integration of the initial concepts, *window opening* has become a synonym of *window pane*.

Thus, in our opinion, if there are distinctly different entities with their own sets of relations and text entries, then they need to be represented by different concepts, even if these individual entities are closely related, and there is a word that can refer to both entities. However, since such entities as *WINDOW (OPENING)* and *WINDOW (ARTIFACT)* are closely related, there should be a relation between respective concepts.

Note that these relations should not be relations of metaphor, metonymy, homonymy, as discussed in [26] (see Section 4.1), because these are linguistic relations between the characters, not between the concepts.

In RuThes, one usually employs the *part-whole* and *external dependence* [33, 34, 35] relations to describe the relation between closely related entities, which may be named by same ambiguous words. In particular, the concept *WINDOW (ARTIFACT)* is externally dependent on concept *WINDOW (OPENING),* since to define concept *WINDOW (ARTIFACT)* one should have concept *WINDOW (OPENING)* already defined (Fig.2).

The existence of relations between the related entities simplifies disambiguation of words like *window,* since it is possible to give a default meaning and to choose it in complex cases.

6.5 Near-Synonyms in RuThes

To describe a set of related meanings of near-synonyms through a set of concepts of a linguistic ontology, the following procedure is applied in RuThes.

The first step is to identify components of the meaning that either always (regardless of the context of use) exist for at least one word of a near-synonyms set, or may occur in certain contexts for several words of a set. In a set of words that are close to the word *similarity* (see respective WordNet synsets in Section 5.1), this element of the meaning is, for example, the *similarity of the external characteristics*:

Likeness, alikeness, similitude *– (<u>similarity in appearance or</u> character or nature between persons or things)*

Resemblance *– (<u>similarity in appearance</u> or external or superficial details).*

The notation suggests that '*similarity in appearance'* meaning is significant for people, and this fact should be reflected in the respective concept.

The second step is to find a suitable name for such a concept. In the case of near-synonyms to the word *similarity*, the name of such a concept could be *SIMILARITY IN APPEARANCE* (34,700 pages in the Google). The concept is introduced in the thesaurus with the chosen name.

The next step is to find different ways of expressing the same concept in the form of phrases and single words, e.g., *resemblance in appearance, similarity of appearance, external resemblance,* etc. All these variants are added as text entries to the concept description.

To reflect the meaning of the words that often express this concept in particular, but can also be used to express the *similarity* in whole, e.g., *resemblance*, this word is referred to as a text entry to the concept *SIMILARITY IN APPEARANCE*, and as a text entry to the general concept *SIMILARITY.*

Fig. 3 presents the resultant set of concepts and their text entries. Such words as *resemblance* and *likeness* are described as text entries for two different concepts because of their vague meaning. Thus, we are trying to create context-independent, distinguishable concepts, which are as close as possible to the linguistic meanings.

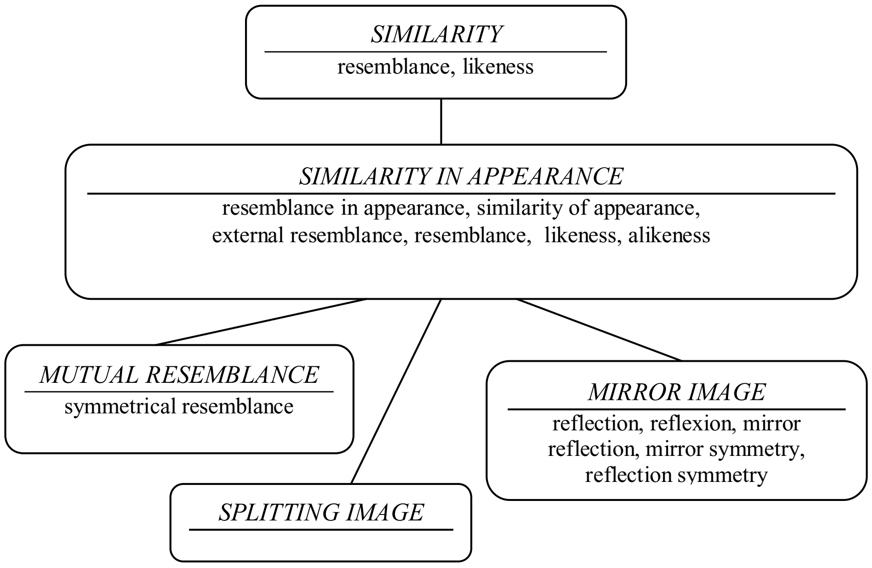

SIMILARITY
resemblance, likeness
SIMILARITY IN APPEARANCE
resemblance in appearance, similarity of appearance, external resemblance, resemblance, likeness, alikeness
MUTUAL RESEMBLANCE
symmetrical resemblance
SPLITTING IMAGE
MIRROR IMAGE
reflection, reflexion, mirror reflection, mirror symmetry, reflection symmetry

Fig. 3. Formation of distinguishable concepts for near-synonyms of word *similarity*

The analysis of word meanings is similar to feature-based analysis that is often accomplished with Formal Concept Analysis methods [38]. But we try to introduce additional taxonomic categories on the basis of existing language expressions.

In the analysis above it was not necessary to use representations of similar meanings in other natural languages. However, taking into account the examples from another language may be very helpful for recognition of poorly distinguishable concepts.

7 Recommendations for Developers of Formal Ontologies

As we argue that concepts of an ontology cannot be fully separated from natural language meanings we can formulate several recommendations that follow from our consideration of concept-meaning interactions in linguistic ontologies.

In applying Formal Concept Analysis to ontology development one encounters the problem of huge number of taxonomic categories (formal concepts) automatically obtained from a formal context [39, 40]. An additional condition of choosing the most important concepts can be based on existence of words or multiword expressions with corresponding meanings. For example, in the lattice from [39] (Fig. 4), at least four of six intermediate concepts can be named using existing phrases: *BRAIN SIGNAL (39), MOUSE BRAIN (38), SIGNAL PATHWAY (77), VERTEBRATE BRAIN (36).*

Developers of an ontology can deal with related entities named by same ambiguous word and, therefore, it can be difficult to distinguish these entities. In these case it can

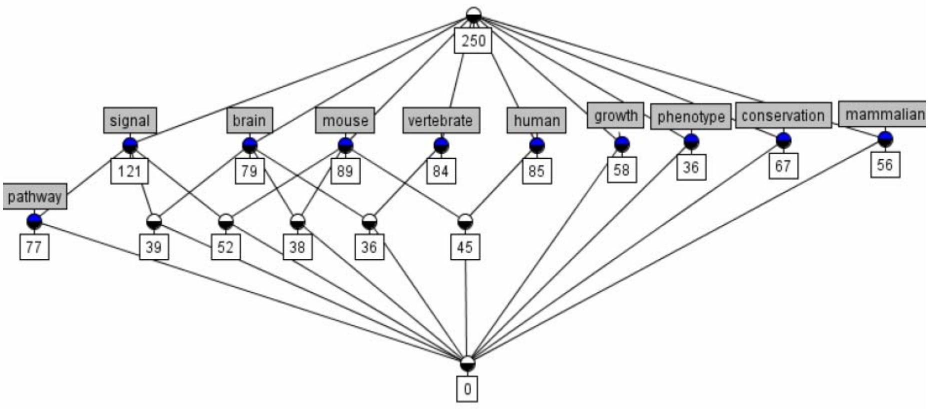


Fig. 4. Some of intermediate concepts in the example from [38] have linguistic names

be helpful to find multiword expressions including the same word (or derivative words) and synonymous to this ambiguous word, which can clarify these meanings. We described this procedure for word *window* in Section 6.4 and used such expressions as *window pane* and *window opening*.

Another example is ambiguous word *"nation"*. This word can denote both *political nation,* which means *'state, country',* and *ethnic nation,* which means *'ethnicity'*. So we can see different entities behind this word and the procedure does not require professional linguistic competence.

At last if an entity seems to have unstable attributes which appear or disappear in different contexts it is possible that the language ambiguity takes place. In such cases it is helpful to use more stable concepts fixing the attributes. We used this procedure in Section 6.5 introducing concept *SIMILARITY IN APPEARANCE*. And again we use a multiword expression to reveal and fix this concept.

8 Conclusion

Ontology developers can hardly avoid the influence of linguistic meanings, linguistic polysemy, since the names of concepts and relations in ontologies have mnemonic names, the knowledge in many subject fields is hidden in texts.

Therefore, it is important for ontology developers to understand the problems related to the formation of concepts on the basis of linguistic meanings, namely:

- The problem of distinguishing the concept and its name.
- The problem of presenting closely related meanings of ambiguous words.
- The problem of splitting meanings of near-synonyms into concepts.

In developing the RuThes as a linguistic ontology we are trying to adhere to two, generally speaking, contradictory criteria.

On the one hand, we form concepts of the thesaurus as close as possible to the meanings of linguistic units. As practice has shown, the excessive generalization and clustering of meanings necessarily lead to distortions in the system of relations, to problems in a specific subject field, or an application.

On the other hand, we try to ensure that a concept is still a concept, i.e. it is, at least, distinguishable from the superordinate concept and the sibling concepts.

Exploitation of really existing multiword expressions helps us mitigate these contradictory requirements. The introduction of concepts on the basis of multiword expressions does not only change the essence of a linguistic ontology, but also makes the distinction between the concepts much clearer.

For the concept of ontology, which is clearly distinguishable from other concepts, it is much easier to find equivalents in another language in the form of single words or multiword expressions. So distinguishable concepts do a linguistic ontology more language-independent.

Taking into account the examples from other language(s) is very useful for recognition of poorly distinguishable concepts.

References

1. Staab, S., Studer, R.: Handbook on Ontologies. Birkhäuser, Basel (2004)
2. Nirenburg, S., Wilks, Y.: What's in a symbol: Ontology, representation, and language. J. of Experimental and Theoretical Artificial Intelligence. 13(1), 9–23 (2001)
3. Nirenburg, S., Raskin, V.: Ontological Semantics. MIT Press, Cambridge (2004)
4. Lenat, D., Miller, G., Yokoi, T.: CYC, WordNet, and EDR: critiques and responses. Communications of the ACM 38(11), 45–48 (1995)
5. Wilks, Y.: The Semantic Web as the apotheosis of annotation, but what are its semantics? IEEE Intelligent Systems 23(3), 41–49 (2008)
6. Brewster, C., Iria, J., Ciravegna, F., Wilks, Y.: The Ontology: Chimaera or Pegasus. In: Dagstuhl Seminar Machine Learning for the Semantic Web, pp. 89–101 (2005)
7. Tsujii, J., Ananiadou, S.: Thesaurus or logical ontology, which one do we need for text mining? In: Language Resources and Evaluation. Springer Science and Business Media B.V., vol. 39(1), pp. 77–90 (2005)
8. Gene Ontology, `http://www.geneontology.org/`
9. Magnini, B., Speranza, M.: Merging Global and Specialized Linguistic Ontologies. In: Proceedings of OntoLex 2002, pp. 43–48 (2002)
10. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: OntoWeb. Technical Roadmap. D.1.1.2. - IST project IST-2000-29243 (2001)
11. Sowa, J.: Building, Sharing and Merging Ontologies, `http://www.jfsowa.com/ontology/ontoshar.htm`
12. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Five papers on WordNet. CSL Report 43, Cognitive Science Laboratory, Princeton University (1990)
13. Loukachevitch, N., Dobrov, B.V.: Development and Use of Thesaurus of Russian Language RuThes. In: Proceedings of workshop on WordNet Structures and Standartisation, and How These Affect WordNet Applications and Evaluation, Gran Canaria, Spain, pp. 65–70 (2002)
14. Noy, N.F., McGuinness, D.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 (2001)

15. Bouaud, J., Bachimont, B., Charlet, J., Zweigenbaum, P.: Methodological principles for structuring an “ontology”. In: Proceedings of IJCAI 1995 Workshop on Basic Ontological Issues in Knowledge Sharing (1995)
16. Miller, G., Hristea, F.: WordNet Nouns: Classes and Instances. J. Computational linguistics 32(1), 1–3 (2006)
17. Guarino, N.: Some Ontological Principles for Designing Upper Level Lexical Resources. In: Proceedings of First International Conference on Language Resources and Evaluation, Granada, Spain, pp. 28–30 (1998)
18. Climent, S., Rodriguez, H., Gonzalo, J.: Definitions of the links and subsets for nouns of the EuroWordNet project. Deliverable D005, WP3.1, EurWordNet, LE2-4003 (1996)
19. Welty, C., McGuinness, D., Uschold, M., Gruninger, M., Lehmann, F.: Ontologies: Expert Systems all over again. In: AAAI 1999 Invited Panel Presentation (1999)
20. ISO 2788-1986 - Guidelines for the establishment and development of monolingual thesauri (1986)
21. Z39.19 – Guidelines for the Construction, Format and Management of Monolingual Thesauri. NISO (2003)
22. Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., Katz, S.: Reengineering Thesauri for New Applications: the AGROVOC Example. J. of Digital Information, Article No. 257 (2004)
23. Chugur, I., Gonzalo, J., Verdejo, F.: Polysemy and sense proximity in the Senseval-2 Test Suite. In: Proceedings of the ACL 2002 Workshop on Word sense Disambigation:recent successes, pp. 32–39 (2002)
24. Agirre, E., Lacalle, L.O.: Clustering Wordnet word senses. In: Proceedings of RANLP 2003 (2003)
25. McCarthy, D.: Relating WordNet Senses for Word Sense Disambiguation. In: Proceedings of NACCL Workkshop on Making Senses of Sense, pp. 17–24 (2006)
26. Gonzalo, J.: Sense Proximity versus Sense Relations. In: Proceedings of International Wordnet Conference (GWC 2004), pp. 5–6 (2004)
27. Fellbaum, C., Miller, G.: Whither WordNet. Presentation of winners of Antonio Zampolli Award. LREC 2006 (2006), `http://www.lrec-conf.org/lrec2006/article.php3?id_article=45`
28. Landes, S., Leacock, C., Tengi, R.I.: Building semantic concordances. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database. The MIT Press, Cambridge (1998)
29. Hovy, E.: Methodologies for the Reliable Construction of Ontological Knowledge. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS (LNAI), vol. 3596, pp. 91–106. Springer, Heidelberg (2005)
30. Hirst, G.: Ontology and the Lexicon. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies in Information Systems, pp. 209–230. Springer, Berlin (2003)
31. Edmonds, P., Hirst, G.: Reconciling fine grained lexical knowledge and coarse-grained ontologies in representation of near-synonyms. In: Proceedings of Workshop on Semantic Approximation, Granularity and Vagueness, pp. 12–22 (2000)
32. LIV (Legislative Indexing Vocabulary). Congressional Research Service. The Library of Congress. Twenty-first Edition (1994)
33. Guarino, N., Welty, C.: A Formal Ontology of Properties. In: Dieng, R., Corby, O. (eds.) EKAW 2000. LNCS, vol. 1937, pp. 97–112. Springer, Heidelberg (2000)
34. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social roles and their descriptions. In: Dubois, D., Welty, C., Williams, M.A. (eds.) Principles of Knowledge Representation and Reasoning, Proceedings of the Ninth International conference KR 2004, pp. 267–277. AAAI Press, Menlo Park (2004)

35. Loukachevitch, N.V., Dobrov, B.V.: Development of Ontologies with Minimal Set of Conceptual Relations. In: Lino, M.T., Xavier, M.F., Ferreira, F., et al. (eds.) Proceedings of Fourth International Conference on Language Resources and Evaluation, vol. 6, pp. 1889–1892 (2004)
36. Loukachevitch, N.V., Dobrov, B.V.: Ontological Types of Association Relations in Information Retrieval Thesauri and Automatic Query Expansion. In: Oltramari, et al. (eds.) Proceedings of OntoLex 2004: Ontologies and Lexical Resources in Distributed Environments, pp. 24–29 (2004)
37. Loukachevitch, N., Dobrov, B.: Sociopolitical Domain as a Bridge from General Words to Terms of Specific Domains. In: Proceedings of Second International WordNet Conference GWC, pp. 163–168 (2004)
38. Priss, U.: Linguistic Applications of Formal Concept Analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 149–160. Springer, Heidelberg (2005)
39. Kuznetsov, S., Obiedkov, S., Roth, C.: Reducing the Representation Complexity of Lattice-Based Taxonomies. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS, vol. 4604, pp. 241–254. Springer, Heidelberg (2007)
40. Cimiano, P., Stumme, G., Hotho, A., Tane, J.: Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies. In: Eklund, P. (ed.) ICFCA 2004. LNCS, vol. 2961, pp. 189–207. Springer, Heidelberg (2004)

Conceptual Graph Rules and Equivalent Rules: A Synthesis

Marie-Laure Mugnier

LIRMM (CNRS - University of Montpellier),
161, rue Ada, F-34392 Montpellier cedex, France
mugnier@lirmm.fr

Abstract. This paper is an extended abstract of the talk given at ICCS'09. Rules have long been considered as an essential component of knowledge-based systems. We focus here on conceptual graph rules and on the semantically equivalent knowledge constructs in logic and databases, namely rules with existential variables and tuple-generating dependencies. The aim of this presentation is to synthesize main decidability, complexity and algorithmic results obtained on this kind of rules. We emphasize the fact that the graph vision of rules has led to new results.

1 Introduction

Rules have long been considered as an essential component of knowledge-based systems. In this talk, we focus on rules in conceptual graphs (CG) and on the equivalent knowledge constructs in logic and databases. For precise definitions of all conceptual graph notions used in this presentation, we refer to [CM08].

A conceptual graph rule (in short $R : H \rightarrow C$) can be seen as a pair (H, C) of basic conceptual graphs, provided with a one to one correspondence between a subset of generic nodes in H and a subset of generic nodes in C. H and C are respectively called the *hypothesis* and the *conclusion* of the rule. The distinguished nodes in H and C are called *connection nodes*. Figure 1 shows a CG rule (pictured with Cogui[1]). The correspondence between connection nodes is visualized by dotted lines. The logical translation of this rule is $\forall x \forall y (Person(x) \wedge Person(y) \wedge siblingOf(x, y) \rightarrow \exists z (Person(z) \wedge Parent(z, x) \wedge Parent(z, y))$. This kind of logical rule is more general than the (positive) rules usually considered in logic programming or deductive databases. Indeed, there might be variables in the conclusion which are *existentially* quantified, hence the name $\forall\exists$-rule given to this kind of formula in [BLMS09].

A $\forall\exists$-rule has the same form as a very general kind of dependency studied in databases called tuple-generating dependency (TGD) [AHV95]. It can also be seen as an abstraction for ontological knowledge expressed in specific knowledge representation languages, f.i. the RDFS rules [Hay04], constraints in F-logic-Lite [CK06][CGK08], as well as some kinds of inclusions in description logics [BCM$^+$03].

[1] http://www.lirmm.fr/cogui

S. Rudolph, F. Dau, and S.O. Kuznetsov (Eds.): ICCS 2009, LNAI 5662, pp. 23–31, 2009.

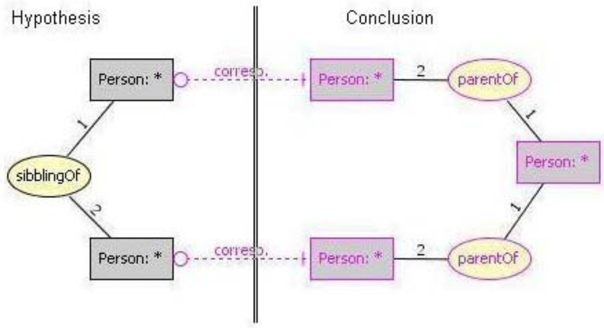


Fig. 1. A conceptual graph rule

Let us point out that, in conceptual graphs, concept types and relations are ordered by a specialization relation, and that the processing of this order is directly integrated in conceptual graph mechanisms. This feature does not add expressivity with respect to ∀∃-rules, since specialization orders can be translated into simple rules of form $\forall x_1...x_k(t_1(x_1...x_k) \rightarrow t_2(x_1...x_k))$, where t_1 is a specialization of t_2, $k = 1$ if t_1 and t_2 are concept types, otherwise k is the arity of the relations. However, the specialization orders are managed with simple label comparisons, which leads to more efficient knowledge processing mechanisms.

The aim of this presentation is to synthesize theoretical and algorithmic results obtained on conceptual graph rules, as well as on the semantically equivalent knowledge constructs in logic and databases, namely ∀∃-rules and tuple-generating dependencies. We emphasize the fact that the graph vision of rules has led to new results.

2 Deduction with CG Rules

A conceptual graph vocabulary, also called *support*, contains finite ordered sets of concept types and of relations (as well as a set of individual markers, relation signatures, assertions of disjointness between concept types, ...). It can be seen as a very basic ontology. Basic conceptual graphs (BG) are used to express facts and queries. They are logically translated into existentially closed conjunctions of atoms. A BG itself can express a boolean database conjunctive query (i.e. with a yes/no answer) and, when generic concept nodes are distinguished to represent the answer part of the query, it is equivalent to a general conjunctive query. Let us consider conceptual graph knowledge bases (KBs) composed of a vocabulary, a set of facts (which can also be seen as a single fact) and a set of rules. Several fundamental problems on these KBs are computationally equivalent, namely *fact deduction* (is a fact deducible from a KB?), *rule deduction* (is a rule deducible from a KB?) and *boolean conjunctive query answering* in presence of incomplete knowledge (is a boolean conjunctive query deducible from a KB ?). Very simple polynomial reductions allow to go from one problem to another. Since a fact can be seen as a rule with an empty hypothesis, fact deduction is a particular case of rule deduction. In turn, rule deduction can be reduced to fact deduction. The following transformation comes from [BV84] (and was applied to TGDs). Let K be the KB and R be the rule for which we want to know if it is deducible from the KB. Let $R' : H' \rightarrow C'$ be obtained from R by replacing, in each pair of corresponding connection nodes, the generic marker by a new individual marker that does not appear in K nor R. Let K' be obtained from K by adding the new fact H'. Then, R is deducible from K if and only if the fact C' is deducible from K'. Since a fact has the same form as a boolean conjunctive query, the equivalence of fact deduction and boolean query answering is immediate. From now on, we focus on *fact deduction*, which we simply call DEDUCTION (and all results concerning this problem can be immediately recast in terms of the other problems).

There are two classical ways of processing rules. *Forward chaining* starts from the facts and applies rules to facts to produce new facts. A derivation is a sequence of rule applications leading from an initial fact to an enriched fact. *Backward chaining* starts from a question, usually called a goal, and tries to build a derivation leading to an answer to this goal in a backward manner. We assume that the reader is familiar with both paradigms.

Conceptual graph rules are provided with sound and complete forward and backward chaining mechanisms, which operate directly on their graph form. For forward chaining, the basic notion is the BG homomorphism, classically called projection in the CG community (however, we do prefer to use the term homomorphism because it relates this notion to relational algebra and graph theory; moreover, the CG projection may be confused with the projection operator in relational algebra). The fundamental property is that BG homomorphism is sound and complete with respect to logical deduction [CM92]: given two BGs G and H built on a vocabulary $\mathcal{V}$, there is a homomorphism from G to H if and only if $\Phi(G)$ can be deduced from $\Phi(H)$ and $\Phi(\mathcal{V})$ (for the completeness part, either H has to be in a normal form, or a variant of homomorphism can be used to avoid this normality condition [CM04]). Homomorphism checking is NP-complete. A rule $R : H \rightarrow C$ can be applied to a fact F if there is a homomorphism h from H to F. Applying R to F according to h consists of "adding" C to F in a way defined by h (each connection node in C is merged with the image by h of the corresponding connection node in H). This yields a sound and complete mechanism: given a KB K composed of a vocabulary $\mathcal{V}$, a fact F and set of rules $\mathcal{R}$, and a query Q (also built on $\mathcal{V}$), there is a derivation from F to a fact F' using rules of $\mathcal{R}$, and a homomorphism from Q to F', if and only if $\Phi(Q)$ can be deduced from $\Phi(K)$ (i.e. $\Phi(\mathcal{V}) \cup \Phi(\mathcal{R}) \cup \{\Phi(F)\}$) [SM96].

Backward chaining relies on a *unification* operation, between part of a current goal and a rule conclusion. In logic programming, the conclusion of a rule consists of a single atom, thus unification involves one atom of the goal and the atom of a conclusion. [GW95] proposed a sound and complete backward mechanism for conceptual graph rules very similar to this mechanism. The goal is split into trivial subgraphs composed of a relation node and its neighbors. Then, unification involves a trivial subgraph of the goal and an atom of the conclusion. In [SM96], a more complex unification operation is defined, which aims at exploiting the complex structure of a conceptual graph rule: it allows one to process conclusions and goals without decomposing them into trivial subgraphs. This mechanism will be detailed in section 4.

It is easily checked that forward chaining may not halt, even with criteria avoiding redundant applications of rules. Backward chaining may not halt either. The fundamental reason is that DEDUCTION is not decidable, but only semi-decidable. This has been first proven for TGDs in [BV84]. Two other proofs for CGs can be found in [Bag01] (with a reduction from the halting problem of a Turing machine, which proves that DEDUCTION with rules is a computation model) and in [BM02] (with a reduction from the word problem in a semi-thue system). It thus important to define large and meaningful cases in which the problem is decidable. Decidable cases may be defined by an *abstract* property which guarantees decidability. However, such an abstract property is generally not provided with a finite procedure allowing to determine whether a given set of rules has the property or not. The next step is thus to exhibit *concrete* cases, which fulfill the abstract property and can be recognized by a finite procedure. The conditions defining concrete cases may be relative to each rule independently or to a set of rules.

Obviously, if the forward chaining is guaranteed to halt with a kind of rules, then DEDUCTION is decidable in this particular case. This leads to the following abstract property: a set of rules is called a *finite expansion* set if it is guaranteed, for any fact, that after a finite number of rule applications, all further rule applications will become redundant, i.e. will produce facts equivalent to the current fact; DEDUCTION is decidable for finite expansion sets of rules [BM02]. Two concrete cases of finite expansion sets of rules are range-restricted rules and disconnected rules. A *range-restricted* rule is such that all concept nodes of the conclusion are either connection nodes or nodes with an individual marker (in logical terms: it is a $\forall\exists$-rule without existentially quantified variable in the conclusion, which corresponds to a range-restricted rule in positive Datalog). A *disconnected* rule has no connection nodes. DEDUCTION with a set of range-restricted rules or a set of disconnected rules is NP-complete (assuming that the arity of relations is bounded), thus is in the same complexity as DEDUCTION with facts only, which involves a simple homomorphism test.

In [BLMS09], a similar abstract property is exhibited in relation with backward chaining. Given a goal Q and a rule R, a *rewriting* of Q with R is a graph obtained by a unification of Q with the conclusion of R. A set of rules $\mathcal{R}$ is called a *finite unification* set if it is guaranteed, for any goal, that there is a finite set $\mathcal{Q}$ of rewritings of Q with rules in $\mathcal{R}$, such that any other possible rewriting of Q is more specific than an element of $\mathcal{Q}$. Two concrete cases of finite unification sets are exhibited in [BLMS09]: atomic hypothesis rules and domain restricted rules. In an *atomic hypothesis rule*, the hypothesis contains a single atom. These rules are particularly well adapted to express necessary properties of concepts or relations in ontological languages, without any restriction on the form of the conclusion (i.e. rules of form $C(x) \rightarrow P$ or $r(x_1 \dots x_k) \rightarrow P$, where C is a concept type, r a k-ary relation and P any set of atoms). The second kind of rules does not put any restriction on the form of the hypothesis but constrains the form of the conclusion: in a *domain restricted* rule, each atom of the conclusion contains all or none of the variables in the hypothesis. The complexity of DEDUCTION for these particular kinds of rules has not been studied yet.

Other decidable cases are not based on individual properties of rules but on interactions between rules and will be presented in the next sections.

3 Equivalent Problems in Databases

Tuple-generating dependencies (TGDs) are a very general class of dependencies, encoding most dependencies in databases [AHV95]. They have exactly the same logical form as ∀∃-rules. If a TGD is not satisfied by a database instance, it is possible to repair the database instance by extending it with new atoms. The procedure that enforces the validity of a set of TGDs is called the *chase*: it is equivalent to forward chaining. The chase was first introduced for the *TGD implication problem*: given a set of TGDs T, and a TGD t, is t implied by T? (this problem is the same as the above rule deduction problem). A related problem is the *query containment problem* under a set of TGDs: given a set of TGDs T, and two conjunctive queries q_1 and q_2, is the the set of answers to q_1 included in the set of answers to q_2 for any database satisfying T (i.e. satisfying each TGD in T) ? A problem more recently introduced is *query answering on incomplete data* [CLR03]: given a set of TGDs T, a database instance D, that may not satisfy T, a conjunctive query q and a tuple of values t, is t an answer to q in a database instance obtained from D by enforcing T ? All these problems can be proven equivalent to DEDUCTION.

Interestingly, very recent results have exhibited classes of TGDs for which the problem is decidable even if the chase does not halt [CGK08]. The abstract property is that when all facts generated by a set of rules have a "bounded treewidth" then DEDUCTION is decidable. Note that this class of rules includes finite expansion sets, but not finite unification sets. Concrete cases of rules satisfying this abstract property are the guarded TGDs, and their generalization to weakly guarded TGDs. A TGD is *guarded* if its body (i.e. hypothesis) includes an atom, called guard, that contains all variables occurring in the body. *Weakly guarded* TGDs are an extension of guarded TGDs that requires guards to contain only some variables in the body (see [CGK08] for a precise definition). This property cannot be checked independently on each rule, but requires to consider the whole set of rules. It is shown that the problem is EXPTIME-complete (with the assumption of bounded predicate arities) for weakly guarded TGDs, and even for guarded TGDs. It becomes NP-complete when, moreover, the number of predicates appearing in the TGDs is bounded.

4 Graph Rules: The Added Value

In this section, we focus on backward chaining and on how the graph structure allows to obtain new results. Graphs are a natural construct for representing complex structures. In previously cited results on TGDs, conclusions of rules are restricted to one atom, as it is usually the case in logic programming. This restriction does not lead to a loss of expressivity since any set of rules can be rewritten (in linear time) as a set of rules with one atom in conclusion. Indeed, a rule $H \rightarrow C$ can be equivalently encoded by a set of rules $\{H \rightarrow R(t_1, ..., t_k), (R(t_1, ..., tk) \rightarrow A_c)_{A_c \in C})\}$, where R is a new predicate assigned to the rule and $t_1...t_k$ are the terms occurring in C. However, beside loss in readability, this rewriting leads to a loss in efficiency and weaker decidability results [BLMS09].

The sound and complete backward chaining outlined in this section is based on the notions of a *piece* and the associated *piece-based unification* [SM96]. Let us mention that these notions have been defined for conceptual graph rules obeying two constraints. First, two corresponding connection nodes have the same type. Secondly, an individual marker always occurs with the same concept type. As a consequence of these restrictions, a rule application to a fact F never restricts labels of existing nodes in F. It only adds new nodes to F. These restrictions do not lead to a loss in expressivity in the sense that concept types can be equivalently represented as unary relations. However, to work directly on general conceptual graph rules, the notions of piece and piece-based unification presented hereafter would need to be extended. These restrictions do not apply in a logical setting, since there is no distinction between concept types and relations, which are all predicates.

A *cutpoint* of a rule is either a connection node or a node with an individual marker. A *piece* of a rule is a (non empty) subgraph of its conclusion, in which any two nodes are connected by a path that does not go *through* a cutpoint (however, a cutpoint can be an extremity of such a path), and to which no more nodes can be added while preserving this property. A way of understanding pieces is as follows: assume that all cutpoints of the rule conclusion are deleted; each connected component obtained after this deletion belongs to a separate piece; each piece itself is obtained from such a connected component by adding again the cutpoints linked to its nodes (if any) as well as the associated edges. For instance, the rule in Figure 1 has two cutpoints and a single piece. These graph notions can be translated into logical notions in a straightforward way (see [BLMS09]). For instance, the rule $R = \forall x \forall y (p(x, y) \rightarrow \exists z \exists t \exists u (p(x, z) \wedge p(z, t) \wedge p(t, x) \wedge p(x, u)))$ has one cutpoint, which is x, and two pieces defined by the sets of atoms $\{p(x, z), p(z, t), p(t, x)\}$ and $\{p(x, u)\}$.

The idea behind piece is that a piece can be seen as a "unit" of knowledge brought by a rule application in forward chaining. Indeed, a rule R can be decomposed into an equivalent set of rules with the same hypothesis and exactly one piece in conclusion. More precisely, any rule $R : H \rightarrow C$, such that C contains k pieces $C_1, \ldots, C_k$, is equivalent to the set of rules $\{H \rightarrow C_i\}_{1 \leq i \leq k}$. Moreover, the conclusions of these rules cannot be further decomposed while keeping a set of conceptual graph rules with the same semantics as R (provided of course that H is not modified, otherwise see the above decomposition).

We do not recall the definition of piece-based unification, which would require more technical developments (see [SM96] [CM08] in the CG framework, and [BLMS09] for $\forall\exists$-rules). The important point for the backward chaining mechanism is that piece-based unification allows it to be guided by the structure of rules and goals. An experimental comparison between piece-based backward chaining and Prolog resolution was led in [CS98].

Another important point is that it allows to characterize exactly the notion of *dependency* between rules. Generally speaking, compiling a knowledge base consists in preprocessing it off-line, so that the compiled form obtained can be used on-line to accelerate reasoning tasks (e.g. query answering). Concerning rules, a classical compilation technique consists in precomputing a graph encoding dependencies between rules. This technique allows us to improve the efficiency of forward and backward chaining mechanisms.

Given rules R_1 and R_2, R_2 is said to depend on R_1 if the application of R_1 on a fact may trigger a *new* application of R_2, i.e. if there exists a fact F to which R_1 can be applied leading to a fact F', such that there is a homomorphism from the hypothesis of R_2 to F', that is not a homomorphism from the hypothesis of R_2 to F. It is easy to define *necessary* conditions for a rule to depend from another: f.i. if R_2 depends on R_1 then there is an atom in H_2 that can be unified (in the logical classical meaning) with an atom in C. Characterizing dependency by effectively computable *necessary and sufficient* conditions is less obvious.

Piece-based unification yields such an effective characterization: R_2 depends on R_1 if and only if there is a piece-based unification of H_2 with R_1 (see [Bag04]) for an equivalent characterization restricted to rules without constants, [BS06] for this result on conceptual graph rules, and [BLMS09] for this result in a logical framework).

Given a set of rules $\mathcal{R}$, the *graph of rule dependencies* (GRD) of $\mathcal{R}$ is the directed graph with node set $\mathcal{R}$, and such that there is a (directed) edge from R_1 to R_2 if and only if R_2 depends on R_1 ("an application of R_1 may trigger a new application of R_2"). As far as we know, piece unification yields the first effective characterization of this graph. Very recently, in the context of databases, [DNR08] defined a notion equivalent to the GRD on TGDs ("the chase graph"), but no constructive characterization of this graph was provided in this paper.

The GRD has two interests. It allows to speed up forward or backward chaining and it leads to new decidability results. About the first point, a simple use of the GRD is the following. Assume that the set of facts is considered as a single graph, say F, and classified in the GRD as a rule with an empty hypothesis. F is necessarily a source (node without ingoing edge) in the GRD. The query or goal, say Q, can also be classified in the GRD as a rule with an empty conclusion. Q is necessarily a sink (node without outgoing edge) in the GRD. Then, to answer a given Q, the only rules to consider are the rules corresponding to nodes in the GRD on a path from F to Q. Let us consider a basic forward chaining algorithm, that proceeds in a breadth-first way, i.e. at each step it computes all new (and non redundant) rule applications w.r.t the current fact, then applies them producing a new fact. The rules to consider in the first step are the successors of F in the GRD. Then, the rules to consider at a given step i, $i > 1$, are the rules successors of rules applied at step $i - 1$. For further improvements of the forward and backward chaining mechanisms, see [Bag04] (forward chaining) and [BS06] (both mechanisms).

Concerning decidability results, the structure of the GRD provides information of how the rules interact with each other. It can be easily checked that if the GRD has no circuit (including no loop), then DEDUCTION is decidable. This result can be extended to a GRD in which each strongly connected component[2] is a finite expansion set of rules, i.e. circuits inside a finite expansion set of rules are allowed [Bag04]. A similar result holds for a GRD in which each strongly connected component is a finite unification set of rules [BLMS09]. Note that DEDUCTION may not be decidable in a GRD where each strongly connected component is either a finite expansion set or a finite unification set. However, there is a way of combining both notions that guarantees a finite procedure [BLMS09]: assume that the set of rules $\mathcal{R}$ can be partitioned into two sets, $\mathcal{R}_1$ and $\mathcal{R}_2$, such that $\mathcal{R}_1$ is a finite expansion set, $\mathcal{R}_2$ is a finite unification set, and there is no edge from a rule in $\mathcal{R}_2$ to a rule in $\mathcal{R}_1$ in the GRD; in this case, one can first use forward chaining on F with $\mathcal{R}_1$, which leads to a fact F'; then, backward chaining is used on F' and $\mathcal{R}_2$ to compute a set of rewritings of Q and check if there is a homomorphism from a rewriting in this set to F'; the mechanism obtained halts in all cases, and is sound and complete [BLMS09]. This result can be extended by combining it with the results in [CGK08]: see [BLMS09], which also provides a map of all known decidable cases.

Let us end by emphasizing the role of the piece notion in these results, which is a natural notion when rules are considered in their graph form because it relies on the path notion. Of course, it can be translated into a logical setting (see [BLMS09]) but it does not rely on logical notions.

5 Conclusion

In this presentation, we have synthesized main decidability and complexity results obtained on a kind of rules which takes several forms in the literature, namely conceptual graph rules, $\forall\exists$-rules and TGDs. We have shown that the graph vision of rules can lead to new notions and results. Some of the decidability results for concrete cases still have to be completed by complexity results. Further work also includes the design of forward and backward chaining algorithms exploiting as much as possible the graph of rule dependencies.

[2] Two nodes x and y are in the same strongly connected component if there is a path from x to y and a path from y to x; strongly connected components in the GRD represent maximal sets of rules that mutually depend, directly or indirectly, on each other.

References

[AHV95] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)

[Bag01] Baget, J.-F.: Représenter des connaissances et raisonner avec des hypergraphes: de la projection à la dérivation sous contraintes. PhD thesis, Université Montpellier II (November 2001)

[Bag04] Baget, J.-F.: Improving the forward chaining algorithm for conceptual graphs rules. In: KR, pp. 407–414. AAAI Press, Menlo Park (2004)

[BCM$^+$03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)

[BLMS09] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proc. of IJCAI 2009 (to appear, 2009)

[BM02] Baget, J.-F., Mugnier, M.-L.: The Complexity of Rules and Constraints. JAIR 16, 425–465 (2002)

[BS06] Baget, J.-F., Salvat, E.: Rules dependencies in backward chaining of conceptual graphs rules. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS, vol. 4068, pp. 102–116. Springer, Heidelberg (2006)

[BV84] Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. Journal of the ACM 31(4), 718–741 (1984)

[CGK08] Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: KR, pp. 70–80 (2008)

[CK06] Calì, A., Kifer, M.: Containment of conjunctive object meta-queries. In: VLDB, pp. 942–952 (2006)

[CLR03] Calì, A., Lembo, D., Rosati, R.: On the decidability and complexity of query answering over inconsistent and incomplete databases. In: PODS, pp. 260–271. ACM Press, New York (2003)

[CM92] Chein, M., Mugnier, M.-L.: Conceptual Graphs: Fundamental Notions. Revue d'Intelligence Artificielle 6(4), 365–406 (1992)

[CM04] Chein, M., Mugnier, M.-L.: Concept types and coreference in simple conceptual graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 303–318. Springer, Heidelberg (2004)

[CM08] Chein, M., Mugnier, M.-L.: Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs. In: Advanced Information and Knowledge Processing. Springer, Heidelberg (2008)

[CS98] Coulondre, S., Salvat, E.: Piece Resolution: Towards Larger Perspectives. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS (LNAI), vol. 1453, pp. 179–193. Springer, Heidelberg (1998)

[DNR08] Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: PODS, pp. 149–158 (2008)

[GW95] Ghosh, B.C., Wuwongse, V.: A Direct Proof Procedure for Definite Conceptual Graphs Programs. In: Ellis, G., Rich, W., Levinson, R., Sowa, J.F. (eds.) ICCS 1995. LNCS (LNAI), vol. 954, pp. 158–172. Springer, Heidelberg (1995)

[Hay04] Hayes, P. (ed.): RDF Semantics. W3C Recommendation. W3C (2004)

[SM96] Salvat, E., Mugnier, M.-L.: Sound and Complete Forward and Backward Chainings of Graph Rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS (LNAI), vol. 1115, pp. 248–262. Springer, Heidelberg (1996)

Two Paradigms Are Better Than One, and Multiple Paradigms Are Even Better

Arun K. Majumdar and John F. Sowa

VivoMind Intelligence, Inc.

arun@vivomind.com, sowa@bestweb.net

Abstract. During the past half century, the field of artificial intelligence has developed a large number of theories, paradigms, technologies, and tools. Many AI systems are based on one dominant paradigm with a few subsidiary modules for handling exceptions or special cases. Some systems are built from components that perform different tasks, but each component is based on a single paradigm. Since people freely switch from one method of thinking or reasoning to another, some cognitive scientists believe that the ability to integrate multiple methods of reasoning is key to human-like flexibility. In his book *The Society of Mind*, Minsky (1986) presented an architecture for intelligence based on a society of heterogeneous agents that use different reasoning methods to solve different problems or different aspects of the same problem. That idea is intriguing, but it raises many serious issues: how to coordinate multiple agents, distribute tasks among them, evaluate their results, encourage agents that consistently produce good results, inhibit agents that produce misleading, irrelevant, or unfruitful results, and integrate all the results into a coherent response. The most difficult problem is to enable multiple heterogeneous agents, acting independently, to produce the effect of a single mind with a unified personality that can pursue and accomplish coherent goals. This article discusses ways of organizing a society of heterogeneous agents as an integrated system with flexible methods of reasoning, learning, and language processing.

Keywords: agents, conceptual graphs, paradigms, reasoning, NLP.

1 Architectures for Intelligent Systems

An In the years since its founding conference in 1956, the field of artificial intelligence has generated an impressive collection of valuable components, but no comparably successful architecture for assembling them into intelligent systems. The following list illustrates the range of AI components that were designed and implemented in the 1950s and '60s:

> Parsers, theorem provers, inference engines, search engines, learning programs, classification tools, statistical tools, neural networks, pattern matchers, analogy finders, problem solvers, planning systems, game-playing programs, question-answering systems, dialog managers, machine-translation systems, knowledge acquisition tools, modeling tools, and robot guidance systems.

During the past forty years, all these systems have spawned extensions, combinations, and variations. A recent handbook covered two dozen systems of logic and knowledge representation, each with multiple versions of techniques that tend to be mutually exclusive (Harmelen et al. 2008). Various AI systems use different techniques, but few, if any, take advantage of the full range of options.

Most large systems are designed around a single paradigm, such as formal deduction, statistical language processing, or case-based reasoning. As an example, the largest knowledge-based system, Cyc (Lenat 1995), has millions of axioms, grouped in several thousand contexts or *microtheories*. To solve different kinds of problems, Cyc uses a few dozen specialized inference engines, but all of them are based on some form of deduction. Partisans of different paradigms have debated their virtues as if they were mutually exclusive, yet most of them have complementary strengths and weaknesses. There should be some way to take advantage of the best features of any or all of them when appropriate.

One way to support divergent methods within a common framework is to partition them among independent processes that run in separate modules. Any such partition would require some way to control the modules and transfer information among them. The fields of AI, computer science, and automata theory have developed several techniques:

> Lambda calculus, abstract machines, subroutines, coroutines, object-oriented protocols, message passing, associative blackboards, Petri nets, π calculus.

Of these, message passing is the most general method for information transfer, and π calculus (Milner 1999) is the most general method for combining control and message passing. Petri nets, for example, can represent single-threaded flow charts, the parallelism of coroutines, object-oriented protocols, and a wide range of asynchronous control mechanisms. Milner showed that π calculus can simulate the mechanisms of both Petri nets and lambda calculus. But π calculus goes beyond the fixed graphs of Petri nets by allowing new links to be dynamically created and destroyed. The Linda method of passing messages and control through associatively accessed blackboards (Gelernter 1985) can support π calculus by its ability to create and destroy links.

The Flexible Modular Framework™ (FMF) proposed by Sowa (2002, 2004) is an architecture for intelligent systems inspired by *The Society of Mind* (Minsky 1986), the Elephant 2000 language (McCarthy 1989), and the message-passing protocols of computer science. As in Minsky's society, each FMF module is an autonomous agent that communicates with other agents by passing messages. As in McCarthy's Elephant, messages can be expressed in logic, but a marker for the speech act indicates the sender's intention. An agent that knows a recipient's identity can send it a message directly, but an agent can find new recipients that can handle a certain kind of message by posting it to a Linda blackboard. The FMF message format has six fields:

1. **Language.** An identifier of the language used in the message. It could be a natural language, a version of logic, or any computer-oriented format.
2. **Source.** An identifier of the module or agent that sent the message.

3. **Message ID.** An identifier generated by the sender.
4. **Destination.** An identifier of the intended receiver, if known. For messages sent to an associative blackboard, this field is null; any module that responds to the message would create a new link.
5. **Speech act.** An identifier of the purpose of the message: command, question, response, assertion, estimate, diagnosis, request, promise, contract, or any other intention.
6. **Message.** Any sentence or list of sentences in the language specified by field #1.

Most message formats include most of these fields. The two characteristic features of the FMF are the null option in field #4 for an associative blackboard and the speech act in field #5, which supports an open-ended variety of interaction modes. Without those fields, the FMF can support useful subsets, such as dataflow graphs or Petri nets. With associative blackboards, the FMF can dynamically create and destroy links among agents. With speech acts, FMF agents can express a wider range of intentions than an ordinary command or query. These two fields enable the agents to discover and take advantage of an expanding and evolving range of services created by the system. They also enable agents to look for alternatives if their familiar collaborators are unable to solve an unusual kind of problem.

These formats enable the FMF to accommodate arbitrary modules, even legacy systems, by enclosing them in a wrapper that maps their inputs and outputs to FMF messages. Several variations of the FMF have been implemented, and they use a lightweight protocol that can be implemented in 8K bytes per agent. Thousands of agents can run simultaneously on a laptop computer, but they can communicate with other agents anywhere across the Internet. The messages to and from any user interface have the same six fields as all other messages in the FMF. Therefore, any user interface can be replaced, revised, or enhanced dynamically just by rerouting the messages to a different module. A version of the FMF can be implemented in any language that supports communication among multithreaded processes. At VivoMind Intelligence, Inc., several versions of the FMF were implemented in Java and a multithreaded version of Prolog. But an FMF module can send messages across the Internet to FMF modules implemented in any combination of hardware and software.

Experience in implementing and using FMF systems has shown that an architecture based on message passing among heterogeneous agents has several advantages over more conventional implementations: flexibility of adding new modules without disrupting operations by the old modules; reduction or elimination of systemic errors caused by biases in any single algorithm or paradigm; performance advantages of a lightweight protocol that can take advantage of multiple CPUs; and fail-soft redundancy, which allows most of the agents devoted to a function to continue even if one or more of them fail. Section 2 of this paper describes Minsky's proposals for a society of agents and ways of implementing them. Section 3 describes an organization of FMF agents in a managerial hierarchy that presents a unified personality to the external world. Section 4 describes the use of FMF societies for language analysis. The concluding Section 5 relates the multiple paradigms to Peirce's semiotics and the logic of pragmatism.

2 The Society of Mind

Systems of multiple agents have been proposed and implemented since the early days of artificial intelligence. But the problems of organizing multiple autonomous agents, allocating resources among them, getting them to focus on the relevant goals, and integrating many partial contributions into a unified result have been challenging:

- **Pandemonium.** Selfridge (1959) designed a system of agents called *demons*. Each demon could observe aspects of the current situation or workspace, perform some computation, and put its results back into the workspace. In effect, Pandemonium was a parallel forward-chaining reasoner. Its major drawback was that the demons generated large volumes of mostly irrelevant data that overflowed storage. A great deal of research has been devoted to measures of relevance, methods for motivating agents to produce relevant results, and ways of allocating resources to those that consistently produce the best results.
- **Rational agents.** At the opposite extreme from simple demons are rational agents that simulate a human-like level of beliefs, desires, intentions, and the ability to reason about them. Van der Hoek and Wooldridge (2008) surveyed versions of logic designed to represent groups or coalitions of such agents. Such logics may be useful for analyzing or simulating the behavior of a group of intelligent agents. But a system with human-like intelligence requires heterogeneous modules specialized for different functions, not a coalition of reasoners that all use the same logic.
- **Reactive agents.** For designing robots, Brooks (1991) noted that the major challenge was not in deliberative planning and reasoning, but in the seemingly simpler insect-like functions of perception, locomotion, and goal seeking. That observation stimulated work on reactive agents whose intelligence is at the level of ants. A society of such agents can cooperate in defending the colony, searching for food, and caring for the eggs and larvae. But no one has shown how a colony of ants could understand language or do complex reasoning and planning.

Complex rational agents and simpler reactive agents operate at different extremes of intelligence. But most systems consist of one kind or the other, not a combination of heterogeneous agents. After many years of examining different ways of designing and implementing intelligent systems, Minsky (1986) argued that no single mechanism, by itself, can adequately support the full range of functions required for a human level of intelligence:

> What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from

> any single, perfect principle. Our species has evolved many effective although imperfect methods, and each of us individually develops more on our own. Eventually, very few of our actions and decisions come to depend on any single mechanism. Instead, they emerge from conflicts and negotiations among societies of processes that constantly challenge one another. (Section 30.8)

In a review and critique of AI systems, Minsky (1991) emphasized that each of the many paradigms had made valuable contributions, but that the goal of a homogeneous system built around a single, ideal paradigm was too narrow to support the full range of human intelligence:

> The functions performed by the brain are the products of the work of thousands of different, specialized sub-systems, the intricate product of hundreds of millions of years of biological evolution. We cannot hope to understand such an organization by emulating the techniques of those particle physicists who search for the simplest possible unifying conceptions. Constructing a mind is simply a different kind of problem — of how to synthesize organizational systems that can support a large enough diversity of different schemes, yet enable them to work together to exploit one another's abilities.

In an earlier paper, Minsky (1980) proposed an administrative organization populated by "mental managers" that employ and direct other agents that perform tasks at varying levels of complexity:

> To develop this idea, we will imagine first that this Mental Society works much like any human administrative organization. On the largest scale are gross "Divisions" that specialize in such areas as sensory processing, language, long-range planning, and so forth. Within each Division are multitudes of subspecialists — call them "agents" — that embody smaller elements of an individual's knowledge, skills, and methods. No single one of these little agents knows very much by itself, but each recognizes certain configurations of a few associates and responds by altering its state.

As an example of the diversity of modules, Figure 1 shows the interconnections among the kinds of modules proposed by linguists. The large box at the bottom would contain an much larger collection of modules for all the aspects of cognition and behavior that provide the subject matter and the goals for language and reasoning.

The diversity of modules that process language is a subset of the even greater diversity in all aspects of cognition and behavior. The integration of language with every aspect of human perception, behavior, and social interaction suggests that the language modules are interconnected with other cognitive modules in dynamically

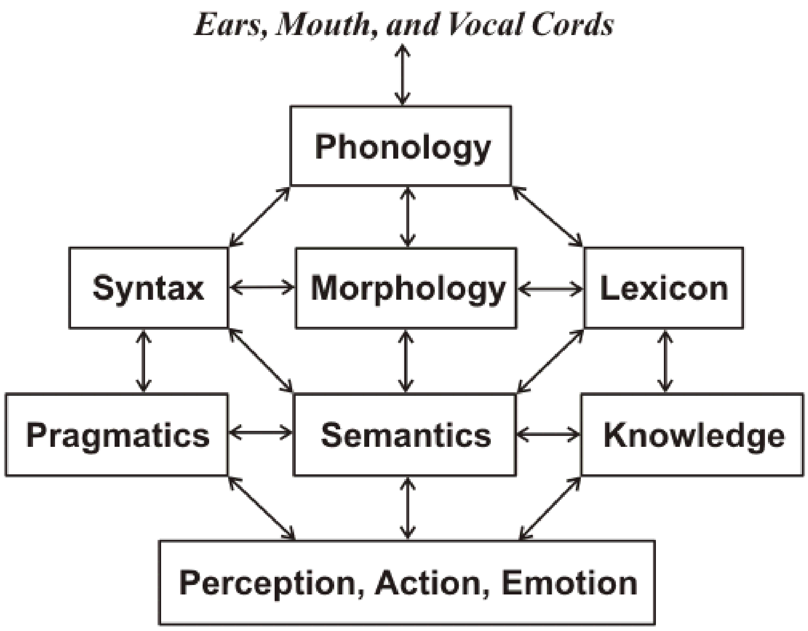


Fig. 1. Interconnections among language modules

changing ways. Whatever the organization, the number of modules is undoubtedly far greater than the eight boxes of Figure 1. Perhaps there is no limit to the number of modules, and every language game and mode of behavior has its own module or even a group of interacting modules. That organization is radically different from a homogeneous system based on a logic that cannot tolerate a single inconsistency. Minsky's goal was to build a flexible, fault-tolerant system out of imperfect, fallible components. Such a system could support logic, just as the flexible, fault-tolerant, and fallible human brain supports logic, mathematics, and every branch of science, business, and the arts. More recently, Minsky (2006) emphasized the role of emotions in driving an engine composed of multiple agents. Without emotions to set the goals, a logic-based theorem prover would have no reason to do anything.

As the underlying mechanism for implementing agents, Minsky continued his long-term research on neural networks. His newer proposals are based on knowledge lines or K-lines that pass information and control to activate agents or even a cascade of agents. In a review of Minsky's theories, Singh (2003) compared the Society of Mind to the Soar architecture for a "unified theory of cognition" (Newell 1990):

> To the developers of Soar, the interesting question is what are the least set of basic mechanisms needed to support the widest range of cognitive processes. The opposing argument of the Society of Mind theory is that the space of cognitive processes is so broad that no particular set of mechanisms has any special advantage; there will always be some things that are easy to implement in your cognitive architecture and other things that are hard. Perhaps the question we should be asking is

> not so much how do you unify all of AI into one cognitive architecture, but rather, how do you get several cognitive architectures to work together?

That question is the central theme of Minsky's book, but Singh concluded that the complexity of the ideas and the lack of detail has discouraged implementers: "While Soar has seen a series of implementations, the Society of Mind theory has not. Minsky chose to discuss many aspects of the theory but left many of the details for others to fill in. This, however, has been slow to happen."

The lack of detail plagues many proposed models of the mind. In the book *What is Thinking?* Baum (2004) surveys attempts to simulate thinking and includes a dozen citations to Minsky's *Society of Mind.* Following Minsky, he assumes "the computation of the mind is rich, with modules connected to modules, flowing in complex flow patterns" (p. 35). He views Minsky's mental managers and administrative organizations as participants in an economy guided by Adam Smith's "invisible hand" (p. 241):

> The agents in the economy will be computer programs, initially random computer programs. They will be rewarded by the economy, and the ones that go broke will be removed. New entrepreneurs will enter. Hopefully, if we get the economic structure right so that the individuals are rewarded appropriately, the system will evolve to solve hard problems... Now, we want to look at what's going on in an economy regarded as an evolutionary system consisting of a bunch of agents, each evolving to pursue its own interest, each evolving purely to increase its pay-in. We want to ensure that this evolution nonetheless promotes the overall functioning of the whole system.

Starting evolution from random computer programs would take a long time, but using economic rewards as a management tool seems promising. In fact, the economists Monderer et al. (2001) propose game theory for devising reward strategies that could motivate AI agents. A working system, however, requires much more attention to implementation detail.

3 A Hierarchy of Managers and Employees

The modules of the Flexible Modular Framework can be organized in an open-ended number of ways, and various strategies have been implemented and tested. One of the first had a graphic interface that allowed a software designer to drag and drop agents on a screen and connect them in a graph that resembles a Petri net. That was a useful tool for rapidly assembling modules, but it did not have a graphic way of showing the links found by means of associative blackboards. Another application replaced the fixed programs of an interactive game with FMF agents. The game graphics and the types of characters and machines were unchanged, but the FMF agents gave them more flexible ways of interacting, behaving, and communicating. The most general version implemented at VivoMind exploits Minsky's idea of a hierarchy of managers and employees. The chief executive officer (CEO) gives the organization a coherent "personality" for external interactions. Beneath the CEO are vice presidents in charge of major divisions, directors of important functions, lower-level managers, and specialists that perform an open-ended variety of cognitive tasks. As an example, Figure 2 shows the upper levels of a hierarchy designed to analyze and interpret natural language texts.

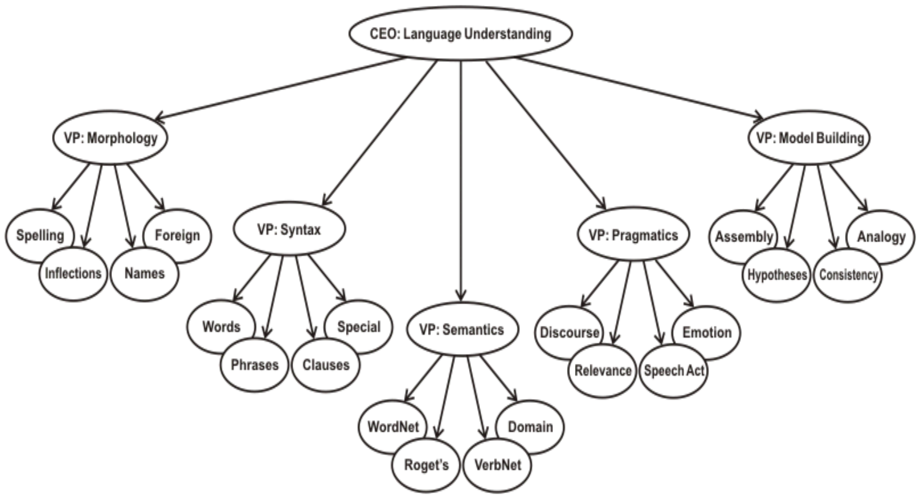


Fig. 2. A management hierarchy of language processing agents

At the top of Figure 2, the CEO of language understanding is responsible for all functions from the analysis of individual words (morphology) to the construction of a "mental model" of the meaning of an entire text, which could be a sentence, a paragraph, a conversation, a report, or a book. Reporting to the CEO are vice presidents in charge of the divisions of morphology, syntax, semantics, pragmatics, and model building. Beneath the vice presidents are directors in charge of functions such as spelling correction in the morphology division and parsing in the syntax division. The semantics division has directors of domain ontologies for the detailed axioms of the subject matter and directors of lexical resources, such as WordNet, Roget's Thesaurus, and VerbNet. For the current implementation, a formal ontology for the upper levels has not been helpful. Detailed reasoning is done with specialized ontologies for the subject matter, and the lexical resources have been adequate for mappings between English text and the specialized domain ontologies. The hierarchy shown in Figure 2 is a composite that shows the typical functions performed by the agents. Most of the implementations have more levels for middle managers, first-level managers, and specialist employees.

As in Minsky's administrative organizations, management control flows down from the CEO at the top, many messages flow up and down the hierarchy, but messages can also flow sideways across the hierarchy. In a review of the SOAR architecture, Minsky (1993) observed that the chunking mechanism of SOAR corresponds to the production of K-lines in the Society of Mind. For the VivoMind implementations, the basic knowledge representation is conceptual graphs (Sowa 2008) represented in the Conceptual Graph Interchange Format (CGIF). Chunking in conceptual graphs is implemented by defining a new concept or relation type by a *lambda abstraction* — an arbitrarily large CG in which one or more concept nodes are identified as formal parameters. A instance chunk can be defined by assigning a name to an entity described by a conceptual graph. These mechanisms can encode frequently occurring patterns of graphs in single concept or relation nodes. The names or type labels correspond to K-lines that link all occurrences of that chunk.

Minsky maintained that a system of heterogeneous agents should allow agents to use a multiplicity of languages tailored for their purposes. The language field in an FMF message supports an open-ended variety of languages, but conceptual graphs are the *lingua franca* for detailed reasoning and natural language processing. Two agents implemented in the same language, such as Prolog, can also exchange the equivalent information in their native language form. An untranslated input language can be represented by a concept node whose referent is an uninterpreted character string:

```
[EnglishSentence: "This is an example of an English sentence."]
```

Minsky (1991) claimed that an AI system should support "neat" methods based on formal logics as well as "scruffy" methods based on informal heuristics. With current technology, any translation from an unrestricted natural language is at best a useful, but scruffy approximation. Some FMF applications also use a version of Common Logic Controlled English (CLCE), which has an unambiguous mapping to conceptual graphs whose semantics are defined by the Common Logic standard (ISO/IEC 24707). Anyone who can read English can read a CLCE statement, but some training in logic is necessary to write syntactically correct CLCE. With a clarification dialog, a person who is not a trained CLCE author can work with a help facility to convert an informal English sentence to a CLCE statement that both the human and the computer can accept. For many applications, however, a scruffy translation from ordinary English can be valuable (Majumdar et al. 2008).

Singh (2003) noted the pitfalls of relying on blackboards as the primary method of communication among agents: "While the blackboard metaphor may work when there are only a few agents using the blackboard, by the time there are hundreds of agents, let alone thousands or millions, the image of them huddled around a blackboard is no longer reasonable, and in fact no one has built a blackboard system of this scale." For that reason, most FMF messages are sent to a known recipient, but an FMF system can have an open-ended number of blackboards, which may be used in various ways:

1. **Newsletter.** Any agent that manages other agents may set up a blackboard for notes that members of the department may post to any or all members of the group. The CEO might use global newsletters to announce information that could be accessed by any agents in the hierarchy, or even by unemployed "freelance" agents.

2. **Agenda.** A blackboard may serve as a queue of tasks to be done, and any available agent that can handle a task could remove it from the queue and do

it. Some kinds of jobs could be performed by multiple agents, and a manager might let more than one perform the job and select the best results.

3. **Want ads.** Sometimes a manager might post a job description to a global blackboard that could be accessed by freelance agents that might offer their services.

4. **Classified advertising.** Freelance agents might offer to sell data or hypotheses on blackboards that are specialized for a variety of purposes.

5. **Committees.** Blackboards used for collaborative reasoning would normally be restricted to a small group of agents that resemble a committee. Such a group would fit the metaphor of collaborators "huddled around a blackboard." Committees provide a collaborative environment for agents to evaluate options, vote for their preferences, or negotiate to combine them.

Variations of these five uses for FMF blackboards have been implemented in systems for processing natural language (Majumdar et al. 2008) and in a game-playing system for knowledge capture (Majumdar et al. 2007).

At VivoMind, the authors have developed a learning technique called *Market-Driven Learning*™ (MDL), which rewards agents with resources: computer space and time to perform their services. A hierarchy that reports to a CEO can earn resources by providing services to external users or systems. The CEO distributes resources as rewards to the vice presidents, who distribute their allotment to the managers that report to them. The managers can use their resources to hire employees, reward employees for good performance, or buy data and hypotheses from freelance agents or from other managers. The managers may combine the data and hypotheses themselves or assign their employees the task of doing the combination. Managers can also serve on committees to negotiate for resources or to produce committee reports to be sent up the hierarchy. Managers at each level of the hierarchy receive rewards from higher levels, they reward their employees for what they produce, they can hire new employees or fire unproductive employees, and they can buy or sell data and services by sideways transfers to other managers.

An MDL society learns by reorganizing itself to produce improved results, which humans or other agents are willing to buy. The reward system addresses the basic problems faced by Pandemonium: increasing resources for the most productive agents; reducing resources for the less productive agents; and reorganizing the hierarchy by growing the more productive branches and shrinking the less productive branches. The rewards pass through the management hierarchy to create an effect similar to the *backward propagation* learning of a neural network. But unlike the simple switches and numeric functions of a neural network, MDL agents can be arbitrarily complex programs or reasoning systems, they can hire or fire other agents, and the messages can be propositions or even large documents stated in some version of logic. If the messages are stated in a dialect of Common Logic, they could be translated to CLCE in order to provide humanly readable explanations or an "audit trail" about the way the FMF system derived its data, hypotheses, and reports. These options are not possible with the numeric weighting schemes of most neural networks. (Note, however, that individual agents in an FMF system could use any computing mechanism internally, including neural networks. But such agents would communicate with other FMF agents by the usual FMF message formats.)

4 Interpreting Natural Languages

A system that interprets natural language must take into account all the aspects of language covered by the eight boxes in Figure 1. As that diagram suggests, every aspect is related, directly or indirectly, to every other aspect. Psycholinguistic studies indicate that people process all those aspects simultaneously, and brain scans indicate that different aspects seem to be processed in different parts of the brain. None of the psychological or neurological studies, however, are sufficiently detailed to show the internal data formats or the kinds of operations performed on that data. As a working hypothesis, many linguists and computational linguists have assumed that the underlying conceptual structures can be conveniently represented by labeled graphs, possibly with nested graphs within graphs. That assumption is very general, since it includes most of the alternatives as special cases: strings, trees, feature structures, and various notations for logic. Conceptual graphs are a semantic representation influenced by the research in linguistics, logic, psycholinguistics, and computational linguistics (Sowa 1984, 2008). They can represent ISO standard Common Logic as a proper subset, but they can also be processed by scruffy heuristics.

For the VivoMind implementations, conceptual graphs are generated in the semantics division in the center of Figure 2, and they are further elaborated in the pragmatics and model-building divisions at the right. For any input text, the morphology and syntax divisions at the left usually begin the processing, but the VP agents that manage the other divisions run concurrently. Therefore, they can begin to make partial contributions to the analysis before the morphology and syntax agents have finished the sentence. As an example, the following sentence appeared in a text about oil and gas exploration:

> The Diana field is situated in the western Gulf of Mexico
> 260 km (160 mi) south of Galveston
> in approximately 1430 m (4700 ft) of water.

If the sentence had ended with the word *Mexico*, the syntax would be unambiguous. But the measures in the next two lines, the parenthetical expressions, and the points of attachment of phrases create multiple ambiguities. Is Diana field or the Gulf of Mexico south of Galveston? What is in the water? Diana field, the Gulf of Mexico, or Galveston? After a devastating hurricane, Galveston was under water, but the ontology should indicate that cities are not expected to be under water.

Agents that process lexical information, context, heuristics, and domain knowledge contribute to the interpretation. A morphology agent expands “ft” to “feet”. An ontology for the geoscience domain indicates that Diana field is a reservoir, which consists of rocks that trap hydrocarbons; such a reservoir is underground; and the ground may be under water. Parenthesized expressions are usually idiosyncratic and ad hoc. One agent detected measures that were approximately equal, but stated in different units. Therefore, it made the hypothesis that the parenthesized expressions were intended to express equality. During the parsing process, the agents can create multiple links as tentative hypotheses. A manager in charge of those agents evaluates the evidence for each alternative and prunes away the unlikely options. The remaining links indicate that Diana field is south of Galveston and in the water.

Many different syntactic parsers have been used to generate conceptual graphs. But theories that focus on the connections between words, such as *dependency grammars* and *link grammars*, are convenient because their links map directly to the nodes and arcs of CGs. Several different parsers have been implemented at VivoMind, and the ones based on link grammar (Sleator & Temperley 1993) have been the easiest to combine with the graph operations for semantics. The most recent VivoMind parser is still based on link grammar, but it has been influenced by ParseTalk, a distributed, concurrent, parser. Hahn et al. (1994, 2000) noted that ParseTalk replaces "the static global-control paradigm" of conventional parsers with "a dynamic, local-control model" that supports "a balanced treatment of both declarative and procedural constructs within a single formal framework." The ParseTalk control structure is based on *actors* implemented in an object-oriented language (Smalltalk). Bröker (1999) added semantic actors to the original syntactic actors of the ParseTalk system. He showed that the control structure based on concurrent actors made it easy to support actors for multiple knowledge sources.

The ParseTalk actors and the FMF agents have similar advantages, but the object-oriented actors are more tightly coupled than the heterogeneous FMF agents. As the developers said, ParseTalk has "a single formal framework." For the FMF agents, the only thing that is common to all of them is the message format with six fields. Different agents can use different languages, different paradigms, and even different hardware located on different continents. The loose coupling of the FMF agents makes it easy to add new agents with new capability without disrupting any of the older functions; it also enables the system to continue if some agent or agents fail. In some applications, one or more FMF agents failed, but the system continued to run without their input. Eventually, the manager of the agents restarted them.

5 Reasoning with Multiple Paradigms

Deduction is the most common method of reasoning used with logic-based systems. But deduction is precise, predictable, and brittle. If everything is perfect, deduction is perfect. Such perfection is only achievable in mathematics. For normal, imperfect computer applications, deduction can magnify and propagate any imperfec-tion to the point of a total collapse. When people reason, they employ some safe-guards. They seldom carry out long chains of deductions. When a conclusion seems odd, a prudent individual would check the facts, ask for advice, and perform a "sanity check" by using an alternative method of reasoning. People don't expect every message to be completely understood. They ask questions, give explanations, negotiate, and compromise. In short, they use multiple paradigms to cross-check their results and avoid the biases that tend to occur with just a single paradigm.

Frege and Peirce were pioneers in logic, who independently discovered equivalent representations for full first-order logic. But they had different goals for logic. Frege applied his logic to mathematics, for which deduction is the primary method of reasoning. But Peirce used logic in a much broader range of applications, including scientific discovery, philosophical analysis, and the definition of words in linguistics and

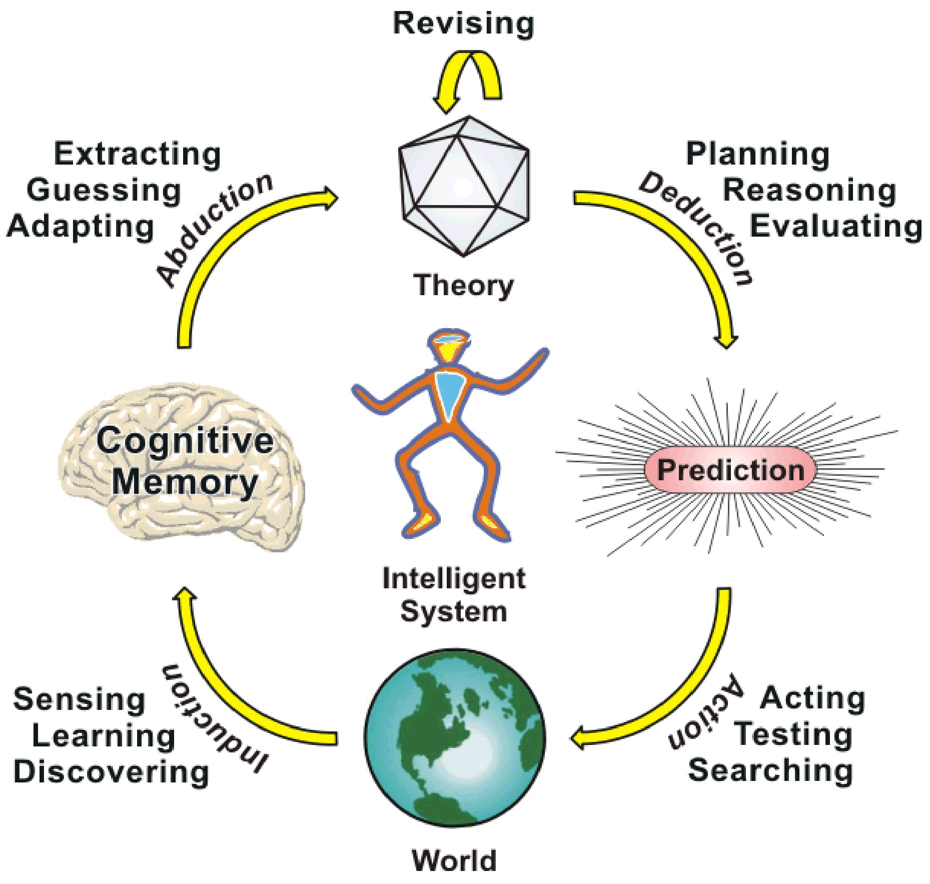


Fig. 3. Peirce's cycle of pragmatism

lexicography. In addition to deduction, Peirce emphasized the use of *induction* in generalizing from examples and *abduction* in forming hypotheses or educated guesses. Unlike many logicians who viewed metaphors and analogies with suspicion, Peirce (1902) included analogy as one of the four ways of reasoning: "Besides these three types of reasoning there is a fourth, analogy, which combines the characters of the three, yet cannot be adequately represented as composite." Figure 3 is a diagram of Peirce's cycle of reasoning in his "logic of pragmatism."

Note that deduction is only 25% of the cycle. By itself, deduction can only derive the consequences of already familiar assumptions. Induction is necessary for forming generalizations from new data, abduction is necessary for guessing or hypothesis formation, and testing is necessary to keep reasoning in touch with reality. Analogy combines aspects of the other three methods of reasoning, and it can be used by itself as the primary method for informal reasoning. The brain in Figure 3, labeled cognitive memory, represents an open-ended associative store of all the knowledge and data acquired by a system, natural or artificial. In capital letters, Cognitive Memory™ is a high-performance associative memory developed by VivoMind.

A critical component of any intelligent reasoner is a high-speed associative memory for finding relevant chunks, K-lines, schemata, or other patterns of knowledge. For conceptual graphs, that would require a high-speed method for indexing and finding relevant graphs and subgraphs. Some of the most advanced research on processing graphs has been done by chemists, who need to classify and search for millions of graphs of organic molecules. An application of chemical algorithms to conceptual graphs led to the first high-speed method for classifying and finding conceptual graphs (Levinson & Ellis 1992); one implementation of that method was used in the web site of a large online retailer (Sarraf & Ellis 2006). More recent work on chemical graphs has produced algorithms for encoding both the graph structure and the labels in numeric vectors, indexing the encodings, and finding all graphs within a small semantic distance of a given query graph (Rhodes et al. 2007); those algorithms are being used to index and search a database of over four million chemical graphs. Those techniques resemble the methods for indexing conceptual graphs and finding analogous graphs in logarithmic time (Sowa & Majumdar 2003):

- Convert each graph to a unique linear representation. For a chemical graph, the conversion is based on its International Chemical Identifier (InChI). Similar conversions can be applied to labeled graphs of any kind.
- Map the linear form to numeric vectors that encode both the graph structure and the ontology (labels) on the nodes and arcs.
- Use a measure of semantic distance between the vectors. For conceptual graphs, that measure takes into account both the structure (ordering, connectivity, and cycles) and the ontology (type labels and hierarchy). For chemical graphs, similar structural properties are used, but the ontology is based on the properties of atoms and chemical bonds.
- Use the semantic distance measure to index the graphs and find graphs within a given distance (threshold).

For conceptual graphs, the time to build the index is proportional to (N log N), where N is the number of graphs. The time to find graphs that are similar to a given query graph is proportional to (log N). If more than one graph is found within a given threshold, structure-mapping algorithms can be used (Falkenhainer et al. 1989), but it's often faster to distinguish graphs by applying additional semantic operations to the encodings.

The Flexible Modular Framework with multiple heterogeneous agents has proved to be a flexible, robust, and efficient system for learning, reasoning, and language processing. The six-field message format together with associative blackboards has the computational power of the π-calculus. The Cognitive Memory system provides a high-speed resource for analogy finding, case-based reasoning, and associative access to knowledge and information of any kind. The Market Driven Learning methods with the rewards of resources for good performance extend the π-calculus to a version of the $-calculus or cost-calculus by Eberbach et al. (2004). A cost measure based on space and time requirements can constrain the excesses of systems like Pandemonium and focus agents toward promising directions. Following Turing (1939), who showed that a Turing machine that could access arbitrary information from the environment (or *oracle*) was strictly more powerful than a Turing machine in isolation, Eberbach et al. claimed that the ability to access information from outside sources served the same purpose as an oracle. Whatever the theoretical power, the FMF with these additions has served as a flexible tool for rapidly building intelligent systems.

References

1. Baum, E.B.: What is Thought? MIT Press, Cambridge (2004)
2. Bröker, N.: Eine Dependenzgrammatik zur Kopplung heterogener Wissensquellen. Max Niemeyer Verlag, Tübingen (1999)
3. Brooks, R.A.: Intelligence without representation. Artificial Intelligence 47, 139–159 (1991)
4. Eberbach, E., Goldin, D., Wegner, P.: Turing's ideas and models of computation. In: Teuscher, C. (ed.) Alan Turing: Life and Legacy of a Great Thinker. Springer, Berlin (2004)
5. Falkenhainer, B., Forbus, K.D., Gentner, D.: The structure mapping engine: algorithm and examples. Artificial Intelligence 41, 1–63 (1989)
6. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems, 80–112 (1985)
7. Hahn, U., Schacht, S., Bröker, N.: Concurrent natural language parsing: The ParseTalk model. International Journal of Human-Computer Studies 41, 179–222 (1994)
8. Hahn, U., Bröker, N., Neuhaus, P.: Let's ParseTalk: Message-passing protocols for object-oriented parsing. In: Bunt, H., Nijholt, A. (eds.) Recent Advances in Parsing Technology. Kluwer, Dordrecht (2000)
9. ISO/IEC 2007 Common Logic (CL) — A Framework for a family of Logic-Based Languages, IS 24707, International Organisation for Standardisation (2007)
10. Lenat, D.B.: Cyc: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33–38 (1995)
11. Levinson, R.A., Ellis, G.: Multilevel hierarchical retrieval. Knowledge Based Systems 5(3), 233–244 (1992)
12. Majumdar, A.K., Sowa, J.F., Stewart, J.: Pursuing the goal of language understanding. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 21–42. Springer, Heidelberg (2008)
13. Majumdar, A., Keeler, M., Tarau, P., Sowa, J.: Semantic distances as knowledge capture constraints. In: First Workshop on Knowledge Capture and Constraint Programming (KCCP 2007), Whistler, BC (2007)
14. McCarthy, J.: Elephant 2000: A programming language based on speech acts (1989), `http://www-formal.stanford.edu/jmc/elephant.html`
15. Milner, R.: Communicating and Mobile Systems: The *Π* calculus. Cambridge University Press, Cambridge (1999)
16. Minsky, M.: K-lines: A Theory of Memory. Cognitive Science 4, 117–133 (1980)
17. Minsky, M.: The Society of Mind. Simon and Schuster, New York (1986)
18. Minsky, M.: Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. AI Magazine 12(2), 34–51 (1991)
19. Minsky, M.: Review of Allen Newell's Unified Theories of Cognition. Artificial Intelligence 59, 343–354 (1993)

20. Minsky, M.: The Emotion Machine: Commonsense Thinking. In: Artificial Intelligence, and the Future of the Human Mind. Simon & Schuster, New York (2006)
21. Monderer, D., Tennenholtz, M., Varian, H.: Economics and artificial intelligence. Games and Economic Behavior 35(1-2), 1–5 (2001)
22. Newell, A.: Unified Theories of Cognition. Harvard University Press, Cambridge (1990)
23. Peirce, C.S.: Logic, Considered as Semeiotic, MS L75, edited by Ransdell, J. (1902), http://members.door.net/arisbe/menu/library/bycsp/l75/l75.htm
24. Rhodes, J., Boyer, S., Kreulen, J., Chen, Y., Ordonez, P.: Mining patents using molecular similarity search. In: Pacific Symposium on Biocomputing, vol. 12, pp. 304–315 (2007)
25. Sarraf, Q., Ellis, G.: Business rules in retail: The Tesco.com story. Business Rules Journal 7(6) (2006), http://www.brcommunity.com/a2006/n014.html
26. Selfridge, O.G.: Pandemonium: A paradigm for learning. In: The Mechanization of Thought Processes, NPL Symposium No. 10, Her Majesty's Stationery Office, London, pp. 511–526 (1959)
27. Singh, P.: Examining the society of mind. Computing and Artificial Intelligence 22(6) (2003)
28. Sleator, D., Temperley, D.: Parsing English with a link grammar. In: Third International Workshop on Parsing Technologies (1993), http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/LG-IWPT93.pdf
29. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
30. Sowa, J.F.: Architectures for intelligent systems. IBM Systems Journal 41(3), 331–349 (2002), http://researchweb.watson.ibm.com/journal/sj41-3.html
31. Sowa, J.F.: Graphics and languages for the Flexible Modular Framework. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 31–51. Springer, Heidelberg (2004)
32. Sowa, J.F.: Conceptual graphs. In: van Harmelen, et al. (eds.), pp. 213–237 (2008)
33. Sowa, J.F., Majumdar, A.K.: Analogical reasoning. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 16–36. Springer, Heidelberg (2003)
34. Turing, A.M.: Systems of logic based on ordinals. Proc. London Mathematical Society, Series 2 45, 161–228 (1939)
35. van der Hoek, W., Wooldridge, M.: Multiagent systems. In: van Harmelen, et al. (eds.), pp. 887–928 (2008)
36. van Harmelen, F., Lifschitz, V., Porter, B. (eds.): Handbook of Knowledge Representation. Elsevier, Amsterdam (2008)

Semantic Search – Using Graph-Structured Semantic Models for Supporting the Search Process

Thanh Tran, Peter Haase, and Rudi Studer

Institute AIFB, Universität Karlsruhe (TH), Germany
{dtr,pha,rst}@aifb.uni-karlsruhe.de

Abstract. Semantic search attempts to go beyond the current state of the art in information access by addressing information needs on the semantic level, i.e. considering the meaning of users' queries and the available resources. In recent years, there have been significant advances in developing and applying semantic technologies to the problem of semantic search. To collate these various approaches and to try to better understand what the concept of semantic search entails, we describe semantic search from a process perspective. We argue that semantics can be exploited in all steps of this process. We describe the elements involved in the process using graph-structured, semantic models and present our existing work on semantic search in terms of this process.

1 Introduction

The availability of structured information on the Semantic Web enables new opportunities for information access. Search is no longer limited to matching keywords against documents, but instead complex information needs can be expressed in a structured way, with precise and structured answers as results [1–3]. We refer to this kind of information access, in which information needs are addressed by considering the *meaning* of the user queries and available resources, as *semantic search*.

In recent years, there have been significant advances in developing and applying semantic technologies to the problem of semantic search. To collate these various approaches and to try to better understand what the concept of semantic search entails, we describe semantic search from a process perspective, i.e. as an information access process. We argue that semantics can be exploited in all steps of this process, beginning with the interpretation of the user information needs, continuing with the actual processing of the queries, over the presentation of results, to the exploitation of user feedback. We describe the elements involved in the process using graph-structured, semantic models of the resource -, schema -, query -, and answer space.

We further describe our existing work on semantic search in terms of this process. In this sense, this paper can be seen as survey of our current work that compiles the individual pieces to construct a whole picture of the entire process of semantic search.

The paper is organized as follows: In Section 2 we introduce the semantic search process, followed by the formalization of the elements involved in this process in Section 3. In the subsequent sections, we describe our work covering the individual steps of the search process, with the translation of user queries in Section 4, query processing in Section 5, and result presentation and query refinement in Section 6. After a discussion of related work in Section 7, we conclude in Section 8.

2 Semantic Search – A Process View on Information Access

In this section, we describe our approach to semantic search. The term semantic search has been used in various contexts. There exist many different conceptions and definitions for semantic search [1, 4, 5]. In this paper, search is regarded as a process. The starting point for this process is some information need. This information need might arise from a concrete task the user aims to accomplish. In this case, the need is very concrete such that the user exactly knows what to look for. In other cases, the information need might be vague initially, but might become more concrete during the process. The main objective of our approach is to cater for these different scenarios and to assist the user throughout the *search process*.

In Fig. 1, we illustrate the different steps involved in this process. Hereby, offline tasks can be distinguished from processing steps that have to be carried out online. Resources that can help in addressing the user information need are either information about real world resources, or documents of various types and formats. At first, these resources need to be represented and indexed in order to make them available for search. These steps are referred to as knowledge representation (in the case of resources) or document representation (in the case of documents) and *indexing*.

Given the information need, the first task to be performed by the user is to formulate this need as a query (*query construction*). The query is then processed against the indices to obtain the relevant resources (*query processing*). The results are then presented to the user (*result presentation*). They might exactly match the user need such that the process would end here. In many cases, especially when the initial query is only a vague or incomplete representation of the user need, further steps are required. The user may browse the intermediate results and navigate to more relevant resources. Alternatively, the user might want to reformulate or refine the initial query posed against the system (*query refinement*). These steps might be performed iteratively until the information need is completely satisfied.

In practice, a system is called a *semantic search* system if semantic technologies are involved in some stages of this process. A distinctive characteristic of such a system shall be the explicit use of semantics. In this regard, semantics is concerned with the meaning of the resources made available for search. Meaning is established through a semantic model, which essentially captures interrelationships between syntactic elements and their interpretations. Various semantic models have been proposed and used in different research communities. There are linguistic models such as thesauri that capture relations between words. In the database community, conceptual models and Entity Relationship diagrams are used to capture relations between data elements [6]. In the semantic web community, ontologies have received widespread acceptance. The notion of ontologies employed by this community is very general. Ontologies constitute rather a family of models, which might differ in the degree of expressivity and formality, ranging from simple taxonomies and lightweight ontologies (e.g. represented in RDF(S)) to formal theories (e.g. represented in Description Logics) where interpretations of symbols and relationships are precise and computable [7].

In this paper, the meaning of the underlying resources is captured through *semantic models*, which essentially, represent classes of entities and relations between them. With respect to the categories mentioned above, these models correspond to lightweight ontologies represented in RDF(S). We elaborate on these models in detail in the next and subsequent sections. Now, we briefly summarize the different steps we have implemented to support the general semantic process introduced previously. In particular, we discuss how semantics represented through our models is exploited throughout this process:

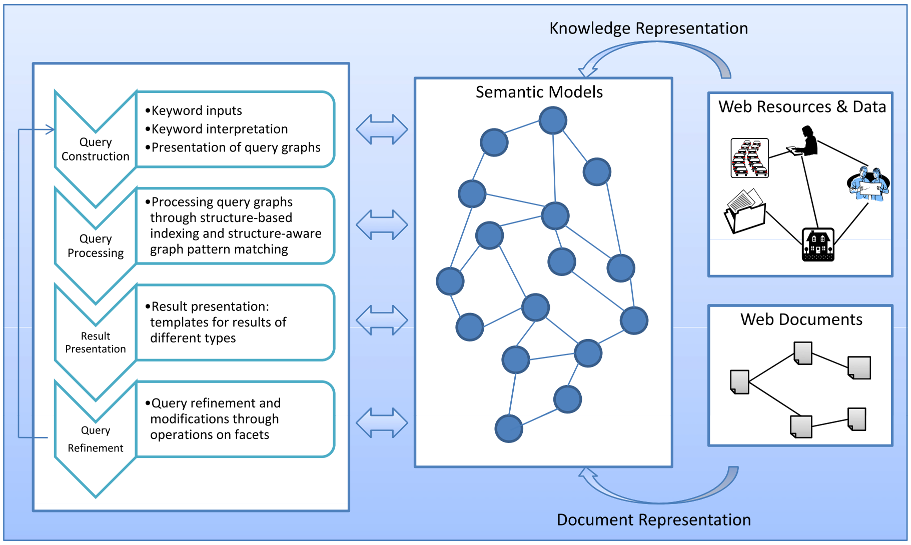

Knowledge Representation
Query Construction
•Keyword inputs
•Keyword interpretation
•Presentation of query graphs
Query Processing
•Processing query graphs through structure-based indexing and structure-aware graph pattern matching
Result Presentation
•Result presentation: templates for results of different types
Query Refinement
•Query refinement and modifications through operations on facets
Semantic Models
Web Resources & Data
Web Documents
Document Representation

Fig. 1. Using semantic models for the search process

– *Specification of keyword inputs.* In our approach, users articulate their information needs using keyword queries. We believe this is the most adequate form, as keyword interfaces have been widely adopted, and users are familiar with them both due to their simplicity and their presence in today's systems. For searching, the users do not need to know about the query syntax, the schema and even the labels of the data elements. They can simply use their own words to express their information needs.
– *Keyword interpretation for computing query graphs.* The meaning of the keywords is computed subsequently. This follows a procedure called keyword translation where different interpretations of the keywords are derived. More precisely, the keywords are transformed into more expressive structured conjunctive queries, which contain elements matching the meaning of the keywords, as well as additional elements that add meaning to the query. For this, we make use of a semantic model to represent the structure and semantics of the *query space*. We apply a graph exploration algorithm to identify different interpretations of the keywords within this query space, i.e. compute query graphs.
– *Presentation of query graphs to user.* In traditional search, users issue a query, obtain (a set of) results, and – if the results do not fulfill the information need – start over with issuing a new query. We introduce an additional step, in which different possible interpretations of the user information need are computed and presented to

the user. Instead of presenting the results directly, which might actually belong to many distinct queries (representing different information needs), we allow the user to select the correct interpretation of the inputs.

We have designed a query ranking scheme that is based on the structure- and semantics-related features of the query space. According to this scheme, the computed interpretations, i.e. structured queries, are sorted and presented to the user. The queries are not presented using a formal syntax, but using an intuitive, graph-based representation. Additionally, we also present snippets of the query results to help the user in understanding the meaning of the query. In certain cases, these snippets may already be the answers to the information need such that the following steps may not be needed.

– *Processing query graphs.* The query graph selected by the user has to be matched against the representation of the system resources to obtain final answers. Similar to the concept of a query space, we employ an answer space, a semantic model that more compactly encodes the search space that has to be explored for computing answers. Instead of matching the query against the system resource, we first process the query against the more concise answer space. This results in a set of candidates that are known to satisfy the structural constraints of the query. In the second step, these candidates are further refined to verify that they also match the concrete entities mentioned in the queries (i.e. constants and distinguished variables). The main advantages of using the answer space are reduction in I/O costs and reduction in the number of joins and unions, i.e. reduced space and time complexity.

– *Presentation of results.* The answers to the selected query are presented to the user. The conjunctive queries we focus on can be classified into three main types: entity queries, factual queries and general conjunctive queries. The query type ultimately determines the structure of query result, and thus, the way it should be presented to the user. We have designed different templates for query results of these different types. In the case of general conjunctive queries for instance, results are sets of tuples that satisfy the conjunctive query. They are presented to the user in a structured, tabular form.

– *Facet-based query refinements.* Refinements to the query may be needed for several reasons. The computed interpretations may not exactly match the information need. Also, the user may start out with a vague information need, not knowing exactly what he is searching for. For these cases, we make use of a semantic model called *schema space* that describes the different types, relations and attributes exhibited by the underlying resources. This semantic information acts as facets describing the resources currently presented to the user. Based on these facets, we provide means for the user to narrow down or expand the resources of interest according to their information need in an interactive way. In particular, the user can add, remove or edit the facets. These operations are transparently converted to changes on the conjunctive query. The query refined this way is immediately evaluated, and new results are presented without the user having to explicitly issue a new query.

3 Data and Semantic Model

We will begin with the most fundamental model of our approach, i.e. the *resource space*, a graph-based model of the underlying resources we make available for search:

Definition 1. *A resource space S_R is a set of resource graphs $g_R(V, L, E)$ where*

- *V is a finite set of vertices. Thereby, V is conceived as the disjoint union $V_E \uplus V_V$ with E-vertices V_E (representing entities) and V-vertices V_V (data values),*
- *L is a finite set of edge labels, subdivided by $L = L_R \uplus L_A$, where L_R are relation labels and L_A are attribute labels.*
- *E is a finite set of edges of the form $e(v_1, v_2)$ with $v_1, v_2 \in V$ and $e \in L$. Moreover, the following types are distinguished:*
  - *$e \in L_A$ (A-edge) if and only if $v_1 \in V_E$ and $v_2 \in V_V$,*
  - *$e \in L_R$ (R-edge) if and only if $v_1, v_2 \in V_E$,*
  - *and $type$, a predefined edge label that denotes the class membership of an entity.*

Example 1. An example resource graph describing relationships between persons, universities and articles is depicted in Fig. 2.

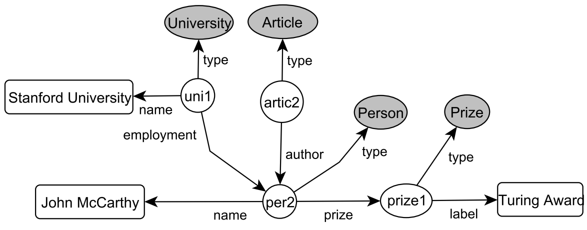


Fig. 2. An example resource graph

As discussed in the last section, searchable resources might be documents and real world entities. Note that in our model, they are commonly represented as entities, i.e. E-vertices of a graph-structured resource space. Documents represent a class of entities, which might have relation to other entities such as author and publisher and special attributes such as title and abstract. They are indexed and retrieved in the same way like other types of resources. In other words, the retrieval of documents and data amounts to the same, namely *entity retrieval.*

In the resource space, we do not distinguish between different types of E-vertices in v_E, i.e. classes and instances. Intuitively, a class denotes a group of instances. Instances in the same class might exhibit similar types of relations and attributes. This semantics about classes and their relations can be explicitly defined in the schema space.

Definition 2. A schema space S_S *is a set of schema graphs $g_S(V, L, E)$ where*

- *V is a finite set of vertices. Here, V is conceived as the disjoint union $V_C \uplus V_R \uplus V_A \uplus V_D$ with C-vertices V_C (classes), R-vertices V_R (relations), A-vertices V_A (attributes), and D-vertices V_D (data types).*

– *L comprises the pre-defined edge labels $subclassof$, $domain$, $range$.*
– *E is a finite set of edges of the form $e(v_1, v_2)$ with $v_1, v_2 \in V$ and $e \in L$, where*
  - *$e = domain$ if and only if $v_1 \in V_A \cup V_R$ and $v_2 \in V_C$,*
  - *$e = range$ if and only if $v_1 \in V_A, v_2 \in V_D$ or $v_1 \in V_R, v_2 \in V_C$, and*
  - *$e = subclassof$ if and only if $v_1, v_2 \in V_C$.*

Example 2. Fig. 3a) illustrates an example schema. While this one comprises of relations between classes only, a typical schema also contains attributes and data types.

Note that with respect to the different types of semantic models discussed in the previous section, the schema graph corresponds to a lightweight ontology. Alternatively, it might also be given as a formal model that is backed by a logical theory (e.g. an OWL ontology). Such a model would allow for reasoning, e.g. to infer additional knowledge that is implicitly captured in the representation of the underling resources. While this inference capability clearly can help in satisfying the information need of the user, it comes at the cost of higher computational complexity. In this paper, we focus on the use of lightweight semantics, i.e. compact descriptions of structures exhibited by the underlying resources. We show how these lightweight semantic models can be used for the interpretation of the user keywords, for guiding the process of query answering, and for helping the user in refining the query and answers.

Note that a schema essentially describes structural relationships exhibited by the underlying data. Such a description might be given explicitly, e.g. in the form of Entity Relationship diagrams or RDF(S) ontologies[1]. However, in many cases, a structural description may be incomplete or may not exist for a given resource space. Also, different tasks require descriptions at different "levels of granularity". For query refinement and query interpretation, it suffices to know which relationships might exist between classes of entities, i.e. we can use the schema space for these tasks. However, for query processing, we need a more precise description which guarantees that there exist some particular relationships. In the following sections, we will elaborate on the different semantic models we use for the semantic search process and also, we will discuss how they can be derived automatically from the structural properties found in the data.

4 Query Space – Enabling Query Construction Using Keywords

In this section, we describe the computation of possible interpretations of the user keywords. These interpretations are presented to the user in the form of query graphs. The computation of query graphs from keywords basically involves three tasks: 1) construction of the query space, 2) top-k query graph exploration, and 3) query graph ranking. The algorithms employed for these tasks have been described in [8]. Here, we focus our discussion on the underlying semantic model, i.e. the *query space*.

Typically, the search space employed for keyword search is the resource graph [9] [10]. It is used for the exploration of substructures that connect keyword elements. Such

[1] Note that the schema graph is close to a RDF(S) ontology. The intuitive mapping from RDF(S) to the schema graph is: resources correspond to entities, classes to classes, properties to either relations or attributes and literals to data values.

an exploration might be very expensive when the resource graph is large. While these approaches focus on computing answers, we are interested in interpreting queries, i.e. we want to derive the query structure from the edges and the constants and variables from the vertices of the resource graph. For this, we employ a more compact representation of the resource graph to keep the search space minimal. We introduce the query space model which aims to capture only information that is necessary for the computation of possible interpretations. Essentially, it consists of two parts: 1) the graph elements that match the user keywords (to identify the query constants) and 2) possible relations between classes of entities (to derive the query predicates):

Definition 3. *A* query space $S_Q(S_S, N_K)$ *comprises of keyword matching elements* N_K *computed for a query* q *which when not already contained, are connected with elements of a special schema space* S_S *consisting of the graphs* $g_S(V, L, E)$ *where*

– V *is conceived as the disjoint union* $V_C \uplus V_R$*,*
– L *comprises of the pre-defined edge labels* $subclassof$*,* $domain$*,* $range$*,*
– E *is a finite set of edges of the form* $e(v_1, v_2)$ *with* $v_1, v_2 \in V$ *and* $e \in L$*.*

To compute N_K, keywords of q are matched against the labels of elements of the resource space. The second part is simply the schema without attributes and data types. For query interpretation, we navigate through different paths of the query space to connect elements in N_K. Note that paths on the schema space end at a $datatype$ vertex. Thus, edges of the form $attribute - data\ type$ represent dead ends. Navigating along such edges do not help in finding further connections. Therefore, they are not considered during the construction of the query space.

In cases where there is no schema information available, the special schema we need can be derived from the resource space by deleting all V-vertices and A-edges, and by the subsequent and exhaustive application of the following clustering rules:

1. Every E-vertex v_e is clustered to a C-vertex v_c if there is an edge $type(v_e, v_c)$. If there is no such C-vertex, v_e is clustered to $Thing$, a special C-vertex denoting the most general class of entities. Here, clustering means that v_e is deleted from the graph and every v_c inherits all the edges from v_e except $type$.
2. Note that the application of the previous rule results in edges of the form $e(v_{c_i}, v_{c_j})$. Two edges $e_i(v_{c1}, v_{c2})$ and $e_j(v_{c3}, v_{c4})$ are then clustered to one if $e_i = e_j$, $v_{c1} = v_{c3}$, and $v_{c2} = v_{c4}$. Here, clustering simply means either to delete e_i or e_j.

Example 3. Fig. 3a) illustrates the schema that is derived from our example resource graph in Example 1. Fig. 3b) shows the query space constructed for the example query q_{ex} “Article Stanford Turing Award”. The keyword elements obtained through matching the user keywords against resource labels are *Article*, *Stanford University* and *Turing Award*. *Article* is already contained by the schema. *Stanford University* and *Turing Award* are connected to the respective vertices of the schema to obtain the search space for q_{ex}.

Given the query space, query interpretation amounts to searching for the minimal query graphs, defined as follows:

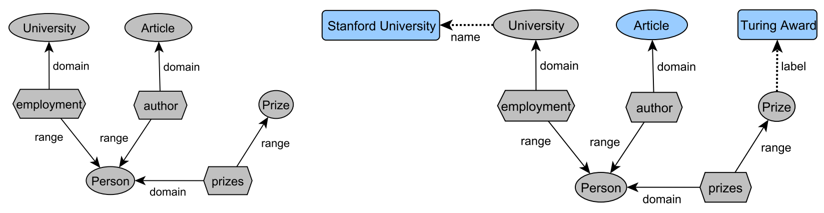


Fig. 3. a) Special schema graph computed from the resource graph in Example 1 b) a query space obtained through connecting keyword-matching elements with elements of the schema

Definition 4. *Let $S_Q = (S_S, N_K)$ be the query space, $K = \{k_1, \ldots, k_n\}$ be a set of keywords, and let $f : K \rightarrow 2^{V_K \cup E_K}$ be a function that maps keywords to sets of corresponding graph elements (where $V_K, E_K \subseteq N_K$). A* query graph *is a matching subgraph $g_q = (V_q, L_q, E_q)$ with V_q, L_q and E_q being elements of S_S and*

– *for every $k \in K$, $f(k) \cap (V_q \cup E_q) \neq \emptyset$, i.e. g_q contains at least one representative keyword matching element for every keyword from K, and*
– *g_q is connected, i.e. there exists a path from every graph element to every other graph element.*

A matching graph g_{q_i} is minimal if there exists no other g_{q_j} of g such that $Score(g_{q_j}) < Score(g_{q_i})$, where $score : g_q \rightarrow [0, 1]$.

We employ a top-k procedure to find such query graphs. It starts from the keyword elements N_K and iteratively explores the query space S_Q for all distinct paths beginning from these elements. During this procedure, the path with the highest score so far is selected for further exploration. For scoring paths we incorporate 1) the popularity of graph elements (e.g. computed via PageRank), 2) the matching score of keyword elements (obtained via the imprecise matching of keywords to element labels), and 3) the length of the path. At some point, an element might be discovered to be a connecting element, i.e. there is a path from that element to at least one keyword element, for every keyword in K. The paths between the keyword elements and the connecting element are merged to form a query graph. The graphs explored this way are added to the candidate list. The process continues until the upper bound score for the query graphs yet to be explored is lower than the score of the k-ranked query graph in the candidate list, i.e. no candidates can beat the k-ranked result. More algorithmic details can be found in [8].

Example 4. Fig. 4a shows an example query space containing elements associated with some scores. Based on these scores, the path score is updated at every step, which is then used to prioritize the "direction" of the exploration. The exploration starts from the keyword elements "Stanford University", "Article" and "Turing Award", as shown in Fig. 4a (labels of "non-keyword elements" are omitted due to lack of space). The three different paths starting from these elements that have been iteratively explored during the top-k procedure are also shown. For the first time, these three paths meet at the vertex with the EF-IDF score = 0.0002, i.e. this vertex is a connecting element. These paths are merged to form the query graph shown in Fig. 4b.

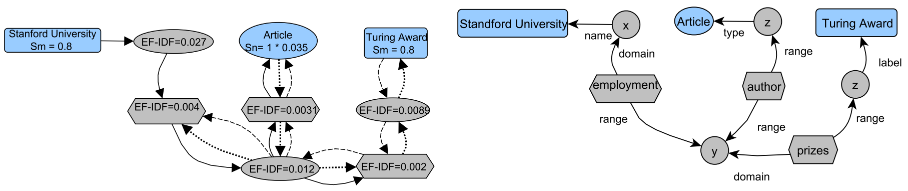


Fig. 4. a) Three paths through the query space and their scores b) A resulting query graph

Note that the semantic model employed here is essentially about entities denoted by keywords (keyword matching elements) and their possible relations. It models the space of possible interpretations of the keywords. Using this model, query computation operates on a more concise representation of the query search space. In experiments presented in [8], substantial performance increase is possible compared to state-of-the-art on keyword search.

5 Answer Space – Enabling Efficient Query Processing

In this section, we discuss another semantic model called the answer space. It represents a more compact representation of the answer search space, which is employed for more efficient query processing. In particular, we are concerned with the matching of query graphs (i.e. the ones resulting from keyword interpretation as shown in Fig. 4b. These graphs represent conjunctive queries, an important fragment of widely used query languages (such as SQL, SPARQL[2]) Here, we focus our discussion on the model and refer the interested reader to [11] for algorithmic details and proofs.

An answer space is essentially a collection of answer graphs, where vertices denote extensions, i.e. a set of elements. In particular, every such extension contains elements of the resource space, which exhibit the same structure, i.e. have same (incoming and outgoing) paths. Intuitively speaking, an answer space is a compact representation of the different structures exhibited by elements of the Resource Graph.

Example 5. Fig. 5 shows an extended example for the resource space. Its associated answer space is shown in Fig. 6a. The extension E_4 in Fig. 6a for instance, comprises of $uni1$ and $uni2$, which as illustrated in the resource space in Fig. 5, are similar in structure.

We formalize the concept of an answer space by the notion of a bisimulation originating from the theoretical analysis of state-based dynamic systems. Essentially, graph nodes are considered bisimilar in this sense if they cannot be distinguished by means of "edge trees" starting from them. Moreover, we parameterize our notion of bisimularity by two sets (forward and backward) of edge labels. We now define our notion of parameterized bisimilarity.

Definition 5. *Given a resource graph $g_R = (V, L, E)$ and two edge label sets $L_1, L_2 \subseteq L$, a (L_1-*forward*-L_2-*backward) bisimulation *on G is a binary relation $R \subseteq V \times V$ on the vertices of G such that for $v, w \in V$, $l_1 \in L_1$ and $l_2 \in L_2$:*

[2] SPARQL is a query language for RDF data recommended by the W3C.

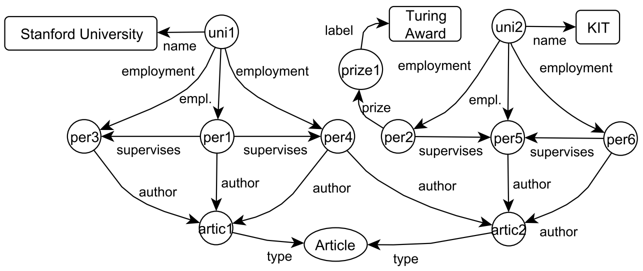


Fig. 5. a) An extended example of a resource space

- *vRw and $l_1(v, v') \in E$ implies there is a $w' \in V$ with $l_1(w, w') \in E$ and $v'Rw'$,*
- *vRw and $l_1(w, w') \in E$ implies there is a $v' \in V$ with $l_1(v, v') \in E$ and $v'Rw'$,*
- *vRw and $l_2(v', v) \in E$ implies there is a $w' \in V$ with $l_2(w', w) \in E$ and $v'Rw'$,*
- *vRw and $l_2(w', w) \in E$ implies there is a $v' \in V$ with $l_2(v', v) \in E$ and $v'Rw'$.*

Two vertices v, w are called bisimilar (written $v \sim w$), if there exists a bisimulation R with vRw.

Our notion of L_1-forward-L_2-backward bisimulation captures as special cases forward bisimulation ($L_1 = L$, $L_2 = \emptyset$), backward bisimulation ($L_1 = \emptyset$, $L_2 = L$) as well as back-and-forth bisimulation ($L_1 = L_2 = L$). Note that $\sim$ is an equivalence relation, and is itself a L_1-forward-L_2-backward bisimulation. In fact, it is the greatest (i.e. most general) one as it subsumes all possible bisimulations R. In the following, we will represent this bisimilarity equivalence by the set of its equivalence classes called *extensions*: $\{[v]_\sim \mid v \in V\}$ with $[v]_\sim := \{w \in V \mid v \sim w\}$. Recall that these equivalence classes form a partition of V, i.e. a family $\mathcal{P}_\sim$ of pairwise disjoint sets whose union is V. We use these classes to define the *answer graph* of g_R.

Definition 6. *For a given resource graph $g_R = (V, L, E)$ with greatest bisimulation $\sim$, the associated* answer graph *$g_R^\sim = (V^\sim, L, E^\sim)$ is defined as follows:*

- *The vertices of the answer graph $g_R^\sim$ are exactly g_R's $\sim$-equivalence classes : $V^\sim = \{[v]_\sim \mid v \in V\}$,*
- *The labels of $g_R^\sim$ are exactly the labels of g_R, and*
- *An edge with a certain label e is established between two equivalence classes $[v]_\sim$ and $[w]_\sim$ exactly if there are two vertices $v^* \in [v]_\sim$ and $w^* \in [w]_\sim$ s.t. there is an edge $e(v^*, w^*)$ in the resource graph:[3] $E^\sim := \{e([v^*]_\sim, [w^*]_\sim) \mid e(v^*, w^*) \in E\}$.*

We will now characterize the properties of the answer graph which justify its usage for query processing.

Proposition 1. *Let g_R be a resource graph with associated answer graph $g_R^\sim$ and let g'_R be another resource graph such that there is a homomorphism h from g'_R into g_R. Then $h^\sim$ with $h^\sim(v) := [h(v)]_\sim$ is a homomorphism from g'_R into $g_R^\sim$.*

[3] Note that from $\sim$ being an equivalence relation follows $[v]_\sim = [v^*]_\sim$ and $[w]_\sim = [w^*]_\sim$.

Roughly speaking, this proposition ensures that, whenever there is a match of a query graph on a resource graph, the query also matches on the answer graph. Moreover, the equivalence classes part of the answer graph match will contain the vertices of the resource graph match. Thus, we can use the more compact answer graph for query processing:

– In the first step, the query is matched against the answer graph, resulting in a set of answer graph matches. They contain data elements that satisfy the structural constraints captured by the query.
– In the second step, we need to verify that these data elements also match the concrete entities mentioned in the query, i.e. constants and distinguished variables, and relations among them. For this, we retrieve data elements contained in the answer graph matches, and join them along the query edges.

Example 6. Fig. 6b depicts a query, which asks for authors y working at $Stanford\ University$ that have won a $Turing\ Award$. Further, y should supervise some u that is author of some v. The matching of the query graph in Fig. 6b on the answer graph in Fig. 6a results in one single match $h = \{x \mapsto E1, y \mapsto E4, z \mapsto E7, u \mapsto E3, v \mapsto E5, Stanford\ University \mapsto E6, Turing\ Award \mapsto E8\}$. Through this structural matching, we know that elements in $E4$ work at some places x, have won some prizes z and supervise u. Further, we also know that u is author of some v. Next, we have to check whether elements in $E4$ match the concrete entities mentioned in the query, i.e. really work at $Stanford\ University$, and have won a $Turing\ Award$. For this, we retrieve data contained in the extensions E6, E1, E4, E7 and E8 and join them along the edges $\langle y\ employment\ x\rangle$, $\langle x\ name\ Stanford\ University\rangle$, $\langle y\ prize\ z\rangle$, $\langle z\ label\ Turing\ Award\rangle$.

Note that through structural matching, we retrieve and join data only for a certain part of the query, i.e. the rest of the query can be pruned away after processing step one. We will give another proposition that more precisely defines the part that can be pruned away. We will call a graph g with a distinguished node r called root *(L_1-forward-L_2-backward) tree-shaped* if its edges interpreted as undirected edges form an undirected

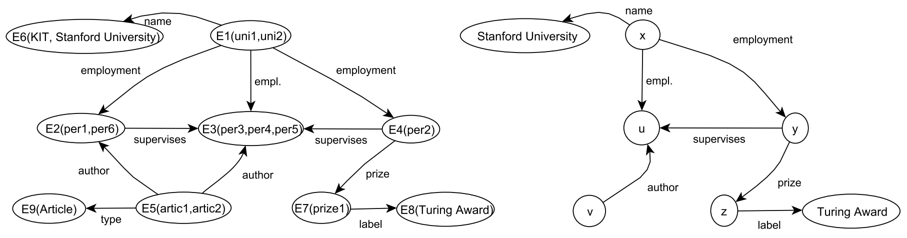


Fig. 6. a) The answer space for the resource space shown in Fig. 5 and b) an extended example of a query graph

tree and from the root node r, every path from this root to the leaves traverses forward only L_1-labeled edges and backward only L_2-labeled edges.[4]

Proposition 2. *Let g_R be a resource graph with associated answer graph $g_R^{\sim}$ (where $\sim$ is a L_1-forward-L_2-backward bisimulation) and let g'_R be a L_1-forward-L_2-backward tree-shaped resource graph with root r. Let h' be a homomorphism from g'_R to $g_R^{\sim}$. Then for every node $v \in h'(r)$, there is a homomorphism h from g'_R to g_R with $h(r) = v$.*

Informally, this proposition ensures the following: Suppose there is an accordingly tree-shaped query graph g'_R where all nodes except possibly the root r are non-distinguished variables, corresponding to a query posed against the data graph g_R. The proposition now states that for any match h' of g'_R against the answer graph $g_R^{\sim}$, every data element v of the extension(s) assigned to r – namely $h'(r)$ – represents a binding to r. In other words, for this special type of tree-like query parts, no verification step will be necessary. Data elements need to be accessed only for the root node of the query g'_R, while the rest of g'_R can be pruned away.

Example 7. Continuing with our previous example, we can see that there are two tree-like parts that contain no distinguished variables, i.e. the paths $\langle x\ employment\ u\rangle$ $\langle v\ author\ u\rangle$ and $\langle y\ supervises\ u\rangle$ $\langle v\ author\ u\rangle$. These parts can be pruned away after step one as all data elements contained in answer graph matches are already known to satisfy these structural constraints, i.e. elements in $E4$ are already known to supervise some u that are authors of v, and elements in $E1$ are known to employ some u that are authors of v respectively.

Compared to state-of-the-art approaches, the main advantages of using the answer space for query processing are the following:

– *Further Reduction of I/O Costs:* Typically, a single index lookup is sufficient for processing a query atom that involves at least one constant. Given a query atom which contains variables only, the entire table needs to be fetched from disk. In our approach, we retrieve only data elements that are known to satisfy the structural constraints of the query. Thus, the amount of data that have to be retrieved from disk might be smaller in both cases. For instance, the entire table created for the property $employment$ has to be fetched to obtain data matching the pattern $\langle x\ employment\ u\rangle$. We retrieve only those bindings to x that match other constraints of the query, i.e. have some names and employ some y. Similar arguments apply when the query atom involves constants, e.g. $\langle x\ name\ Stanford\ University\rangle$: there might be many elements in the resource space with the name $Stanford\ University$ while the number of those exhibiting the structure specified in the query should be far less.

[4] Equivalently but more formally, this can be inductively defined: every edgeless single-vertex rooted graph (g, r) with $g = (\{r\}, L, \emptyset)$ is L_1-forward-L_2-backward tree-shaped. Moreover let (g_1, r_1) and (g_2, r_2) with $g_1 = (V_1, L, E_1)$ and $g_2 = (V_2, L, E_2)$ be two L_1-forward-L_2-backward tree-shaped graphs with disjoint vertex sets. Let $v \in V_1$ and let $e \in \{l(v, r_2) \mid l \in L_1\} \cup \{l(r_2, v) \mid l \in L_2\}$. Then the rooted graph (g_3, r_1) with $g_3 = (V_1 \cup V_2, L, E_1 \cup E_2 \cup \{e\})$ is L_1-forward-L_2-backward tree shaped.

– *Further Reduction of Unions and Joins:* While state-of-the-art approaches makes join processing more efficient, it does not directly solve the proliferation of joins and unions. Like in [12] and [13], we also sort values to enable fast merge joins. Additionally, we can largely reduce the number of joins. As noted, joins are only needed to verify the relations between concrete entities mentioned in the query (denoted by constants and distinguished variables). Through pruning the two paths in our example, the number of joins rendered unnecessary makes up almost 50 percent of the total number of joins that would be needed. In the extreme cases where no answer graph matches can be found, we can skip the entire second step to avoid data access and joins completely.

Note that the semantic model employed here is about extensions of entities that exhibit similar structures. These extensions are not much different from the classes, i.e. C-vertices, that can be found in the schema space or query space. However, whereas entities of the same class might have similar properties and attributes (i.e. similar w.r.t the outgoing edges), entities of the same extension are guaranteed to have similar structures (i.e. similar w.r.t incoming and outgoing paths). An answer space thus can be regarded as a more fine-granular version of the schema space. It more compactly encodes the space of possible answers. Using this model, query processing operates on a more concise representation of the answer search space. In experiments with large-scale datasets using queries of different shapes and complexities, we have shown that this approach is 5-6 times faster than the state-of-the-art [11].

6 Presentation and Refinement

In this section, we elaborate on the use of our semantic models for query presentation, answer presentation and query refinement. These concepts have been implemented in the context of data web search [14] and semantic wiki search [15]. Based on our semantic wiki search implementation, we will now discuss the main ideas and refer the interested readers to [14] and [15] for more algorithmic and implementation details.

6.1 Query and Result Presentation

Since our search aims at lay end users, both intermediate and final results have to be presented to the user in an intuitively understandable way. We have developed a concept for visualizing query graphs along with results. We choose a table layout as it is a general pattern that can accommodate queries and results of different structures and complexities. As shown in Fig. 7, the table is divided horizontally into two sections from top to bottom: a query view and a result view.

The query graph selected by the user is presented in the query view. There is one column for every variable of the query. The column labels denote classes or data types and arrows between them represent relations or attributes respectively. This information is encoded in the query space. In particular, labels for columns and arrows can be derived from the labels of elements contained in query graph. The result view shows variable bindings, i.e. entities (or data values) that satisfy the constraints specified in the query.

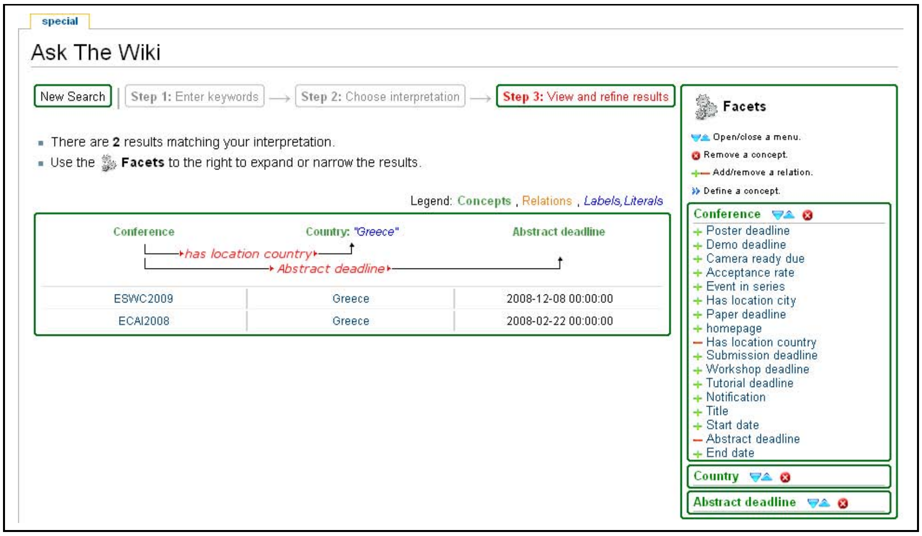
This table-based presentation template is sufficiently general for dealing with different structures, the following three types of queries in particular:

1. *Entity Queries* ask for specific entities like searching a person, e.g. someone with the name “Thanh Tran”. For this, one single column is needed to display the class of the entity in question.
2. *Fact Queries* ask for a concrete relation (an attribute) of a particular entity like the mail address of “Thanh Tran”. Two columns are needed, one for the class of the entity in question and another for displaying the requested data value (or entity).
3. *General Graph-structured Queries* ask for n-ary tuple sets. Several columns and arrows are needed to show the classes, data types, relations and attributes mentioned in the query.

6.2 Facet-Based Query Modifications

When an interpretation is chosen, the facets view is shown to the user. This view is depicted on the right side of the screenshot in Fig. 7. Using this view, facets are displayed for the different entity classes mentioned in the query. In particular, the facets view contains windows for every class shown in the query view. These windows display the possible facets, which are relations or attributes that can be derived from the schema space, i.e. outgoing edges of the C-vertex denoting the selected class.

Using this facets view, query modifications are supported by adding or deleting facets. For example, in Fig. 7 the user can drill down and refine the search result by adding the facet “submission deadline” to the class “Conference”. The user can more


special

Ask The Wiki

New Search | Step 1: Enter keywords → Step 2: Choose interpretation → Step 3: View and refine results

- There are **2** results matching your interpretation.
- Use the **Facets** to the right to expand or narrow the results.

Legend: Concepts , Relations , Labels,Literals

has location country
Abstract deadline

Conference	Country: "Greece"	Abstract deadline
ESWC2009	Greece	2008-12-08 00:00:00
ECAI2008	Greece	2008-02-22 00:00:00

Facets

Open/close a menu.
Remove a concept.
Add/remove a relation.
Define a concept.

Conference
+ Poster deadline
+ Demo deadline
+ Camera ready due
+ Acceptance rate
+ Event in series
+ Has location city
+ Paper deadline
+ homepage
– Has location country
+ Submission deadline
+ Workshop deadline
+ Tutorial deadline
+ Notification
+ Title
+ Start date
– Abstract deadline
+ End date

Country

Abstract deadline

Fig. 7. Screenshot of our semantic wiki search implementation, which exploits semantic models throughout the search process. In the center of the screenshot, the selected query “Abstract deadlines of conferences located in Greece” along with the results are shown. On the right side is the menu providing the facet-based refinement capabilities.

precisely define this facet by entering a concrete value for "submission deadline". The new query is immediately evaluated and the results are presented to the user. Instead of adding facets, the user can also expand the search results by removing the facet "has located country". He would then get all conferences and their abstract deadlines without the constraint that they are located in Greece. Furthermore, the user can remove a query variable and all facets bound to it by pressing the "x" button in the head line of the respective facet window (on the right of Fig. 7).

According to the user study we have presented in [15], the majority of 14 users found the presentation of the results understandable. Unlike keyword search, faceted search seems to be not a commonly used paradigm. Three participants stated that they did not know how to do it. However, the ones who used it found the feature helpful. Interestingly, the use of facets was particularly effective for the more complex tasks.

Note the same principle that has been applied throughout the many steps of the semantic search process. Semantics has been used to enable a more focussed exploration of query interpretation and to enable a more guided matching of query graphs respectively. In this section, semantics is used to guide the user through the process. Based on the query space, the query and answers are presented according to their structures to facilitate user comprehension. The schema space represents a compact view over the answer space. It is used to derive facets that help the user in modifying and refining the answer space.

7 Related Work

There exist many different conceptions and definitions for semantic search [1,4,5]. A state-of-the-art analysis can be found in [3], which provides a review of different semantic search tools and focuses on different modes of user interaction. In this paper, we regard search as a process. We have discussed how semantics can be exploited throughout the process. With respect to existing concepts, we offer a novel, process-centric perspective on sematic search.

The other dimensions of related work concern with the steps involved in the search process, which we will briefly discuss in the following.

Query Construction. Much work has been carried out in order to facilitate query construction. This daunting task is mainly addressed in approaches on relaxed-structured query models [16–19] and the structure free keywords-based query model. For the keywords-based search, native approaches can be distinguished from the ones that extend existing databases with keyword search support. Native approaches operate directly on the data (i.e. on the resource graph), and thus have the advantage of being schema-agnostic [20–22]. Database extensions require a schema, but can leverage the infrastructure provided by an underlying database. Example systems implemented as database extensions are DBXplorer [23] and Discover [24]. These systems translate keywords to candidate networks, which are essentially join expressions constructed using information given in the schema. Thus, instead of using the resource graph, the exploration for join expressions (i.e. queries) operate on a smaller search space based on the schema. Our approach for query interpretation combines the advantages of these two approaches: in line with native approaches, it is also schema agnostic in that a schema space is automatically derived from the resource space. Unlike the natives approaches, the exploration does not operate directly on the resource space, but on the schema space.

Query Processing. The answer space used in our approach is similar to a structure index, a concept that has been widely used for semi-structured and XML data [25–29]. In particular, *dataguide* [30] is a well-known concept that has been proposed for rooted graphs, i.e. graphs having one distinguished root node. A strong dataguide is established by grouping together nodes sharing edge label sequences of incoming paths starting from the root. As opposed to our answer space, the resulting grouping is not a partition, i.e. one vertex may be assigned to several blocks. Thus, the size of the dataguide can get exponentially larger than that of the original data graph. The *1-Indices* [25, 26] prevent this worst-case exponential blow-up.Instead of backward bisimulation only, both back- and forward bisimulation is employed for the construction of a covering index for XML branch queries [27].

The main difference is that while we can derive an answer space from general graph-structured data, the construction techniques used for the indexes mentioned above rely on the resource graph being rooted thereby imposing a structural constraint on the resources that is hard to realize. Particularly in the Semantic Web context, the graph-structured model (i.e. RDF) has been explicitly designed for representing information from diverse sources, which might have to be integrated in a non-hierarchical way.

Result Presentation and Query Refinement. For the presentation of structured results and the refinement of queries, faceted browsing is increasingly used in search applications. Many websites already feature some sort of faceted search to improve the precision of their search results. A crucial aspect of faceted search is the design of a user interface. This has been studied by [31, 32] and applied in systems like Flamenco[5], Exhibit[6] or Parallax[7]. In a Semantic Wiki context, this paradigm has been applied in the form of Semantic Drill Down[8] for browsing from top to bottom along the wiki's categories. Another cornerstone of faceted browsing is the question what is actually used as facets, which obviously depends on the resources and theirs structures. Systems like Flamenco and Exhibit require a predefined set of properties, which are used as facets. We make explicit use of the schema space to compute the relevant set of facets in a dynamic way.

8 Conclusions

The problem of semantic search, i.e. addressing information needs at the level of the meaning, has received increased attention in recent years. Numerous approaches to semantic search have been proposed that make use of semantics in different ways. In this

[5] `http://flamenco.berkeley.edu`
[6] `http://simile.mit.edu/exhibit`
[7] `http://mqlx.com/~david/parallax/`
[8] `http://semantic-mediawiki.org/wiki/Help:SMW\_extensions# Semantic\_Drilldown`

paper we have described semantic search from a process perspective that considers all the relevant steps of information access. Further, we have discussed how lightweight semantics can be exploited throughout this search process, i.e. using graph-structured models of resources, schemas, queries, and answers. These lightweight semantic models clearly go beyond the state-of-the-art in information retrieval, as they do not treat the resources on a word level, but instead more explicitly capture the semantics of the elements. At the same time, the complexity of these graph-structured models and associated algorithms are still computationally manageable, such that they can be applied to search problems on a large scale. Specifically, based on these semantic models, we have presented algorithms for constructing structured queries from keywords, for answering these queries as well as for the presentation and further refinement. The search process and presented techniques have been successfully applied in a number of semantic search systems, such as [15] and [14].

Acknowledgements

This work was partially supported by the European Commission in the IST projects NeOn (IST-2006-027595) and X-Media (IST-FP6-026978). Also, we would like to thank our colleagues from the APEX Lab of the Shanghai Jiao Tong University (especially Haofen Wang) for the fruitful discussions.

References

1. Guha, R.V., McCool, R., Miller, E.: Semantic search. In: WWW, pp. 700–709 (2003)
2. Mangold, C.: A survey and classification of semantic search approaches. International Journal of Metadata, Semantics and Ontologies 2(1), 23–34 (2007)
3. Uren, V.S., Lei, Y., Lopez, V., Liu, H., Motta, E., Giordanino, M.: The usability of semantic search tools: a review. Knowledge Eng. Review 22(4), 361–377 (2007)
4. Zhang, L., Yu, Y., Zhou, J., Lin, C., Yang, Y.: An enhanced model for searching in semantic portals. In: WWW, pp. 453–462 (2005)
5. Chu-Carroll, J., Prager, J.M., Czuba, K., Ferrucci, D.A., Duboué, P.A.: Semantic search via xml fragments: a high-precision approach to ir. In: SIGIR, pp. 445–452 (2006)
6. Chen, P.P.: The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. 1(1), 9–36 (1976)
7. Staab, S., Studer, R. (eds.): Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004)
8. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: ICDE (to appear, 2009)
9. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: Chan, C.Y., Ooi, B.C., Zhou, A. (eds.) SIGMOD Conference, pp. 305–316. ACM, New York (2007)
10. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
11. Tran, D.T., Rudolph, S., Ladwig, G.: Efficient rdf query processing through structure-aware rdf graph matching and structure-based partitioning: A technical report (2009), `http://sites.google.com/site/kimducthanh/research/strucIdx-TR.pdf`
12. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable semantic web data management using vertical partitioning. In: VLDB, pp. 411–422 (2007)

13. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. In: PVLDB, vol. 1(1), pp. 1008–1019 (2008)
14. Wang, H., Penin, T., Xu, K., Chen, J., Sun, X., Fu, L., Liu, Q., Yu, Y., Tran, T., Haase, P., Studer, R.: Hermes: A travel through semantics in the data web. In: SIGMOD Conference (to appear, 2009)
15. Haase, P., Herzig, D., Musen, M., Tran, T.: Semantic wiki search. In: ESWC (to appear, 2009)
16. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: A semantic search engine for xml. In: VLDB, pp. 45–56 (2003)
17. Li, Y., Yu, C., Jagadish, H.V.: Schema-free xquery. In: VLDB, pp. 72–83 (2004)
18. Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: Flexpath: Flexible structure and full-text querying for xml. In: SIGMOD Conference, pp. 83–94 (2004)
19. Al-Khalifa, S., Yu, C., Jagadish, H.V.: Querying structured text in an xml database. In: SIGMOD Conference, pp. 4–15 (2003)
20. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE, pp. 431–440 (2002)
21. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
22. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: SIGMOD Conference, pp. 305–316 (2007)
23. Agrawal, S., Chaudhuri, S., Das, G.: Dbxplorer: enabling keyword search over relational databases. In: SIGMOD Conference, p. 627 (2002)
24. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB, pp. 670–681 (2002)
25. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding structure to unstructured data. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 336–350. Springer, Heidelberg (1996)
26. Milo, T., Suciu, D.: Index structures for path expressions. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 277–295. Springer, Heidelberg (1998)
27. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: Franklin, M.J., Moon, B., Ailamaki, A. (eds.) SIGMOD Conference, pp. 133–144. ACM, New York (2002)
28. Kaushik, R., Shenoy, P., Bohannon, P., Gudes, E.: Exploiting local similarity for indexing paths in graph-structured data. In: ICDE, pp. 129–140. IEEE Computer Society Press, Los Alamitos (2002)
29. Qun, C., Lim, A., Ong, K.W.: D(k)-index: An adaptive structural summary for graph-structured data. In: Halevy, A.Y., Ives, Z.G., Doan, A. (eds.) SIGMOD Conference, pp. 134–144. ACM, New York (2003)
30. Goldman, R., Widom, J.: Dataguides: Enabling query formulation and optimization in semistructured databases. In: VLDB, pp. 436–445 (1997)
31. Hearst, M., Elliott, A., English, J., Sinha, R., Swearingen, K., Yee, K.P.: Finding the flow in web site search. Communications of the ACM 45(9), 42–49 (2002)
32. Hearst, M.: Design recommendations for hierarchical faceted search interfaces. In: ACM SIGIR Workshop on Faceted Search (August 2006)

Human Being and Mathematics Logical and Mathematical Thinking*

Rudolf Wille

Technische Universität Darmstadt, Fachbereich Mathematik,
Schloßgartenstr. 7, D–64289 Darmstadt
wille@mathematik.tu-darmstadt.de

Summary. *Logical thinking* as an expression of human reason grasps the actual reality by the basic forms of thinking: concept, judgment, and conclusion. *Mathematical thinking* abstracts from logical thinking to disclose a cosmos of forms of potential realities hypothetically. *Mathematics* as a form of mathematical thinking can therefore support *humans* within their logical thinking about realities which, in particular, promotes sensible actions. This train of thought has been convincingly differentiated by *Peirce's philosophical pragmatism* and concretized by a "*contextual logic*" invented by members of the mathematics department at the TU Darmstadt.

1 Logical Thinking

Already Pythagoras' pupil Alkmaion of Croton defined a human being as "zoon logon echon" (in latin: animal rationale), i.e. as "reasonable living being". This basic anthropological understanding of a human being has been lasted in western philosophy until Scheler's duality of "mind" and "body" and even further ([Fa73]; p.895). *"Reason"* is here understood as mental means of human beings to gain insights, to form judgments, and to act in accordance to those judgments ([Du95]; p.3694). Since those means are substantial for human beings, the formation of humans should achieve to learn thinking and acting in a reasonable manner. To what extent mathematics could play a role here, this shall be discussed in the following. In particular, the claim shall be examined that logical thinking can be supported by mathematics. How close are the meanings of "thinking logically" and "thingking reasonably", this may become clear by noticing that the meanings of both linguistic expressions are apprehended in English by one word, the verb "reason".

To understand what is meant by "logic thinking", one has to clarify what is meant by "logic". According to the "Duden: Das große Wörterbuch der deutschen Sprache", *logic* is the doctrine of the structure, the forms, and the laws of thinking ([Du95]; p.2145). Therefore, "logical thinking" means a thinking which activates logical (i.e. to logic belonging) structures, forms, and laws. In the philosophy since the 16th century, the basic forms of logical thinking are considered as the *concepts* (as basic units of thinking), the *judgments* (as connections between concepts), and the *conclusions* (as inferences gaining judgments from other

* This article is an English version of the German publication [Wi01b].

judgments). How, founded on those basic forms, further logical structures, forms, and laws can be developed, this has already been shown 1662 by Antoine Arnauld and Pierre Nicole in their guiding book "La Logique ou l'Art de penser" [AN85]. Even when a new understanding has become dominant in the course of the stronger formalization of logic - after Logic was ascribed to the task of recognizing the laws of truthness (since the midle of the 19th century) - , the everyday uses of logical thinking has basically speaking not changed if one is orientated on general dictionaries.

This understanding of logical thinking, that it is based on concepts as the basic units of thinking, has been further deepened by *Jean Piaget* in his structure-genetic theory of cognition. For him, the logical thinking of a human being has its roots in the *coordinations of actions* which are already present before the development of the language; from those coordinations, mental operations and with them logical structures come into being in the psychic development ([Pi73], p.26ff). Piaget's approach, by which he tries to clarify the question about the logic of conceptual thinking and the truthness of knowing, consequently run out according to Thomas Bernhard Seiler toward a theory which understands concepts as basic units of recognizing, thinking, and knowlege. Piaget identifies concepts with cognitive structures with which and through which the organism examines its environment in an acting manner, adapt to it, and in which the organism reconstructs the aspects of the environment relevant for its acting and thinking and which provide for it the basis for interpreting the meaning of signs (cf. [Se01]; p.164f.).

Most simple preforms of concepts are the *sensorimotor schemas* which arise already early out of coordinations of actions. The next step of evolution forms the *ideas* which abstract from the observed objects and correlated actions. If such structures of cognition can also be applied to new objects and other structures of cognition, then Piaget speaks of *preconcepts*. Structures of cognition have finally reached the step of *concepts* if they have been freed to a large extent from the intuitive view and have let coordinated to formal operations. Only the construction of complex concept systems and their systematic coordination allows a differentiated reconstruction of reality and leads to consistant concept orders, the availability of which is a necessary condition for the development of logical thinking. Piaget sees further conditions in the system properties of complex structures of action, the necessity of communicative negotiation and the compulsion to justify herself in the society ([Se01]; p.171).

Which priority meaning the logical thinking has for the recognition and action of human beings, this has been made distinct in particular by *Charles Sanders Peirce* in his philosophical pragmatism. For this the Cambridge Conferences Lectures are an impressive source which Peirce has given in 1898 about the theme *"Reasoning and the Logic of Things* [Pe92]. These lectures offer an introduction into Peirce's late philosophy which tries to make it intelligible for all. Logic is understood in this lectures as normative science about forms and laws of thinking which, as a philosophical discipline, has as theme to make understandable the relationship between thinking and reality. Peirce sees the foundation for the understanding of forms of thinking in his *categories* of Firstness, Secondness, and Thirdness which he defines as follows ([Pe92]; p.146ff.): *Firstness* is the mode in which anything would be only for itself, irrespective of anything else; *Secondness* is the mode in which anything would be related to something else, irrespectively of anything third; *Thirdness* is the mode in which a First is joined with a Second by a Third. For instance, a concept as a third joins a concept word as a First with an object as a Second.

According to his categories, Peirce distinguishes between three kinds of logical conclusion: the abduction, the induction, and the deduction. The *abduction* creates out of the horizon of self-evidence a hypothesis as a First; the *induction* confirms a hypothesis by actually given facts as a Second; the *deduction* concludes a hypothesis out of valid premisses by logical laws as a Third. This means: "The deduction proves that something *must* be the case; the induction shows that something is *actually* efficient; the abduction only assumes that something *might be* the case ([Pe91]; p.400). In his Cambridge Lectures Peirce elucidates the three kinds of logical conclusions by the syllogistic figures of conclusion: the deduction by the figure Barbara, the induction by the figure Datisi, and the abduction (retroduction) by the figure Cesare ([Pe92]; p.141f.); with that he clarifies in particular that the three figures of conclusion distinghish essentually from each other, which Imanuel Kant challenced 1762 in his paper "Über die falsche Spitzfindigkeit der vier logistischen Figuren" ([Ka83a]; p.597ff.). With the reached understanding of the triadic nature of the logical conclusion, Peirce overcame the difficulties to express geometric and algebraic conclusions by syllogisms in the way that he extended the Boolean logic [Bo58] to the *logic of relations* ([Pe92]; p.150ff.), which was for him the formal foundation for all logical conclusions. The limitation of syllogisms, which was for Peirce essentially depend on their mechanistic nature, becomes surmounted in the logic of relations by an open diagrammatic conclusion which gives space for different types of conclusions.

Logical thinking was generally characterized by Peirce as follows: "Reasoning is the process by which we attain a belief which we regard as the result of previous knowledge" ([Pe98]; p.11). Peirce discusses in his first Cambridge Lecture about "Philosophy and the Conduct of Life" the logical thinking in everyday life, which succeeds for him as well without help by theoretical logic as with it. Primarily he sees the logical thinking detemined by the instinct and the sentiment of human beings and warns therefore for superficial logical conclusions that does not pay attention to instict and sentiment. As the logical thinking grows out of the human expierence, so instinct and sentiment develop in human beings from inner and outer experiences, and that takes place in a slow and deep process which brings out mental energy and vitality. Peirce considers this process as so important that he views *instinct and sentiment* as the real substance of the human mind ([Pe92]; p.110).

For this reason, the *"training in reasoning*" - so the theme of the fifth Cambridge Lecture - must, according to Peirce, concern the human mind as a whole; for this three mental operations are important for him: observation, experimentation, habituation.

Observation consists of two parts: the first as subconscious induction by which an associational potency arises on repeatedly reviewing an object of perception with a tendency to call up other ideas; the second as conscious formation of schematic ideas which are able to react on perceivable objects. The associational potency which arises out of the subconcious induction is according to Peirce the most important constituent of practical thinking, while the consciously formed schematic ideas are indispensable for theoretical thinking ([Pe92]; p.182). For logical thinking it is particularly important to train powers of discrimination; according to this, Peirce writes: "I never knew a man whose sagacity as a reasoner compelled my admiration without finding in him a considerably cultivated discrimination" ([Pe92]; p.183). For the observation the most important precondition is passivity, i.e. not to give way to the natural presure to immediately mix the observation with own ideas.

For the *experimentation* however, an active energy, a persistence, and a strong contribution of will is essential. For Peirce there is no doubt that, what ever strengthens the will also strengthens the power of logical thinking ([Pe92]; p.187). Experimentation needs furthermore a certain measure of resourcefullness, i.e. of movability of the creative imaginative faculty, of flair for significant questions and answers as well as of persistence to clarify advantages and disadvantages of different answers. For training logical thinking one should again and again be activated to experiment systematically; for this, systematic recordings are indispensable. In general, Peirce recommends to record on paper cards all what is noteworthy. For an eager student Peirce estimates approximately 20.000 paper cards per year by which he can built up a rich treasure of experience for his experimental thinking.

Habituation contains as mental operation the power of readily taking habits and of readily throwing them off; for Peirce there is no habit more useful than this habit taking up and easily throwing off mental habits ([**?**]). Important for logical thinking is to win new connections of thoughts; the necessary readiness to take up something new determines also the readiness to give up something old. For Peirce the learner of logical thinking has therefore to be like a child with all its uprightness and naivety of childlike imaginations and all of the plasticity of childlike states of mind. By reading a lot the aimed flexibility of thinking can be trained; for Peirce, reading 50 up to 100 books in a year would be desirable. The right way of reading consists in trying to understand the author and to assimilate his style of thinking. According to Peirce, the power of habituation can be improved in three directions: by exercises in distinguishing and classifying, by exercises in defining and logically analysing of ideas, and by excises in compressing theories and trains of thought ([Pe92]; p.192).

The distinct openess of logical thinking has worked out by Peirce mostly in his fourth Cambridge Lecture on "The first rule of logic". After this basic logical rule, logical thinking show a tendency to correct itself and that is not only by its conclusion, but also by its premisses ([Pe92]; p.165). The quality of self-correction, which already G. W. F. Hegel has considered as constitutive for the dialectic process of growing reason [He86], is important for the logical thinking of each kind of science. Peirce realizes that "research of every type, fully carry out, has the vital power of self-correction and of growth. This is a property so deeply saturating its inmost nature that it may truly be said that there is but one thing needfull for learning the truth, and that is a hearty and active desire to learn what is true" ([Pe92]; p.170).

The self-correction of logical thinking stands in the direct connection with another property of logical thinking, that is the principal *criticizability.* This property has, according to Peirce, to be understood first of all as sense-critics in the view of the *pragmatic maxim*, which in particular founds a connection between logic and ethics. Peirce writes in 1902/03: " ... which makes logic and ethics to peculiar normitive sciences is this: nothing can be logically true or morally good without a purpose in regard to that it can be named. Since a sentence and in particular the conclusion of an argument which would be only accidentally true, that is not logic" ([Ap75]; p.175). 1903 Peirce finished his Havard-Lectures about pragmatism with the maxim: "The elements of each concept enter into the logical thinking through the door of perception and go out again through the door of purposeful action; and all, what cannot be identified at the two doors, has to be detained as not authorized by the reason" ([Pe91]; p.420).

2 Mathematical Thinking

With the theme "Human Being and Mathematics" the relationship of logical thinking and mathematical thinking shall be examined in this contribution; therefore the *mathematical thinking* shall now be considered in more detail. To keep the connection with logical thinking in mind, it shall be first explained how Peirce makes mathematics and mathematical thinking in his Cambridge Lectures on *"Reasoning and the Logic of Things*" [Pe92] to his theme. For the authors of the extensive introduction for the first complete edition of these lectures, Keneth Laine Ketner and Hilary Putnam, the mathematics in the lectures play such a dominant role that they could even prefer the titel *"The Consequences of Mathematics*". For this they stated several reasons: First Peirce had already planed and elaborated some provisional lectures as advanced contributions stimulated by the invitation to give a lecture series; these lectures were primarily planed mathematical. When he as well under the pressure of his promotor William James took back considerably the mathematical parts - because of the general understandability - , the basic character of mathematics however remained in the lectures. This links with a second reason that namely Peirce understood his philosophy, under which in the lectures also the logic is incorporated, as a consequence of mathematics. Thirdly Ketner and Putnam see in the expression "Consequences of Mathematics" an even deeper lying importence; they write: "Peirce argued that, epistemologically at any rate, mathematics was an observasional, experimental, hypothesis-confirming, inductive science that worked only with pure hypotheses without regard of their application in "real" life. Because it explored the consequences of pure hypotheses by experimenting upon representative diagrams, mathematics was the inspirational source for the pragmatic maxim, the jewel of the methodological part of semeotic, and the distinct feature of Peirce's thought" ([Pe92]; p.2).

At the end of his first Cambridge Lecture Peirce classifies the sciences ordered by the abstractness of its objects. He places *mathematics* as the most abstract of all sciences, because mathematics is for him the only science which is not concerned to explore what the actual facts are, but inquires hypotheses ([Pe92]; p.114). The objects of mathematics have consequently no actual existence, but are only modi of potential being. The goal that the pure mathematics approaches by making stepwise accessible an expending cosmos of forms of abstract thinking, that is - in the long run - the *potential world of reality.* As the formal science of potential reality, mathematics delivers formal-hypothetical foundations for all other sciences and humanities. In this sense logic is founding on mathematics. Thus Peirce judges: "All necessary logical reasoning is strictly speaking *mathematical reasoning*, that is to say, it is performed by *observing* something equivalent to a mathematical diagram" ([Pe92]; p.116). For the mathematical reasoning Peirce has developed as a kind of algebraic logic the mathematical logic of relations which he introductary explains in his third Cambridge Lectures.

To understand better how mathematical thinking is able to develop a mutual play between abstracting and concretizing, respectively, and to make it effective in the thinking and acting of human beings, the nature of mathematical thinking shall be made more understandable. For this, opinions and discoveries shall be used which Philip Kitcher explains in his book *"The nature of mathematical knowledge*" [Ki84]. For Kitcher there are three obvious insights: "First, we originally acquire much of our mathematical knowledge from teachers, on whose authority we accept not only basic principles but also conceptions of the nature of mathematical resoning. Second, some of this knowledge is acquired with the help of perceptions. Our early training is aided by the use of rods and beads; later, we appeal to diagrams. Third, mathematics has a long history. The origins of mathematical knowledge lie in the practical activities of Egyptians and Babylonians (or, perhaps, people historically are more remote)" ([Ki84]; p.91f.). Kitcher worked out these insights in his book to a convincing *Theory of the Mathematical Thoughts and Knowledge.* This process began in the earliest time with rudimentary perceptions and ideas which developed a first understanding of an arithmetic of small numbers and of a geometry of simple plane figures. Out of those roots, a mathematical thinking has been developed erected on existing knowledge, respectively, and renewed by changes for which Kitcher dicusses in detail the general activities of answering questions, generating questions, generalizing, rigorous changing and systematizing; in doing so, he examines the process of development of mathematical thinking in the sense of Kuhn's thesis that scientific change means a change of practice and not only of theory.

Kitcher explains the relationship of mathematical thinking to the real world in particular at *general actions of thinking* as collecting, segregating, combining, correlating etc. and their idealizations to mathematical operations of thinking. For example, he represents the set theory as an idealized theory of forming collections. How fruitful those mathematical idealizations of general actions of thinking are can be demonstrated by the action of thinking *"summerize to a whole"* which is based on Cantor's definition of sets. For instance, in the script of a lecture on "Linear Algebra I" mathematical elements have been summerize to sets as a whole which can be demonstrated as follows:

1. the real numbers to the whole $\mathbf{R}$ of all real numbers,
2. the triples of real numbers to the analytic representation $\mathbf{R^3}$ of the space of intuition,
3. the sections of the same length and direction to a vector,
4. scalars to a matrix,
5. the even numbers to the binary cipher 0 and the odd numbers to the binary cipher 1,
6. objects, attributes, and a joining relation to a formal context,
7. elements with the same properties to a set,
8. the subsets of a set S to the power set $\mathbf{P}(S)$,
9. a family of sets to their union, to their intersection, and to their direct product,
10. ordered pairs of sets to a relation,
11. equivalent elements to an equivalence class,
12. the equivalence of an equivalence relation to the appertaining quotient set,
13. relating arrows to a mapping,
14. the permutations of a set M and their concatenations $\circ$ to the symmetric group S_M,
15. the symmetries of a geometric figure F and their concatenations $\circ$ to to the symmetry group $Sym(F)$,
16. the cosets of a normal subgroup and the representational association to the appertaining quotient group,
17. the real numbers with addition and multiplication to the field $\mathbf{R}$ of the real numbers,
18. scalars to an n-tuple,
19. the n-tuples of elements of a field K and their componentwise additions and multiplication with a scalar to the vector space K^n,
20. the algebraic structures in which the vector space axioms are valid to the concept of the vector space,
21. elements of a vector space and the apppertaining scalars to a linear combination,
22. all linear combinations of elements $a_1, ..., a_k$ of a vector space to the subspace $< a_1, ..., a_k >$ generated by the given elements,
23. the elements of a vector space which a linear mapping ϕ maps on 0 to the subspace $Ker\phi$,
24. linear equations to a linear system of equations,
25. the solutions of a linear system of equations in n-variables to the affine subspace of the vector space K^n.

The mathematical examples make clear that the combination to a whole may end up quite different depending on what is formally mend by combining to the whole. What can hardly be differentiated in the common language may become transparent in the mathematical language: In 1., 2., 3., 5., 7., 8., 10., 11., 12., 13., 22., 23., 25., the forming of sets are of different nature; also 24. could be seen as a forming of sets, but the word "system" indicates that there is more what is expressed in equalities of variables. The formation of tuples and matrices in 18. and 4. are usually not considered as set formation, just as the structure formations in 9., 14., 15., 16., 17., and 19. In 6. and 21. one has sets and elements, respectively, which are formed by terms and in 20. by concepts. Further differentiations are obtained when the combined whole is mathematically characterized, which however shall not be elaborated. An extensive investigation of mathematical thinking in linear algebra has been presented by Katja Lengning and Susanne Prediger in [LP00].

On the basis of the rich treasure of mathematical forms, *the mathematical thinking* has the special ability to formally arrange and structure contents of thinking in great variety, by which more transparency and clearness can be usually gained. For Martin Heidegger this ability is even characteristic for modern thinking, and that is in the sense that not only the content is arranged by forms of thinking, but that also the content is understood at all by the corresponding forms of thinking. Heidegger sees this basic character of modern thinking and knowledge in the knowledge claim which he calls the *"mathematical"*. About this, Heidegger writes in his book "Die Frage nach dem Ding": "The mathematical is that basic position to the things in which we propose the things to what they are already given. The mathematical is therefore the basic assumption about the knowledge of the things" ([Hd62]; p.58). Mathematical thinking can hence not only be understood by the lexical meaning as the thinking belonging to mathematics, but more general as a thinking of forms able to the design which according to Heidegger is set "for which we actually consider the things, as what they are acknowledged in advance" ([Hd62]; p.71). Then the mathematical thinking is not explainable out of mathematics, but the *mathematics* is itself only a certain formation of mathematical thinking. Such an understanding of mathematics is closely related to the view which Reuben Hersh propagates in his book "What is Mathematics, Really?" [Hr97]. The historical, social-cultural forming of mathematics can be understood in such a way that out of figures and operations of mathematical form-thinking, which are again and again activated in communications, formal systems of thinking are formed in a process of a progressive conventionalizations and constituted out of this a culture of thinking which is called *"mathematics"* [Wi00a], [Wi01].

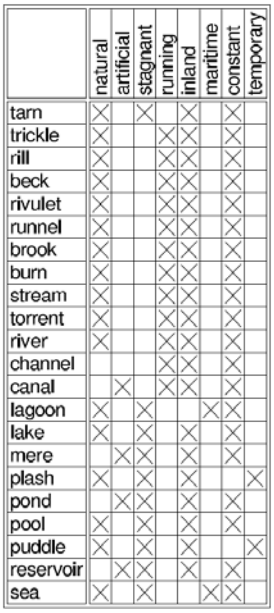
3 Human Being, Mathematics and Reality

The previous discusion about the relationship of human being and mathematics started from the understanding that it is intrinsic for a human being to think and to act reasonable, i.e. in particular to win insights, to form a judgment, and to follow after that in all actions. That mathematics supports the reasonable thinking and acting has its central reason in the *close connection of logical thinking and mathematical thinking.* Therefore the effort is worth to understand this connection between the logical and the mathematical and to make it effective. According to Peirce's pragmatic maxim, this means that logical thinking in his relationship to reality should be mathematized in a way that the connection of mathematizing with the manifold of potentially appertaining realities can be better understood and activated.

An attempt to that has been made in our "Darmstadt Research Group on Concept Analysis" with the elaboration of a *"contexual logic"* which is understood as a mathematization of the traditional philosophical logic with its doctrines of concept, judgment, and conclusion [Wi00b]. The basis of this philosophical logic underlies the view that the human recognition and thinking activates the basic logical structures concept, judgment, and conclusion by bringing realities under concepts, forming judgments from concepts, and concluding judgments out of other judgments. On this base, Gottlob Benjamin Jäsche makes clear in his introduction to the logic-lectures of Imanuel Kant (edited by Jäsche) with Kant's explicit explanation that "it is nothing else allowed to include in the actual treatise of logic and particularly in the elementary treatise as the theory of the three essential main functions of thinking - the concepts, the judgments, and the conclusions ([Ka83b]; p.424). Since the contextual logic is elaborated as a mathematical theory the basic structures of which are abstracted out of the traditional philosophical logic (cf. [Pr00]), the contextual logic is classified in a "contextual concept logic", a "contextual judgment logic", and a "contextual conclusion logic"; in its whole, the contextual logic is founded on the set-theoretic semantics of modern mathematics.

For the *contextual concept logic* it is first to answer the basic question: What is the properly abstracting linguistic set definition of the concept of concept? According to Piaget, concepts are cognitive structures which can only fulfill their task of the differentiating reconstruction of the reality, when they can be coordinated systematically and constructed by its complex concept systems; concepts are therefore formed in a relational structure which is constitutive for them. Therefore it counts first of all to introduce relational structures for creating abstract concepts as set structures in the greatest possible generality. That became successful - as rich experiences in the last thirty years have shown - with the conception of the *formal context* formed by objects, attributes, and a joining relation. A *"formal context"* is defined as a set structure (G, M, I) which consists of two sets G and M and a relation I between the sets G and M; the elements of G are called $(formal)$ *objects*, the elements of M are called $(formal)$ *attributes*, and the relational connection gIm is read: the object g has the attribute m. In the sense of Peirce's categories, an object is considered in a formal context as a First with an attribute as a Second which are linked by the context relation as a Third. ($Fig.$ 1)

Formal contexts can be understood as mathematization of real-world cross-tables. For instance, the *cross-table* presented in $Fig.$ 1, which is taken out of the publication "Kontrastive Untersuchung von Wortfeldern im Englischen und Deutschen" [Kr79], can be abstracted to a formal context (G_W, M_W, I_W) as follows: the object set G_W consists out of words of the investigated semantic



	natural	artificial	stagnant	running	inland	maritime	constant	temporary
tarn	×		×		×		×	
trickle	×			×	×		×	
rill	×			×	×		×	
beck	×			×	×		×	
rivulet	×			×	×		×	
runnel	×			×	×		×	
brook	×			×	×		×	
burn	×			×	×		×	
stream	×			×	×		×	
torrent	×			×	×		×	
river	×			×	×		×	
channel				×	×		×	
canal		×		×	×		×	
lagoon	×		×			×	×	
lake	×		×		×		×	
mere		×	×		×		×	
plash	×		×		×			×
pond		×	×		×		×	
pool	×		×		×		×	
puddle	×		×		×			×
reservoir		×	×		×		×	
sea	×		×			×	×	

Fig. 1. Formal context partly representing a lexical field "bodies of waters"

field "waters" in [Kr79] and the attribute set M_W consists of noems (smallest elements carring a meaning) by which the words are characterized according to their contents, and the relation I_W are grasped by the relationships which are indicated by the crosses; i.e. the mathematical expression "$puddle I_W temporary''$ stands for the linguistic relationship "the word 'puddle' has the noem 'temporary' " indicated by a cross in the cross-table. In general, the cross-table has to be distinguished from the formal context which is abstracted from the cross-table; thus, a cross-table has a logical structure with which real relationships can be presented, but a formal context is a mathematical structure which first of all challenges the activation further mathematical structures and connections. In spite of their location, cross-table and formal context form a model for the close connection of logical and mathematical thinking.

For the mathematization of 'concept', the formal context as mathematization of the nessecary relational structure can now be assumed: A *formal concept* of a formal context (G, M, I) is defined as a pair (A, B) where A is a subset of G and B is a subset of M so that A consists of all those objects in G which have all attributes of B and B consists of all those attributes in M which apply to all objects in A; A is named the *extent* and B is named the *intent* of the formal concept (A, B). This mathematization proceeds from the philosophical understanding of concept; according to that, a concept is a unit of thought consisting of an extension and an intension, as it was already presented by the logic of Port Royal [AN85] in the 17th century (cf. also [Wa73], [Wi95]). A formal concept (A, B) of (G, M, I) is called a *subconcept* of a formal concept (C, D) in

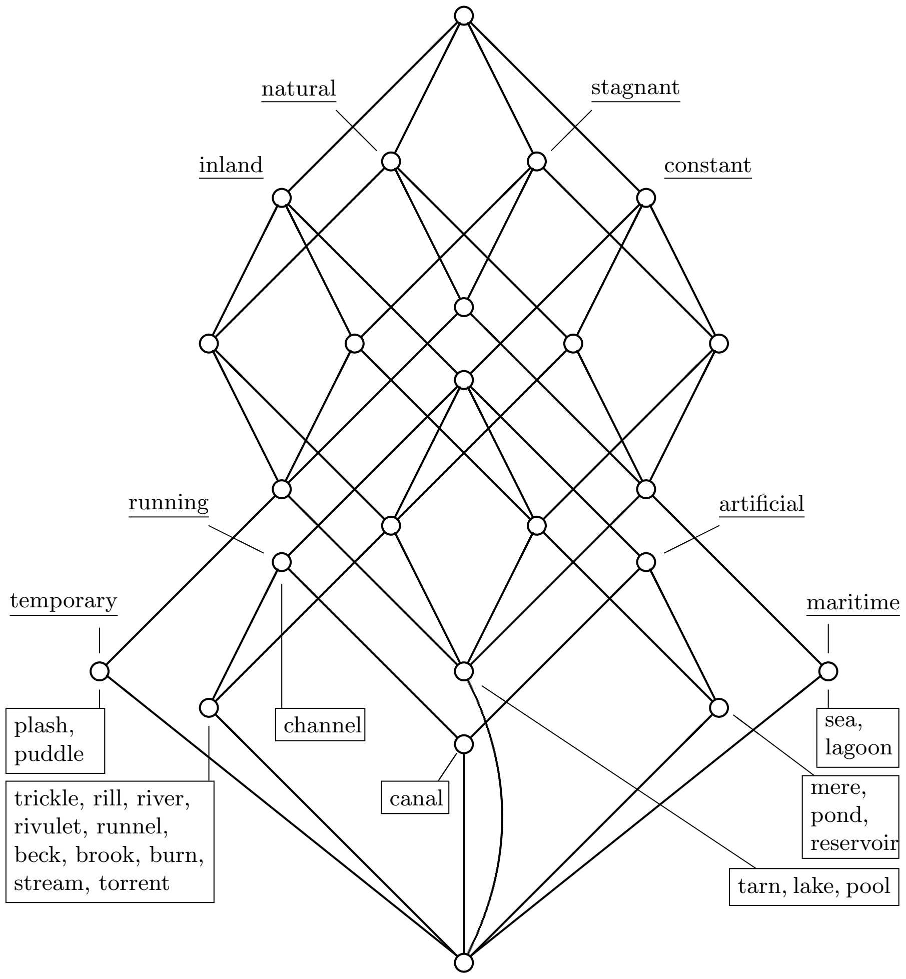


Fig. 2. Concept lattice of the formal context in Fig. 1

(G, M, I) and (C, D) a *superconcept* of (A, B) if the extent A is contained in the extent C and, equivalently, if the intent B contains the intent D.

The *logical reciprocity* “the greater the concept extent the smaller the concept intent”, which becomes visible by this equivalence, is winning conciseness and fruitfullness by the contextual mathematization of concept which lastingly moves the mathematical thinking. The reciprocity can be formulated by the definition of “derivation operators” of a formal context (G, M, I): For $X \subseteq G$ anf $Y \subseteq M$ the *derivation* is defined, respectively, by

$$X^I := \{m \in M | gIm \text{ for all } g \in X\} \text{ and } Y^I := \{g \in G | gIm \text{ for all } m \in Y\};$$

i.e. the derivation X^I is the set of all attributes out of M which all objects have, and the derivation Y^I is the set of all objects out of G which all attributes have. For $A \subseteq G$ and $B \subseteq M$, (A, B) is then obviously a formal concept of (G, M, I) if and only if $A = B^I$ and $B = A^I$. The logical reciprocity now finds its differentiated expression by the following mathematical discovery: For $U, V \subseteq G$ or $U, V \subseteq M$ we obtain:

$$(1)\text{: } U \subseteq V \text{ implies } U^I \supseteq V^I,\ (2)\text{: } U \subseteq U^{II},\ (3)\text{: } U^I = U^{III}.$$

For the task to determine the formal concepts of a formal context (G, M, I), the equation in (3) is basic because it follows from (3) that for $X \subseteq G$ and $Y \subseteq M$, respectively, the pairs (X^{II}, X^I) and (Y^I, Y^{II}) are formal concepts of (G, M, I); in particular, the special case of the *object concepts* $\gamma g := (\{g\}^{II}, \{g\}^I)$ and the *attribute concepts* $\mu m := (\{m\}^I, \{m\}^{II})$ are important. The mathematical potential of the derivation operators which become transparent by the relationships in (1), (2), and (3) cannot be estimated high enough; they represent mathematical connections which in general have been studied and activated multifariously as set-theoretic and logical dualities (also called *Galois conections*).

The set of all concepts of a formal context (G, M, I) forms with the subconcept-superconcept relation a mathematical structure of a complete lattice, which therefore is called the *concept lattice* of (G, M, I). The mathematical structure of a concept lattice can be made effectively accessible to logical thinking by (inscribed) *line diagrams*. The line diagram in Fig. 2 [KW87] represents the concept lattice of the formal concext which is presented by the cross-table in Fig. 1. The little circles of the line diagram represent the formal concepts of the appertaining formal context and the ascending line segment represent the subcontext-superconcept-relation. Hence the little circle in Fig. 2 to which the label "artificial" is assigned represents a subconcept of the concepts with the labels "inland" and "constant"; this indicates that, according to [Kr79], there is the logical relationship in English that each "artificial" water has the attributes "inland" and "constant". In general, the extent and intent of formal concepts can be read from the line diagram as follows: The concept extent consists of all objects the names of which are attached to a circle linked by an ascending sequence of line segments to the circle of the chosen concept. In Fig. 2, for instance, the little circle directly above the circle with the label "artificial" represents a concept the extent of which consists of the words "sea", "lagoon", "tarn", "lake", and "pool" and the intent of the noems "natural", "stagnant", and "constant". From this discussion it follows in particular that the underlying context can be reconstracted from the line diagram, i.e. no data are lost by the construction of the concept lattice and line diagram. Therefore the logical connections of the data represented in the cross-table can completely be reconstructed.

The logical connections which usually demand special interest are the *contextual implication between attributes*. From the line diagram of Fig. 2 one reads for instance that, according to [Kr79], each running water is always also constant and inland. Also of interest are the *classification of objects* by suitable combinations of attributes. The line diagram in Fig. 2 shows that the smallest of such classification consists of six concept extents: {*"plash"*, *"puddle"*}, {*"trickle"*, *"rill"*, *"river"*, *"rivulet"*, *"runnel"*, *"beck"*, *"brook"*, *"burn"*, *"stream"*, *"torrent"*}, {*"canal"*}, {*"tarn"*, *"lake"*, *"pool"*}, {*"meer"*, *"pond"*, *"reservoir"*}, and finally {“*sea*′′, “*lagoon*′′}. Respectively, a sequence of further forms of investigations and activations of logical connections in data contexts are treated in the papers [Wi87] and [Wi00c]. In the contrastive study in [Kr79] the *comparison of German and English semantic fields* with the same noems, respectively, are standing in the foreground. Remarkable is the finding that the first eight noems yield the same kind of concept lattices in German and English, which has the consequence that also the logical implications between the noems are equal. This is different at the classifications of objects, already because the English has considerably more words for waters as the German. This is also the reason for it, that further noems thoroughly result in different concept structures.

Line diagrams of concept lattices inspire again and again to *critics* and *self-correction* on the basis of background knowledge. A reseach project which provided multifarious examples for this was a common project of the Darmstadt research group on Formal Concept Analysis and of the ministry of building constructions and housing projects of the province “Nordrhein-Westfalen” [EKSW0]. The developed exploration system was supposed to support the administrative office with its supervision of building works to consider the legal regulations and technical determinations during the planing, examination, and execution of building projects in the necessary extent. For the exploration system an extensive data context was elaborated, the objects of which are the constructional relevant paragraphs or text-units of the pertinent laws and regulations and the attributes of which, understood as search words, are concerned with the structural components and their demands which are related to the text units. For the exploration system frequent concept lattices from the underlying data context were derived and represented by line diagrams to be able to use them as conceptual searching structures.

Already during the system development, line diagrams have multifariously fulfilled to make logical connections transparent. In this way the line diagrams have always again qualified the building experts to find mistakes in the extensive data contexts which has contributed to a conciderable improvement of the data quality. An instructtive case of *criticism and self-correction* has happened by means of the line diagram presented in Fig. 3, that makes available information to the theme “function rooms in a hospital”: For testing the readability of such diagrams, a secretary was included into the meeting in the ministry. The secretary became much surprised that §51 of the “BauONW” (“Bauordnung Nordrhein-Westfalen”), which demands expansions necessary for handicapped people, was only attached to the circle with the label “toilet” (in the version of the diagram “function rooms in a hospital” at that time); she could not understand why the wash- and bathrooms do not have to meet requirements for handicapped people too. Even the experts became surprised when they checked again §51 and saw that only toilets are mentioned in connections with handicapped people. Only after a comprehensive discussion the experts came to the conclusion that, by superior aspects of law, §51 should apply also to wash- and bathrooms. Finally, by similar reasons, the consulting rooms and the residential rooms (bedrooms) were also included so that, in the underlying cross table, three

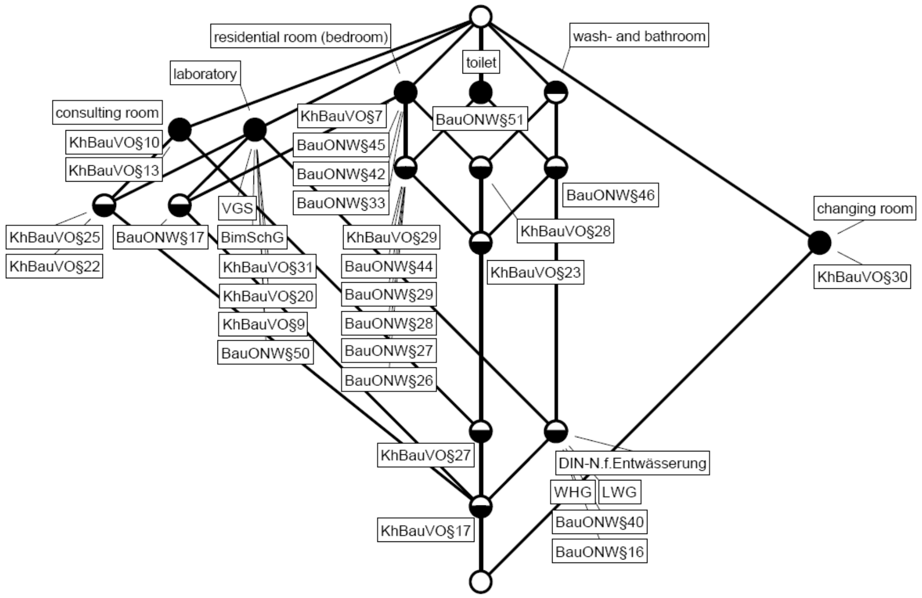


Fig. 3. Query structure "functional rooms in a hospital" of a TOSCANA information system about laws and regulations concerning building construction

more crosses were added in the row headed by "BauONW§51" so that, in the line diagram of Fig. 3, the label "BauONW§51" moved down to the circle with the label "KhBauVO§27".

The *Contextual Judgment Logic*, developped since 1996, builds up on the Contextual Concept Logic because judgments are formed by concepts. An elaborated informing Judgment Logic is already present since more than thirty years by Sowa's *Theory of Conceptual Graphs* [So84] which is founded on the logic of existential graphs of Charles Sanders Peirce and the logic of semantic networks of artificial intelligence. Conceptual graphs, as the simple example in Fig. 4 [So92], are understood as logical abstractions of linguistic expressions; they represend semantic judgments, i.e. valid statements. The conceptual graph in Fig. 4 represents the sentence "John is going to Boston by bus" as *judgment-logical structure*, where the sentence is logically further differentiated with assistance of background knowledge: John is identified as an instance of the concept "Person" and Boston as an instance of the concept "City"; furthermore, three valences of the concept "Go" are specified by the semantic relation "agent", "destination", and "instrument". The presented conceptual graph can be described in detail as follows: "There is some going which has as agent the person John, as destination the city Boston and as instrument some bus." The further *"logical differentiation*" discloses not only the background of a language, but supports further treatments as the translation to other languages, the preparation for document management etc. How a technical text can be judgment-logically processed has been, for instance, made clear in ([MSW99]; p.426) by

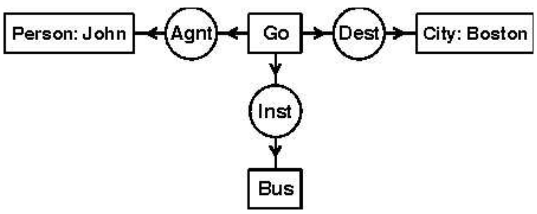


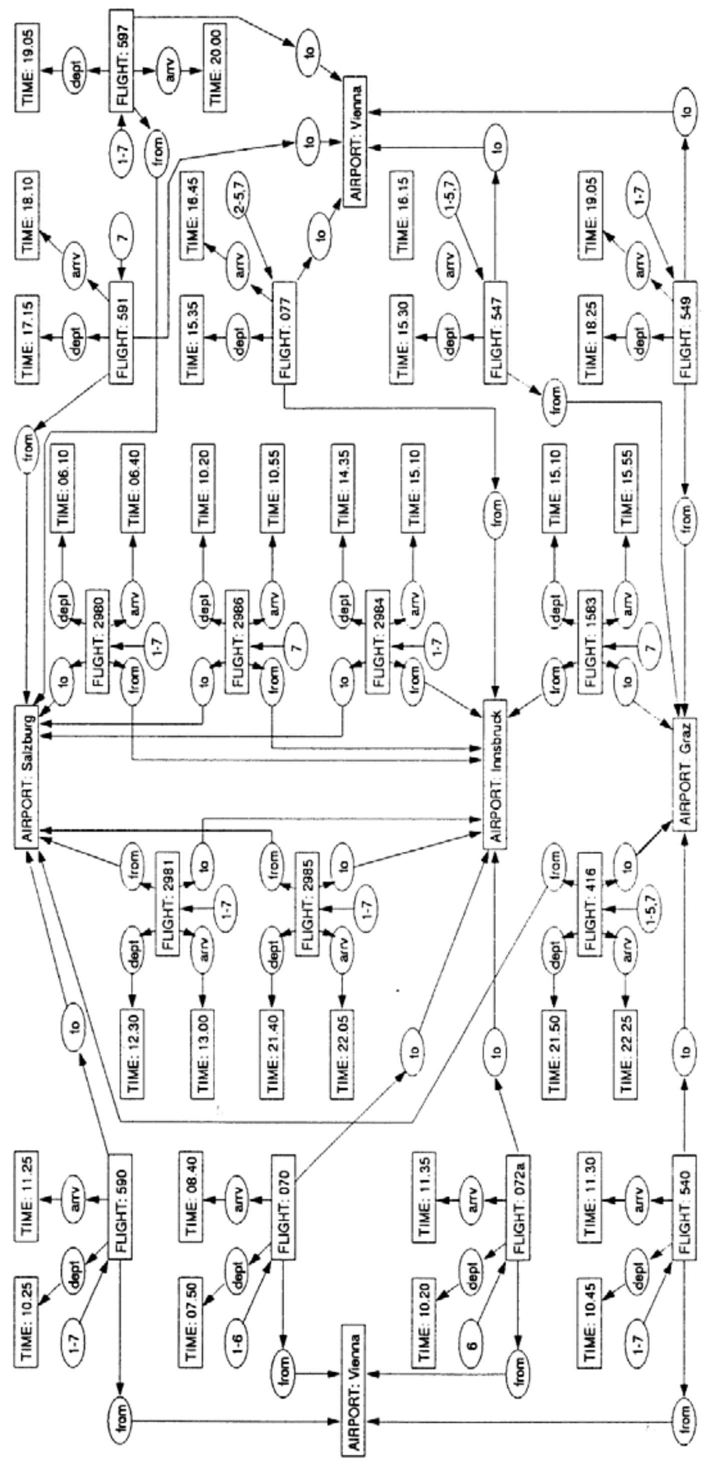
Fig. 4. A simple conceptual graph

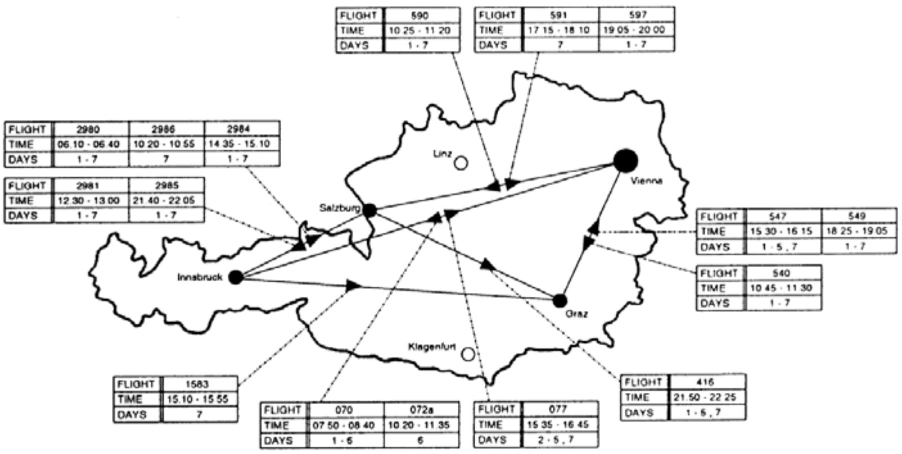
the conceptual graph which represents the instruction for decalcifying a coffee machine.

The Contextual Judgment Logic adapts the Sowa graphs by taking the concepts and relations of the conceptual graphs as formal concepts of already given contexts; with that the conceptual graphs become mathematical structures for which the modified naming *"concept graphs"* has been chosen to distinguish between the mathematical and the logical (s. [Wi97], [Wi00b]). Within the Contextual Judgment Logic, judgments are represented by concept graphs which are therefore also named *formal judgments*. With the abstraction of judgments to mathematical structures, mathematical theories and methods can be activated for the judgment logic in a wide range. The promising method which, up to now, has been stimulated and made possible the mathematization of the judgment logic is the derivation of concept graphs out of relational data basis, which are mathematized contextual-logically in a suitable manner (s. [PW99]); i.e. expressed slogan-like: with this method, relational data bases can be "made speaking". Fig. 5 gives an insight into an informatoly application of this method: The upper diagram shows a concept graph derived out of a flight data base represented as a Sowa graph, which shows the possible flights of a weekend traveller from Vienna to Salzburg, Innsbruck, Graz, and back to Vienna; the lower diagram is a user-friendly representation of the same graph which uses more background knowledge of the traveller (s. [EGSW0], [Wi00c]).

The *Contextual Conclusion Logic* has already concept-logical and judgment-logical precursors by the Contextual Attribute Logic [GW99] and the Contextual Logic of Relations [Wi00d] which adapted the Peircean algebraic logic as reconstructed in [Bu91]. The *Contextual Conclusion Logic* however obtains its full foundation by the interplay of an elaborated syntax and semantics for concept graphs for which Susanne Prediger made available in [Pr98] convincing conceptions and results. Certainly, the interplay of mathematical structural thinking, the diagrammatic conclusions of Charles Sanders Peirce, and the logical thinking in general have to be understood even more deeper, in particular in relationship to the concrete intercourse with such a culture of thinking.

The close connection between the logical thinking and the mathematical thinking, which becomes visible in the frame of contextual logic, makes possible multifariously an effective support of logical thinking through mathematical thinking which can also be extended to *rich mathematical structures*. For example, the contextual-logical concept theory was already extended to an


AIRPORT: Vienna
AIRPORT: Salzburg
AIRPORT: Innsbruck
AIRPORT: Graz
FLIGHT: 597
FLIGHT: 591
FLIGHT: 077
FLIGHT: 547
FLIGHT: 549
FLIGHT: 2980
FLIGHT: 2986
FLIGHT: 2984
FLIGHT: 1583
FLIGHT: 2981
FLIGHT: 2985
FLIGHT: 416
FLIGHT: 590
FLIGHT: 070
FLIGHT: 072a
FLIGHT: 540
TIME: 19.05
TIME: 20.00
TIME: 17.15
TIME: 18.10
TIME: 15.35
TIME: 16.45
TIME: 15.30
TIME: 16.15
TIME: 18.25
TIME: 19.05
TIME: 06.10
TIME: 06.40
TIME: 10.20
TIME: 10.55
TIME: 14.35
TIME: 15.10
TIME: 15.10
TIME: 15.55
TIME: 12.30
TIME: 13.00
TIME: 21.40
TIME: 22.05
TIME: 21.50
TIME: 22.25
TIME: 10.25
TIME: 11.25
TIME: 07.50
TIME: 08.40
TIME: 10.20
TIME: 11.35
TIME: 10.45
TIME: 11.30
dept
arrv
from
to
1-7
7
2-5,7
1-5,7
1-6
6



FLIGHT	590
TIME	10.25 - 11.20
DAYS	1 - 7

FLIGHT	591	597
TIME	17.15 - 18.10	19.05 - 20.00
DAYS	7	1 - 7

FLIGHT	2980	2986	2984
TIME	06.10 - 06.40	10.20 - 10.55	14.35 - 15.10
DAYS	1 - 7	7	1 - 7

FLIGHT	2981	2985
TIME	12.30 - 13.00	21.40 - 22.05
DAYS	1 - 7	1 - 7

FLIGHT	547	549
TIME	15.30 - 16.15	18.25 - 19.05
DAYS	1 - 5 , 7	1 - 7

FLIGHT	540
TIME	10.45 - 11.30
DAYS	1 - 7

FLIGHT	1583
TIME	15.10 - 15.55
DAYS	7

FLIGHT	070	072a
TIME	07.50 - 08.40	10.20 - 11.35
DAYS	1 - 6	6

FLIGHT	077
TIME	15.35 - 16.45
DAYS	2 - 5 , 7

FLIGHT	416
TIME	21.50 - 22.25
DAYS	1 - 5 , 7

Fig. 5. Two representations of the same concept graph concerning flight connections

algebraic concept analysis [Vo94], to a contextual topology [Ha92], [Sa01], and to a relational concept analysis [Ps98]. These extensions correspond with the three structure types of the Bourbaki architecture of mathematics [Bo74] which Jean Piaget has recognized in the close connection to the structures discovered by him in the thinking of young children. In ([Pi73]; p.34f.), Piaget writes about the discussion with Jean Dieudonn, the founder of the Bourbaki-Group: "... to our great surprise we both found out that there exists a very direct connection between the three mathematical structures and the three structures of the operational thinking of children." Even if the activation of this relationship in the "New Math"-movement was exceeded one-sidedly, an appropriate presentation of that relationship would enrich the learning of mathematics and would contribute to an efficient connection from the mathematical to the logical thinking.

Naturally, the logical thinking with its reference to reality has also inversely a lasting effect on the development of mathematics by stimulating always further differentiations of mathematical thinking. Deputizing for the large manifold of such differentiations, it shall finally be mentioned a *new view on mathematics* which has been produced during the elaboration of the contextual logic under the influence of the triadic doctrine of categories of Charles Sanders Peirce: Different real world connections have shown that the elementary connection "an object has an attribute" should be specified in which way, under which conditions, by which arguments, on which purpose, in which situation such a connection is valid. This caused to extend the set structure of a formal context to a triadic structure [LW95], the appertaining concept structure of which was mathematically characterized by a so-called "trilattice" [Bi98]. The thereby possible mathematical *theory of triadic concepts* could already be applied within the Contextual Judgment Logic to receive mathematically the modal character of judgments [Wi98], [Pr98], [DW00]. To what extend the triadic view can generally be made productive for mathematics, this has to be explored by further research.

References

[Ap75] Apel, K.-O.: Der Denkweg von Charles Sanders Peirce. Suhrkamp-Taschenbuch Wissenschaft 141, Frankfurt (1975)

[AN85] Arnauld, A., Nicole, P.: La Logique ou l'Art de penser. Sixième Édition revue et de nouveau augmentée. A Amsterdam, Chez Abraham Wolfgang (1685)

[Bi98] Biedermann, K.: A foundation of a theory of trilattices. Dissertation, TU Darmstadt 1998. Shaker Verlag, Aachen (1998)

[Bo58] Boole, G.: An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities. Macmillan, Basingstoke (1954); Reprented by Dover Publ., New York (1958)

[Bo74] Bourbaki, N.: Die Architektur der Mathematik. In: Otte, M. (Hrsg.) Mathematiker über Mathematik, pp. 140–159. Springer, Heidelberg (1974)

[Bu91] Burch, R.W.: A Peircean reduction thesis. Texas Tech University Press

[DW00] Dau, F., Wille, R.: On the modal understanding of triadic contexts. In: Decker, R., Gaul, W. (eds.) Classification and information processing at the turn of the millennium, pp. 83–94. Springer, Heidelberg (2000)

[Du95] Duden - Das große Wörterbuch der deutschen Sprache in 8 Bänden. 2. Aufl. Dudenverlag, Mannheim 1993–1995, S(2145) (1993–1995)

[EGSW0] Eklund, P., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of TOSCANA. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 453–467. Springer, Heidelberg (2000)

[EKSW0] Eschenfelder, D., Kollewe, W., Skorsky, M., Wille, R.: Ein Erkundungssystem zum Baurecht: Methoden der Entwicklung eines TOSCANA-Systems. In: Stumme, G., Wille, R. (Hrsg.) Begriffliche Wissensverarbeitung: Methoden und Anwendungen, pp. 254–272. Springer, Heidelberg (2000)

[Fa73] Fahrenbach, H.: Mensch. In: Krings, H., Baumgartner, H.M., Wild, C. (Hrsg.) Handbuch philosophischer Grundbegriffe, pp. 888–913. Kösel-Verlag, München (1973)

[GW99] Ganter, B., Wille, R.: Formale concept analysis: mathematical foundation. Springer, Heidelberg (1999)

[Ha92] Hartung, G.: Topological representation of lattices: an approach via formal concept analysis. Dissertation, TU Darmstadt 1992 (1992)

[Hd62] Heidegger, M.: Die Frage nach dem Ding. Niemeyer, Tübingen (1962)

[He86] Hegel, G.W.F.: Phänomenologie des Geistes. Suhrkamp Taschenbuch Wissenschaft 603, Frankfurt (1986)

[He08] Hereth Correia, J.: Towards mathematical foundations of concept-oriented databases: relation graphs and contextual logic. Dissertation, TU Dresden 2008 (2008)

[Hr97] Hersh, R.: What is mathematics, really? Oxford University Press, New York (1997)

[Ka83a] Kant, I.: Werke. Bd. 2: Vorkritische Schriften bis 1768. Wissenschaftliche Buchgesellschaft, Darmstadt (1983)

[Ka83b] Kant, I.: Werke. Bd. 5: Schriften zur Metaphysik und Logik. Wissenschaftliche Buchgesellschaft, Darmstadt (1983)

[Kr79] Karcher, G.L.: Konstrastive Untersuchung von Wortfeldern im Deutschen und Englischen. Peter Lang, Frankfurt (1979)

[KW87] Kipke, U., Wille, R.: Formale Begriffsanalyse erläutert an einem Wortfeld. LDV-Forum 5, 31–36 (1987)

[Ki84] Kitcher, P.: The nature of mathematical knowledge. Oxford University Press, Oxford (1984)

[LW95] Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: Ellis, G., Rich, W., Levinson, R., Sowa, J.F. (eds.) ICCS 1995. LNCS (LNAI), vol. 954, pp. 32–43. Springer, Heidelberg (1995)

[LP00] Lengnink, K., Prediger, S.: Mathematisches Denken in der Linearen Algebra. Zentralblatt für Didaktik 32, 111–122 (2000)

[MSW99] Mineau, G., Stumme, G., Wille, R.: Conceptual structures represented by conceptual graphs and formal concept analysis. In: Tepfenhart, W., Cyre, W. (eds.) ICCS 1999. LNCS (LNAI), vol. 1640, pp. 423–441. Springer, Heidelberg (1999)

[Pe91] Peirce, C.S.: Schriften zum Pragmatismus und Pragmatizismus. Herausgegeben von K.-O. Apel. Suhrkamp-Taschenbuch Wissenschaft 945, Frankfurt (1991)

[Pe92] Peirce, C.S.: Reasoning and the logic of things. In: Ketner, K.L., Putnam, H. (eds.), Havard University Press, Cambridge (1992)

[Pe98] Peirce, C.S.: The Essential Peirce (1893-1913). Edited by the Peirce Edition Project, vol. 2. Indiana University Press, Bloomington (1998)

[Pi73] Piaget, J.: Einführung in die genetische Erkenntnistheorie. Suhrkamp-Taschenbuch Wissenschaft 6, Frankfurt (1973)

[Pr98] Prediger, S.: Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik. Dissertation, TU Darmstadt 1998. Shaker Verlag, Aachen (1998)

[Pr00] Prediger, S.: Mathematische Logik in der Wissensverarbeitung: Historisch-philosophische Gründe für eine Kontextuelle Logik. Mathematische Semesterberichte 47, 165–191 (2000)

[PW99] Prediger, S., Wille, R.: The lattice of concept graphs of a relationally scaled context. In: Tepfenhart, W.M. (ed.) ICCS 1999. LNCS (LNAI), vol. 1640, pp. 401–414. Springer, Heidelberg (1999)

[Ps98] Priss, U.: Relational concept analysis: semantic structures in dictionaries and lexical databases. Dissertation, TU Darmstadt 1996. Shaker Verlag, Aachen (1998)

[Sa01] Sacarea, C.: Towards a theory of contextual topology. Dissertation, TU Darmstadt 2000. Verlag Shaker, Aachen (2001)

[Se01] Seiler, T.B.: Begreifen und Verstehen. Ein Buch über Begriffe und Bedeutungen. Verlag Allgemeine Wissenschaft, Mühltal (2001)

[So84] Sowa, J.F.: Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading (1984)

[So92] Sowa, J.F.: Conceptual graph summary. In: Nagle, T.E., Nagle, J.A., Gerholz, L.L., Eklund, P.W. (eds.) Conceptual structures: current research and practice, pp. 3–51. Ellis Horwood (1992)

[SW00] Stumme, G., Wille, R. (Hrsg.): Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Springer, Heidelberg (2000)

[Vo94] Vogt, F.: Bialgebraic contexts. Dissertation, TU Darmstadt 1994. Verlag Shaker, Aachen (1994)

[Wa73] Wagner, H.: Begriff. In: Krings, H., Baumgartner, H.M., Wild, C. (Hrsg.) Handbuch philosophischer Grundbegriffe, 1st edn., pp. 191–209. Kösel-Verlag, München (1973)

[Wi87] Wille, R.: Bedeutungen von Begriffsverbänden. In: Ganter, B., Wille, R., Wolff, K.E. (Hrsg.) Beiträge zur Begriffsanalyse. B.I.-Wissenschaftsverlag, Mannheim (1987)

[Wi95] Wille, R.: Begriffsdenken: Von der griechischen Philosophie bis zur Künstlichen Intelligenz heute. Diltheykastanie, Ludwig-Georgs-Gymnasium, Darmstadt, 77–109 (1995)

[Wi96] Wille, R.: Restructuring mathematical logic: an approach based an Peirce's pragmatism. In: Ursini, A., Agliano, P. (eds.) Logic and Algebra, vol. 196, pp. 267–281. Marcel Dekker, New York

[Wi97] Wille, R.: Conceptual graphs and formal concept analysis. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS (LNAI), vol. 1257, pp. 290–303. Springer, Heidelberg (1997)

[Wi98] Wille, R.: Triadic concept graphs. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS (LNAI), vol. 1453, pp. 194–208. Springer, Heidelberg (1998)

[Wi00a] Wille, R.: Bildung und Mathematik. Mathematische Semesterberichte 47, 11–25 (2000)

[Wi00b] Wille, R.: Contextual logic summary. In: Stumme, G. (ed.) Working with conceptual structures: Contributions to ICCS 2000, pp. 265–276. Shaker-Verlag, Aachen (2000)

[Wi00c] Wille, R.: Begriffliche Wissensverarbeitung: Theorie und Praxis. Informatik Spektrum 23, 357–369 (2000)

[Wi00d] Wille, R.: Lecture notes on conceptual logic of relations. FB4 Preprint, TU Darmstadt (2000)

[Wi01] Wille, R.: Lebenswelt und Mathematik. In: Hauskeller, C., Liebert, W., Ludwig, H. (Hrsg.) Wissenschaft verantworten: soziale und ethische Orientierung in der Technischen Zivilisation, pp. 51–68. Agenda-Verlag, Münster (2001)

Default Conceptual Graph Rules: Preliminary Results for an Agronomy Application

Jean-François Baget[1,2], Madalina Croitoru[2], Jérôme Fortin[2], and Rallou Thomopoulos[3,2]

[1] INRIA Sophia Antipolis, 2004 Route des Lucioles 06902 Sophia Antipolis, France
baget@lirmm.fr
[2] LIRMM (CNRS & Université Montpellier II), F–34392 Montpellier Cedex 5, France
{croitoru,fortin}@lirmm.fr
[3] INRA, UMR1208, F–34060 Montpellier Cedex 1, France
rallou.thomopoulos@supagro.inra.fr

Abstract. In this paper, we extend Simple Conceptual Graphs with Reiter's default rules. The motivation for this extension came from the type of reasonings involved in an agronomy application, namely the simulation of food processing. Our contribution is many fold: first, the expressivity of this new language corresponds to our modeling purposes. Second, we provide an effective characterization of sound and complete reasonings in this language. Third, we identify a decidable subclass of Reiter's default logics. Last we identify our language as a superset of $\mathcal{SREC}^-$, and provide the lacking semantics for the latter language.

1 Introduction and Motivation

The modeling need that motivated this paper came from an agronomy application: the simulation of food processing (more specifically the pasta drying process). In this application, successive unit operations involved in the drying process have different impacts on product qualities. These impacts can be positive or negative, non monotonically depending on the considered quality and the concerned unit operation. 46 kinds of qualities have been identified for pasta products, moreover these qualities can themselves be subdivided into taxonomies of components (e.g. sub-families of vitamins) that behave differently, hence the need to account for particular cases concerning specific subfamilies [1]. The choice of Conceptual Graphs (CGs) as a modeling language stems from the intuitiveness of their graphical representation as well as the possibility to use their structure for optimization purposes.

Generally, languages of the CG family have a semantics that can be expressed in first-order logic (FOL). The non-monotonic features of the knowledge we want to represent for this application calls for an extension of these languages. The extension we consider here is based upon Reiter's default logics. This formalism has been designed to extend FOL with knowledge like "something is true unless we believe something else". On top of the traditional constructs of FOL-based CG languages (support, facts, rules, constraints), a knowledge base of our new language also consists of default CG rules inspired by Reiter's defaults. These default CG rules fully generalize CG rules and not only the type hierarchy as done in [2]. In this paper we formally present this language and illustrate it with motivating examples from our application.

We define the classical notions of conceptual graphs, rules and constraints in SECT. 2. SECT. 3 is devoted to introducing default reasoning. We recall in SECT. 3.2 Reiter's default formalism, and we introduce the syntax and semantics of default CG rules in SECT. 3.3. Finally, theoretical results are presented in SECT. 4: in SECT. 4.1 we introduce the derivation tree for default conceptual graphs and present a subclass of default CG rules for which this tree is finite. In SECT. 4.2 we use this tree for sound and complete reasonings. Finally, in SECT. 4.3 we relate our new language with the $\mathcal{SREC}^-$ of [3]. The paper concludes with future directions of work.

2 Conceptual Graphs, Rules and Constraints

In this section, we recall essential results about conceptual graphs (CGs). The different languages presented here are described in more detail in [3]. They all form a subset of first-order logics (FOL) since all objects introduced (support, graphs, rules or constraints) have a FOL semantics obtained via the transformation Φ ($\Phi(X)$ is thus the *logical interpretation* of the object X). In all these languages, we will consider a *knowledge base* (KB) containing different objects (*i.e.* support, graphs, rules and/or constraints).

Definition 1 (Semantics of a knowledge base). *The logical interpretation $\Phi(\mathcal{K})$ of a KB $\mathcal{K}$ is the conjunction of the logical interpretations $\Phi(X)$ of the objects X it contains. A KB $\mathcal{K}$ is said* satisfiable *if the FOL formula $\Phi(\mathcal{K})$ is satisfiable. If Q is a simple CG, we say that Q can be* deduced *from $\mathcal{K}$, and note $\mathcal{K} \models Q$, if $\Phi(Q)$ is a semantic consequence of $\Phi(\mathcal{K})$.*

We are interested here in the $\mathcal{X}$-SATISFIABILITY and $\mathcal{X}$-DEDUCTION problems, where $\mathcal{X}$ is a language of the CG family defined by the kinds of objects allowed in a KB. A $\mathcal{SG}$ KB contains only a support and a (set of) simple CG(s). A $\mathcal{SR}$ KB is the union of a $\mathcal{SG}$ KB with a set of rules, and a $\mathcal{SGC}^-$ KB the union of a $\mathcal{SG}$ KB with a set of negative constraint. A $\mathcal{SRC}^-$ KB is the union of a $\mathcal{SR}$ KB and of a $\mathcal{SGC}^-$ KB. The three following subsections successively present the syntax of the different objects that can compose a KB, their logical interpretations, and recall essential results allowing to compute $\mathcal{X}$-SATISFIABILITY and $\mathcal{X}$-DEDUCTION in these different languages.

2.1 Simple Conceptual Graphs: The $\mathcal{SG}$ Language

Syntax: Support and Simple CGs (SGs) A KB of the $\mathcal{SG}$ language is composed solely of a *support* (encoding a type hierarchy) and of a set of *simple CGs* (that represent entities and relationships between them).

Definition 2 (Support). *A* support *is a tuple $\mathcal{S} = (T_C, T_R^1, \cdots, T_R^k, M)$ whose elements are partially ordered, pairwise disjoint sets, respectively of* concepts types, relation types *of arity $1, \cdots, k$, and of* markers. *All partial orders are noted $\leq$. Markers are partitioned into an infinite set M_G of* (named) generic *markers (if m is generic, then $\forall m' \in M, m' \leq m$) and a set M_I of* individual *markers (that are pairwise non-comparable).*

Definition 3 (Simple conceptual graph). *A* simple conceptual graph *(or SG) defined on a support $\mathcal{S} = (T_C, T_R^1, \cdots, T_R^k, M)$ is a tuple $G = (C, R, \gamma, \epsilon)$ where C and R are disjoint finite sets, respectively of* concepts *and* relations. *The mapping $\gamma : R \rightarrow C^+$ associates to each relation a tuple of concepts $\gamma(r) = (c_1, \cdots, c_p)$ called the* arguments *of the relation. We note $\gamma_i(r) = c_i$ its i^{th} argument. The mapping $\epsilon : C \cup R \rightarrow (2^{T_C} \times M) \cup_{1 \leq i \leq k} T_R^i$* labels *each concept and relation. If c is a concept of C, then $\epsilon(c) = (t, m) \in 2^{T_c} \times M$ (t is called the* type *of c, m is called its* marker*). If $m \in M_G$ then c is called* generic, *otherwise it is called* individual. *If r is a relation of R, then its type $\epsilon(r) \in \cup_i T_R^i$ must have the correct arity, i.e. $|\gamma(r)| = j \Leftrightarrow \epsilon(r) \in T_R^j$.*

A simple CG representation of information about "a pasta product that contains peroxidase and is undergoing a late end-of-cycle temperature drying" is presented in FIG. 1.

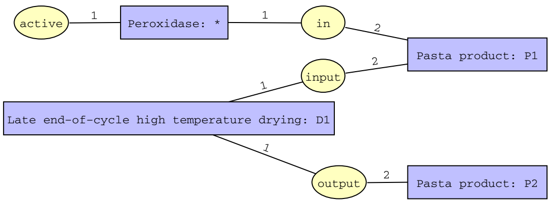


Fig. 1. The SG representing "a pasta product that contains peroxidase and is undergoing a late end-of-cycle temperature drying"

Note that (as required by our modeling, and as defined, for example, in [4]), the type of a concept can be a set of concept types of T_C (called a *conjunctive type*). A concept c can thus be an instance of many distinct types (*e.g.*, $\epsilon(c) = \{$`Protein`,`Enzyme`$\}$).

FOL Semantics. Supports and SGs can be translated into first order logic (FOL) to obtain a precise semantics for our syntactic objects. We consider concept types (resp. relation types of arity i) as *predicate names* of arity 1 (resp. of arity i), generic markers as *variables* and individual markers as *constants*. If t and t' are two predicate names of arity i, and $t \leq t'$, then their logical interpretation is the FOL formula $\phi(t, t') = \forall x_1 \cdots \forall x_i (t(x_1, \cdots, x_i) \rightarrow t'(x_1, \cdots, x_i))$. The logical interpretation of a support $\mathcal{S}$ is the formula $\Phi(\mathcal{S})$ obtained from the conjunction of the formulas $\phi(t, t')$, for all t, t' such that t' covers[1] t in $\mathcal{S}$.

[1] We say that t' covers t if $t \leq t'$ and there is no other t'' (apart from t and t') such that $t \leq t'' \leq t'$.

Let $G = (C, R, \gamma, \epsilon)$ be a SG. If c is a concept, we note $\phi(c)$ the conjunction of atoms $t(m)$, where $t \in \mathit{type}(c)$ and m is the marker of c. If r is a relation, we note $\phi(r) = t(m_1, \cdots, m_i)$ where $t = \epsilon(r)$ and, $\forall 1 \leq j \leq i$, m_j is the marker of $\gamma_j(r)$. We note $\phi(G) = \bigwedge_{c \in C} \phi(c) \wedge \bigwedge_{r \in R} \phi(r)$. Then the logical interpretation $\Phi(G)$ of G is the existential closure of $\phi(G)$.

Theorem 1. *Every $\mathcal{SG}$ KB $\mathcal{K} = (\mathcal{S}, G)$ is satisfiable.*

Reasonings. Though $\mathcal{SG}$-SATISFIABILITY is a trivial problem, $\mathcal{SG}$-DEDUCTION is an important problem that has been studied both inside and outside the CG community. Classically, $\mathcal{SG}$-DEDUCTION can be computed using a kind of graph homomorphism known as *projection*. It maps concepts having the same marker of the query Q to concepts of the SG G in the KB, while preserving the existence of relations and possibly decreasing labels, as allowed by the order relation defined in the support $\mathcal{S}$. We note $G \preceq_{\mathcal{S}} Q$ when there exists such a mapping. For more details on projection/homomorphism, as defined for simple CGs with conjunctive types, the reader can refer to [5]. Projection is a sound operation w.r.t. our FOL semantics, but to be complete, the SG G must be put into its *normal form* $\mathit{nf}(G)$ (a semantically equivalent SG whose concepts have all different markers). Then:

Theorem 2 (Soundness and completeness). *Let $\mathcal{K} = (\mathcal{S}, G)$ be a $\mathcal{SG}$ KB, and Q be a SG. Then $\mathcal{K} \models Q \Leftrightarrow \mathit{nf}(G) \preceq_{\mathcal{S}} Q$.*

As HOMOMORPHISM, $\mathcal{SG}$-DEDUCTION is thus a NP-complete problem. By imposing some restrictions to the SG Q (*e.g.*, when Q admits a bound hypertree decomposition, see [5,6]), the problem becomes polynomial.

2.2 Adding Rules: The $\mathcal{SG}$ Language

Syntax. A $\mathcal{SR}$ KB is obtained by adding CG rules of form (hypothesis, conclusion) to a $\mathcal{SG}$ KB.

Definition 4 (CG rule). *A CG rule over a support $\mathcal{S}$ is a tuple $R = (H, C)$ where H and C are two SGs. $H = \mathit{hyp}(R)$ is called the hypothesis of the rule and $C = \mathit{conc}(R)$ its conclusion.*

FOL Semantics. The transformation Φ defined in SECT. 2.1 and can be extended to take CG rules into account. If $R = (H, C)$ is a CG rule, we note $\phi_H(C) = \exists x_1 \cdots \exists x_p \phi(C)$ where $x_1, \cdots, x_p$ are all variables of $\phi(C)$ that do not also appear in $\phi(H)$. Then we note $\phi(R) = \phi(H) \rightarrow \phi_H(C)$ and the logical interpretation $\Phi(R)$ of the rule R is the universal closure of $\phi(R)$. The interpretation $\Phi(\mathcal{R})$ of a set of CG rules $\mathcal{R}$ is the conjunction of the interpretations $\Phi(R)$, for all rules $R \in \mathcal{R}$.

Theorem 3. *Every $\mathcal{SR}$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R})$ is satisfiable.*

Reasonings. Rules increase the complexity of our reasonings: $\mathcal{SR}$-DEDUCTION is *semi-decidable* (if $\mathcal{K} \models Q$, then a sound and complete algorithm will stop, but no sound and complete algorithm is ensured to stop otherwise). [7] provides a sound and complete *forward chaining algorithm.* It relies upon the application of a CG rule $R = (H, C)$ to a SG G. R is said applicable if there is a projection, say π from H to G. In that case, the application of R to G following π produces a SG $\alpha(G, R, \pi)$ obtained by juxtaposing G and C, then for each concept of c whose marker also appears in a concept c' of H, by fusioning c with $\pi(c')$. Note that other generic markers of C have to be renamed (a safe substitution), and that $\alpha(G, R, \pi)$ must be put into its normal form.

If $\mathcal{R}$ is a set of rules, we note $\alpha_{\mathcal{S}}(G, \mathcal{R})$ the SG obtained by applying all rules in $\mathcal{R}$ to G following all the projections of their hypothesis. Then we define inductively $\alpha^i_{\mathcal{S}}$ by $\alpha^0_{\mathcal{S}}(G, \mathcal{R}) = nf(G)$ and $\forall 1 \leq i, \alpha^i_{\mathcal{S}}(G, \mathcal{R}) = \alpha_{\mathcal{S}}(\alpha^{i-1}_{\mathcal{S}}(G, \mathcal{R}), \mathcal{R})$.

Theorem 4 (Soundness and completeness). *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R})$ be a $\mathcal{SR}$ KB, and Q be a SG. Then $\mathcal{K} \models Q \Leftrightarrow \exists i, \alpha^i_{\mathcal{S}}(G, \mathcal{R}) \preceq_{\mathcal{S}} Q$.*

If $\mathcal{K} = (\mathcal{S}, G, \mathcal{R})$, we note $\mathcal{K}^* = lim_{i \to \infty} \alpha^i_{\mathcal{S}}(G, \mathcal{R})$. Note that, in general, $\mathcal{K}^*$ is an *infinite* SG. To ensure that forward chaining stops, even when $\mathcal{K} \not\models Q$, [3] relies upon the notion of *finite expansion sets* of rules, ensuring that $\mathcal{K}^*$ is finite.

Definition 5 (Finite expansion set (f.e.s.)). *Let $\mathcal{S}$ be a support, and $\mathcal{R}$ be a set of rules. We say that $(\mathcal{S}, \mathcal{R})$ is a* finite expansion set (or f.e.s.) *iff for every $\mathcal{SR}$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R})$, $\mathcal{K}^*$ is finite.*

If $(\mathcal{S}, \mathcal{R})$ is a f.e.s., forward chaining is ensured to stop (when $\alpha^i_{\mathcal{S}}(G, \mathcal{R}) \equiv \alpha^{i+1}_{\mathcal{S}}(G, \mathcal{R}) \equiv \mathcal{K}^*$). Finding large subsets of rules that have the finite expansion property is thus an important task. [3] provides two examples of f.e.s.: *disconnected rules (d.r.)*, that share no generic marker in the hypothesis and the conclusion, and *range restricted rules (r.r.)*, where all generic markers of the conclusion are already in the hypothesis. In both cases, $\mathcal{SR}$-DEDUCTION is NP-complete. [8] introduced the notion of *rules dependencies* (R_2 depends upon R_1 when an application of R_1 can trigger an new application of R_2). When the graph encoding these dependencies has no circuit, then the set of rules is a f.e.s. More importantly, when all strongly connected components of this graph are f.e.s., then we also obtain a f.e.s.

2.3 Adding Negative Constraints: The Languages $\mathcal{SGC}^-$ and $\mathcal{SRC}^-$

Theorems 1 and 3 point out that all SGs and SG rules are satisfiable. However considering that every KB is satisfiable is not always realistic in practice. For example, in our application, we do not want an enzyme to be active and inhibited at the same time. Though various mechanisms have been proposed to introduce the notion of insatisfiability to conceptual graphs, we focus here on *negative constraints.*

Syntax. By enriching a KB of the $\mathcal{SG}$ (respectively $\mathcal{SR}$) language with *negative constraints*, we obtain a KB of the $\mathcal{SGC}^-$ (resp. $\mathcal{SRC}^-$) language. A negative constraint encodes that some knowledge must not be found in a graph.

Definition 6 (Negative constraint). *A* negative constraint*, defined over a support $\mathcal{S}$, is noted $N = \neg G$, where G is a SG over $\mathcal{S}$.*

FOL Semantics. The notation $\neg H$ stems from the semantic of negative constraints since the interpretation of $N = \neg G$ is defined by $\Phi(N) = \neg\Phi(G)$. If $\mathcal{N}$ is a set of negative constraints, then $\Phi(\mathcal{N})$ is the conjunction of all $\Phi(N)$, for $N \in \mathcal{N}$.

It is then possible with negative constraints to express cases of insatisfiability. For example, a KB containing the SG G of FIG. 1 as well as the negative constraint represented by the same FIG. is unsatisfiable.

Theorem 5 (Insatisfiability). *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N})$ be a $\mathcal{SRC}^-$ KB (it is a $\mathcal{SGC}^-$ when $\mathcal{R} = \emptyset$). Then $\mathcal{K}$ is unsatisfiable iff there exists $N = \neg C$ such that $(\mathcal{S}, G, \mathcal{R}) \models C$.*

$\mathcal{SGC}^-$-SATISFIABILITY is thus co-NP complete, and $\mathcal{SRC}^-$-SATISFIABILITY is truly undecidable (though $\mathcal{SRC}^-$-UNSATISFIABILITY is semi-decidable). The polynomial subclasses of SECT. 2.1 apply for $\mathcal{SGC}^-$-SATISFIABILITY while the decidable subclasses of SECT. 2.2 apply for $\mathcal{SRC}^-$-SATISFIABILITY.

Reasonings. Since negative constraints encode negative information and the query encodes positive formulae, negative constraints play no more role in reasonings when the KB is satisfiable.

Theorem 6 (Deduction). *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N})$ be a $\mathcal{SRC}^-$ KB, and Q be a SG. Then $\mathcal{K} \models Q$ iff $\mathcal{K}$ is unsatisfiable or $(\mathcal{S}, G, \mathcal{R}) \models Q$.*

$\mathcal{SGC}^-$-DEDUCTION is thus a NP-complete problem and $\mathcal{SRC}^-$-DEDUCTION is semi-decidable. As previously discussed, particular subclasses of SECT. 2.1 and SECT. 2.2 still apply.

3 Adding Defaults to Conceptual Graphs

3.1 The Need for Default Reasonings

In FIG 2 an agronomy application example is depicted: “if a pasta product undergoes a quick drying, then it is subject to cracking unless the drying is accompanied by vapor-injection”. To deal with such non monotonic knowledge in the following we propose to introduce default reasoning in the CG model, in order to express rules that will be applied in the default case, i.e. unless they are a source of insatisfiability.

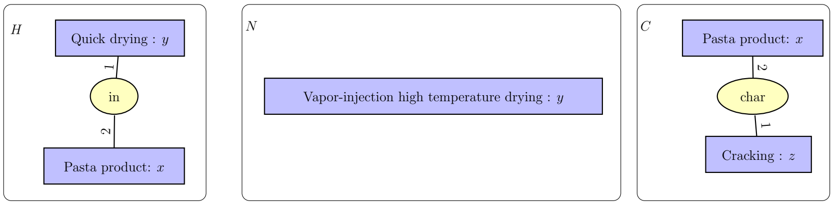


Fig. 2. An example of a default CG rule

3.2 Reiter's Default Logics

In this section we recall some basic definitions of Reiter's default logics [9,10]

Definition 7 (Reiter's default logic). *A Reiter's* default theory *is a pair (Δ, W) where W is a set of FOL formulae and Δ is a set of* defaults *of form $\delta = \frac{\alpha(\overrightarrow{x}):\beta_1(\overrightarrow{x}),\cdots,\beta_n(\overrightarrow{x})}{\gamma(\overrightarrow{x})}$, $n \geq 0$, where $\overrightarrow{x} = (x_1, \cdots, x_k)$ is a set of variables, $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ and $\gamma(\overrightarrow{x})$ are FOL formulae for which each free variable is in $\overrightarrow{x}$.*

The intuitive meaning of a default δ is "For all individuals $(x_1, \cdots, x_k)$, if $\alpha(\overrightarrow{x})$ is believed and each of $\beta_1(\overrightarrow{x}), \cdots, \beta_n(\overrightarrow{x})$ can be consistently believed, then one is allowed to believe $\gamma(\overrightarrow{x})$". $\alpha(\overrightarrow{x})$ is called the *prerequisite*, $\beta_i(\overrightarrow{x})$ are called the *justifications* and $\gamma(\overrightarrow{x})$ is called the *consequent*. A default is said to be *closed* if $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ and $\gamma(\overrightarrow{x})$ are all closed FOL formulae. A default theory (Δ, W) is said to be *closed* if all its defaults are closed. In this case we can omit the $\overrightarrow{x}$ notation.

Intuitively, an *extension* of a default theory (Δ, W) is a set of formulae that can be obtained from (Δ, W) while being consistently believed. More formally, an extension E of (Δ, W) is a minimal deductively closed set of formulae containing W such that for any $\frac{\alpha:\beta}{\gamma} \in \Delta$, if $a \in E$ and $\neg\beta \notin E$, then $\gamma \in E$.

The following theorem provides an equivalent characterization of extensions that we use here as a formal definition.

Theorem 7 (Extension). *Let (Δ, W) be a closed default theory and E be a set of closed FOL formulae. We inductively define $E_0 = W$ and for all $i \geq 0$, $E_{i+1} = Th(E_i) \cup \{\gamma \mid \frac{\alpha:\beta_1\cdots,\beta_n}{\gamma} \in \Delta, \alpha \in E_i$ and $\neg\beta_1, \cdots, \neg\beta_n \notin E\}$*[2]*.*

Then E is an extension of (Δ, W) iff $E = \cup_{i=0}^{\infty} E_i$.

Note that extensions are only defined here for closed theories. In practice open defaults are transformed into the sets of their ground instances over the Herbrand universe.

Note also that this characterization is not effective for computational purposes since both E_i and $E = \cup_{i=0}^{\infty} E_i$ are required for computing E_{i+1}.

Some closed default theories can have no extension. It is for example the case of the default theory $(\Delta, W) = (\{\frac{\top:\beta}{\neg\beta}\}, \emptyset)$. However, normal default theories are ensured to have extensions.

[2] We note $Th(E_i)$ the deductive closure of E_i.

Definition 8 (Normal defaults). *A default is said* normal *if its consequent is semantically equivalent to the conjunction of its justifications. Defaults of form* $\delta = \frac{\alpha(\overrightarrow{x}):\beta(\overrightarrow{x})}{\beta(\overrightarrow{x})}$ *are normal.*

The meaning of a normal default is *if α is true and it is consistent to deduce β, then deduce β.*

Theorem 8. *Every closed normal default theory has an extension.*

Let us see a classical example of a default theory. Suppose that we want to model the knowledge that, in general, *birds fly, penguins are birds*, and *penguins do not fly.* Finally we add a penguin called *Tweety* in our knowledge base. This knowledge can be model by the following default theory :

$$(\Delta, W) = \Big(\Big\{\frac{p(x) : \neg f(x)}{\neg f(x)}, \frac{p(x) : b(x)}{b(x)}, \frac{b(x) : f(x)}{f(x)}\Big\}, \{p(Tweety)\}\Big)$$

where $b(x)$ means that the individual x is a bird, $f(x)$ means that x flies, and $p(x)$ means x is a penguin. Note that the knowledge *penguins are birds* have no known exception, and so a rule $\forall x, p(x) \rightarrow b(x)$ can be added to W instead of the default rule $\frac{p(x):b(x)}{b(x)}$ in D. This default theory can lead to 2 different extensions, which are $E_1 = Th(\{p(Tweety), b(Tweety), \neg f(Tweety)\})$ and $E_2 = Th(\{p(Tweety), b(Tweety), f(Tweety)\})$.

Some problems that must be addressed in Reiter's default logics are the following:

- EXTENSION: Given a default theory (Δ, W), does it have an extension?
- SKEPTICAL DEDUCTION: Given a default theory (Δ, W) and a formula Q, does Q belong to all extensions of (Δ, W)? In this case we note $(\Delta, W) \models_S Q$.
- CREDULOUS DEDUCTION: Given a default theory (Δ, W) and a formula Q, does Q belong to an extension of (Δ, W)? In this case we note $(\Delta, W) \models_C Q$?

In the previous example (Δ, W) admits two extensions. Both $f(Tweety)$ and $\neg f(Tweety)$ can be credulously deduced, but neither can be skeptically deduced.

Note that even when restricting these problems to closed normal default theories, the expressive power of FOL makes them undecidable.

3.3 Introducing Default CG Rules: The $\mathcal{SRDC}^-$ Language

Syntax. A KB of the $\mathcal{SRDC}^-$ language is obtained from a $\mathcal{SRC}^-$ KB enriched with default CG rules inspired by Reiter's defaults.

Definition 9 (Default CG rule). *A* default CG rule *over a support $\mathcal{S}$ is a tuple $D = (H, N_1, \cdots, N_n, C)$, with $n \geq 0$, H and C are SG's, and the N_i are negative constraints over S. As in Reiter's defaults, we call H the prerequisite, N_i the justifications, and C the consequent.*

Intuitively, such default means that "if H is believed, the negative constraints (justifications) are each satisfied, and *it is consistent to believe C*, then it is allowed to believe C".

Default Semantics. The interpretation of $\mathcal{SRDC}^-$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ is a default theory $\Upsilon(\mathcal{K}) = (\Upsilon(\mathcal{D}), \Phi((\mathcal{S}, G, \mathcal{R}, \mathcal{N})))$ where Φ is the FOL interpretation of the KB as defined in SECT. 2, and $\Upsilon(\mathcal{D}) = \{\Upsilon(D), D \in \mathcal{D}\}$. The mapping Υ translates each default CG rule D into a default in Reiter's sense $\Upsilon(D)$ called the *default interpretation* of D.

Let $D = (H, N_1, \cdots, N_n, C)$ be a default CG rule, where $N_i = \neg G_i$. Its default interpretation $\Upsilon(D)$ is built as follows:

– Let $\overrightarrow{h}$ be the variables occurring in $\phi(H)$, $\overrightarrow{f}$ the variables occurring both in $\phi(C)$ and in $\phi(H)$, and $\overrightarrow{c}$ the variables occurring in $\phi(C)$ and not in $\phi(H)$.
– For $\zeta \in \{\phi(C), \phi(G_1), \cdots, \phi(G_n)\}$, the formula $sk(\zeta)$ is obtained by replacing for $c_i \in \overrightarrow{c}$, each occurrence of c_i by the functional term $f_i^D(\overrightarrow{f})$ in ζ.
– For $\xi \in \{sk(G_1), \cdots, sk(G_n)\}$, $sk^*(\xi)$ is obtained by existentially quantifying all variables of ξ that are not in $\overrightarrow{h}$. Finally:

$$\Upsilon(D) = \frac{\phi(H) : sk(C), \neg sk^*(G_1), \cdots, \neg sk^*(G_n)}{sk(C)}$$

Let us illustrate this by the transformation of the default CG rule $D = (H, N, C)$ of FIG. 2. In the next equation, $QD(x)$ means that product x undergoes a *quick drying*, $P(x)$ signifies that x is a *pasta product*, $C(x)$ signifies the *Cracking* property of pasta and $VIHTD(y)$ specifies a *vapor-injection high temperature drying* y. While at the representation level the formula below has the same meaning as FIG. 2, the authors consider that FIG. 2 conveys its meaning in a more intuitive manner.

$$\Upsilon(D) = \frac{QD(y) \wedge in(y,x) \wedge P(x) : P(x) \wedge char(f_1^D(y,x),x) \wedge C(f_1^D(y,x)), VIHTD(y)}{P(x) \wedge char(f_1^D(y,x),x) \wedge C(f_1^D(y,x))}$$

The problems defined in Reiter's default logics are easily recast in $\mathcal{SRDC}^-$:

– $\mathcal{SRDC}^-$-EXTENSION: Given a $\mathcal{SRDC}^-$ KB $\mathcal{K}$, does $\Upsilon(\mathcal{K})$ have an extension?
– $\mathcal{SRDC}^-$-SKEPTICAL DEDUCTION: Given a $\mathcal{SRDC}^-$ KB $\mathcal{K}$ and a SG Q, does $\Upsilon(\mathcal{K}) \models_S \Phi(Q)$? In this case we note $\mathcal{K} \models_S Q$.
– $\mathcal{SRDC}^-$-CREDULOUS DEDUCTION: Given a $\mathcal{SRDC}^-$ KB $\mathcal{K}$ and a SG Q, does $\Upsilon(\mathcal{K}) \models_C \Phi(Q)$? In this case we note $\mathcal{K} \models_C Q$.

Modeling Choices. Two features of our chosen semantics might seem surprising to the reader. First, the presence of $sk(C)$ as an added justification. This is due to the fact that we need to be able to represent normal defaults in our language (if a default rule $D = (H, C)$ has no negative constraint then $\Upsilon(D)$ is a normal default). Second, we have introduced functional terms in the interpretation of a default. This is due to the fact that the default interpretation is composed of many formulae and functional terms are the only way to link up the variables of these formulae.

4 Reasoning in $\mathcal{SRDC}^-$

4.1 The Defaults Derivation Tree (d.d.t.)

The defaults derivation tree (d.d.t.) of a $\mathcal{SRDC}^-$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ is a rooted, labeled and possibly infinite tree $ddt(\mathcal{K})$ used as a tool to compute extensions. To define this tree, we need new objects generalizing negative constraints, that we call attached constraints.

Attached Constraints. Let G be a SG. A constraint attached to G is a pair (A, μ) where A is a SG and μ is a partial mapping from the concepts of A to the concepts of G. We say that G *violates* (A, μ) iff there exists a projection π from A into G such that π extends μ. Otherwise G *satisfies* (A, μ).

Note that attached constraints generalize negative constraints (the latter occurs in the case of $\mu = \emptyset$). A $\mathcal{SR}$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R})$ violates a constraint (A, μ) attached to G iff there exists $i \geq 0$ and a projection π from A to $\alpha^i_{\mathcal{S}}(G, \mathcal{R})$ such that π extends μ. It satisfies (A, μ) otherwise. $\mathcal{K}$ violates a set $\mathcal{A}$ of constraints attached to G iff it violates one $(A, \mu) \in \mathcal{A}$. It satisfies $\mathcal{A}$ otherwise. The complexity of computing satisfiability with attached constraints remains the same as for negative constraints.

Note that if (A, μ) is a constraint attached to G and G' is a SG containing G (such as a SG obtained by applying rules on G), then we can consider (A, μ) as a constraint attached to G'. In the same way, many algorithms rely on finding a smaller equivalent SG G' by fusioning concepts of the SG G. Then, for every (A, μ) attached to G, we attach a constraint (A, μ') to G' such that if there is a concept c in A such that $\mu(c)$ has been fusioned into c' in G', then $\mu'(c) = c'$, and $\mu'(c) = c$ otherwise.

Vertices of the d.d.t. The d.d.t. intuitively represents a kind of derivation tree. Each node v is labeled by $\lambda(v) = (G_v, \mathcal{A}_v)$. G_v represents a state of knowledge derived from the initial KB and $\mathcal{A}_v$ represents the suppositions that we made to derive G_v. For example, consider the application of the default CG rule represented in FIG. 2 on a pasta product A which undergoes a quick drying Q. To conclude that A is subject to cracking, we need to suppose (and remember in $\mathcal{A}_v$ for further derivation) that Q is not a *vapor-injection high temperature drying.* To remember this ensures that no further derivation can conclude that Q was a *vapor-injection high temperature drying.*

A vertex v of $ddt(\mathcal{K})$ is labeled by $\lambda(v) = (G_v, \mathcal{A}_v)$ where G_v is a SG and $\mathcal{A}_v$ is a set of constraints attached to G_v. The root r of $ddt(K)$ is labeled by $\lambda(r) = (G, \emptyset)$. A vertex v of $ddt(K)$ is *satisfiable* iff $(\mathcal{S}, G_v, \mathcal{R}, \mathcal{N})$ is satisfiable and $(\mathcal{S}, G_v, \mathcal{R})$ satisfies $\mathcal{A}_v$.

If v is satisfiable, then for each $D = (H, N_1, \cdots, N_n, C)$, for each projection π into some $G' = \alpha^i_{\mathcal{S}}(G_v, \mathcal{R})$, if π is not "blocked" v admits a child $v' = \delta(v, \pi)$.

Let us consider the SG $G'' = \alpha(G', (H, C), \pi)$. For each justification N_k, we build the constraint (N_k, μ_k) attached to G'' where μ_k is defined as follows: if c is a concept of N_k whose generic marker appears in a node c' of H then $\mu_k(c) = \pi(c')$. Otherwise, if this marker appears in a node c' of C then $\mu_k(c)$ is a concept obtained from a copy of c' in G''. We note $\mathcal{A}'_v = \mathcal{A}_v \cup \{A_k\}_{1 \leq k \leq n}$. Finally π is *blocked* iff there exists $j \geq i$ such that $\alpha(\alpha^j_{\mathcal{S}}(G_v, \mathcal{R}), (H, C), \pi)$ violates $\mathcal{A}'_v$. If π is not blocked, then $\lambda(v') = (G'', \mathcal{A}'_v)$.

Building Finite d.d.t. Given the expressive power of SG rules, $ddt(\mathcal{K})$ is an infinite tree: it can have an infinite depth and each vertex can have an infinite number of children. To be able to finitely build d.d.t., let us now extend the notion of finite expansion sets (SECT. 2.2).

Definition 10 (Finite expansion property). *If $D = (H, N_1, \cdots, N_n, C)$ is a default CG rule, we note $fol(D) = (H, C)$ its associated SG rule. If $\mathcal{D}$ is a set of default CG rules, we note $fol(\mathcal{D}) = \{fol(D)\}_{D \in \mathcal{D}}$. Then a $\mathcal{SRDC}^-$ KB $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ is said to have the* finite expansion property *iff $\mathcal{R} \cup fol(\mathcal{D})$ is a finite expansion set.*

If $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ has a finite expansion property, then for every vertex v of $ddt(\mathcal{K})$, with $\lambda(v) = (G_v, \mathcal{A}_v)$, G_v is a subgraph of the finite SG $(\mathcal{S}, G, \mathcal{R} \cup fol(\mathcal{D}))^*$. Then, since the graph G_v labeling each vertex v is bigger than the graph labeling is parent, the depth of $ddt\mathcal{K}$ is finite. And since $(\mathcal{S}, G_v, \mathcal{R})^*$ is finite, there is a finite number of projections of the defaults in it, so the number of children of v is finite, and its satisfiability can be computed in finite time. It follows that:

Theorem 9. *If a $\mathcal{SRDC}^-$ KB $\mathcal{K}$ has the finite expansion property then $ddt(\mathcal{K})$ can be computed in finite time.*

4.2 Sound and Complete Reasoning w.r.t. Υ

Let us now show that the d.d.t. can be used for sound and complete reasonings in $\mathcal{SRDC}^-$.

Theorem 10. *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ be a $\mathcal{SRDC}^-$ KB, and Q be a SG. Then either $(\mathcal{S}, G, \mathcal{R}, \mathcal{N})$ is unsatisfiable or the following assertions are equivalent:*

i There exists an extension E of $\Upsilon(\mathcal{K})$ such that $E \models \Phi(Q)$.
ii There exists a satisfiable leaf v of $ddt(\mathcal{K})$ with $\lambda(v) = (G_v, \mathcal{A}_v)$ such that $(\mathcal{S}, G_v, \mathcal{R})$ modelsQ.

Due to space requirements the proof of this theorem is omitted in this paper.
It follows that:

Theorem 11 (Soundness and completeness). *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{N}, \mathcal{D})$ be a $\mathcal{SRDC}^-$ KB, and Q be a SG. Then $\mathcal{K} \models_S Q$ (resp. $\mathcal{K} \models_C Q$) iff either $ddt(\mathcal{K})$ has a unique unsatisfiable vertex, or, for all satisfiable leaves, (resp. there exists one satisfiable leaf) v in $ddt(\mathcal{K})$ with $\lambda v = (G_v, \mathcal{A}_v)$, $(\mathcal{S}, G_v, \mathcal{R}) \models Q$.*

This latter theorem provides us with an effective characterization of the deduction problems in $\mathcal{SRDC}^-$. Thanks to THM. 9, this characterization also provides a halting algorithm when $\mathcal{K}$ has the finite expansion property.

4.3 Relationship with $\mathcal{SREC}$

The Language $\mathcal{SREC}$. [3] presents a family of CG languages. The most expressive one in this language hierarchy is the language $\mathcal{SREC}$. In this language a KB $\mathcal{K}$ is composed of a support $\mathcal{S}$, a SG G, a set $\mathcal{R}$ of *inference rules* (that behave exactly as CG rules), a set $\mathcal{E}$ of *evolution rules* of form (H, C), and a set $\mathcal{C}$ of constraints. By restricting constraints to the negative constraints presented here we obtain the language $\mathcal{SREC}^-$.

Reasoning in $\mathcal{SREC}^-$. Reasonings in $\mathcal{SREC}^-$ rely upon building a "tree of possible worlds", akin to the d.d.t. presented in this paper. Since $\mathcal{SREC}^-$ does not dynamically generate constraints, a possible world is only labeled by a SG. Children of a possible world are generated as if we considered each evolution rule as a default rule without justification. Finally, an answer to a SG Q can be found in any possible world, not only in the leaves as done in $\mathcal{SRDC}^-$. However, default rules translating evolution rules are normal, and thus any possible world is an ancestor or an extension (THM. 8). Therefore, if an answer to Q can be found in a possible world v, the same answer can be found in all leaves/extensions having v as an ancestor.

Default Semantics for $\mathcal{SREC}^-$. By comparing the reasonings in $\mathcal{SREC}^-$ and $\mathcal{SRDC}^-$ we obtain an interesting equivalence result that provides the formally lacking semantics of $\mathcal{SREC}^-$. Let us consider the bijection τ from $\mathcal{SREC}^-$ KBs to $\mathcal{SRDC}^-$ KBs that transforms each evolution rule into a default CG rule without justification (*i.e.* a normal default CG rule).

Theorem 12. *Let $\mathcal{K} = (\mathcal{S}, G, \mathcal{R}, \mathcal{E}, \mathcal{N})$ be a $\mathcal{SREC}^-$ KB, and Q be a SG. Then $(\mathcal{S}, G, \mathcal{R}, \tau(\mathcal{E}), \mathcal{N}) \models_C Q$ iff $(\mathcal{S}, G, \mathcal{R}, \mathcal{N})$ is unsatisfiable or $\mathcal{K} \models Q$ ($\models$ being the deduction used in $\mathcal{SREC}^-$).*

We can finally provide a logical semantics $\Upsilon_{\mathcal{E}}$ to the $\mathcal{SREC}^-$ language, by defining:

$$\Upsilon_{\mathcal{E}}((\mathcal{S}, G, \mathcal{R}, \mathcal{E}, \mathcal{N})) = \begin{cases} (\bot, \emptyset) \text{ if } (\mathcal{S}, G, \mathcal{R}, \mathcal{N}) \text{ is unsatisfiable} \\ \Upsilon((\mathcal{S}, G, \mathcal{R}, \tau(\mathcal{E}), \mathcal{N})) \text{ otherwise.} \end{cases}$$

$\mathcal{SREC}^-$ is thus the subset of $\mathcal{SRDC}^-$ restricted to normal defaults, and:

Theorem 13. *Deduction in $\mathcal{SREC}^-$ is sound and complete with respect to credulous deduction according to the $\Upsilon_{\mathcal{E}}$ semantics.*

5 Conclusion and Perspectives

In this paper we have formally defined the syntax and semantics of a new language of the $\mathcal{SG}$ family, namely the $\mathcal{SRDC}^-$ language. This extension was necessary in the agronomy application we are involved in, and the semantics of this language are expressed in Reiter's default logics. Since this subset of default logics is built upon a particular subset of FOL, we were able to provide a constructive characterization of Reiter's extensions (THM. [10]). Using the finite expansion sets that form a decidable subclass of $\mathcal{SR}$, we defined a new decidable subclass of Reiter's default logics (THM. [9]). Finally, we showed that the $\mathcal{SREC}^-$ language of [3] is a strict subclass of $\mathcal{SRDC}^-$, and provided a formerly lacking default logic semantics for that language.

Some problems are still to be addressed to be able to encode the knowledge required by our application and to compute deductions in an efficient way:

Functional Relations. For more precise reasonings we need to be able to represent numerical information in a knowledge base and to express functional constraints such as the following rule given by a domain expert: a high temperature for drying has to be above Naples average spring temperature. [11] extends the language $\mathcal{SR}$ to handle such knowledge. This language could provide the foundations for a functional extension of $\mathcal{SRDC}^-$.

Other Decidable Subclasses of CG Rules. The KB obtained from our preliminary modeling has the finite expansion property that ensures finite reasonings. It may be possible that with the introduction of new knowledge this property no longer holds. It would then be essential to investigate other kinds of decidable KBs. An interesting research direction could be to extend other kinds of decidable subclasses of $\mathcal{SR}$ to $\mathcal{SRDC}^-$. Such decidable subclasses could be finite unification sets (that ensure a finite backward chaining rewriting) or a bounded treewidth sets (a strict generalization of f.e.s. ensuring that $\mathcal{K}^*$ has a bounded treewidth)[12].

Reasoning with Preferences. Default logics can be extended to take defaults preferences into account. In this model, one can define an order (partial or total) on the set of defaults. Our default CG rule model provides a natural order on defaults: a default CG rule D_1 should be preferred to a default D_2 if the prerequisite of D_1 is a specialization of the prerequisite of D_2. This is exactly what is intuitively needed in our agronomy scenario. The consequent problems of computing extensions are then transformed into finding the most preferred extensions [10]. Even when defaults are totally ordered, the procedure that chooses the application of the most preferred unblocked default at each vertex of the d.d.t., is not ensured to lead to a preferred extension. Formally defining and finding preferred extensions is left for further work.

References

1. Abécassis, J.: Qualité du blé dur de la semoule et des pâtes alimentaires, 7–11 (1991)
2. Faron, C., Ganascia, J.: Representation of defaults and exceptions in conceptual graphs formalism. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 153–167. Springer, Heidelberg (1997)
3. Baget, J.F., Mugnier, M.L.: Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints. Jour. of Artificial Intelligence Research 16, 425–465 (2002)

4. Baget, J.F., Mugnier, M.L.: The sg family: Extensions of simple conceptual graphs. In: Proc of Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, pp. 205–212. Morgan Kaufmann, San Francisco (2001)
5. Baget, J.F.: Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS, vol. 2746, pp. 229–242. Springer, Heidelberg (2003)
6. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural csp decomposition methods. Artificial Intelligence 124 (2000)
7. Salvat, E., Mugnier, M.L.: Sound and complete forward and backward chainingd of graph rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS, vol. 1115. Springer, Heidelberg (1996)
8. Baget, J.F.: Improving the forward chaining algorithm for conceptual graphs rules. In: Proceedings of the Ninth International Conference (KR 2004), Whistler, Canada, pp. 407–414. AAAI, Menlo Park (2004)
9. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 81–132 (1980)
10. Brewka, G., Eiter, T.: Prioritizing default logic: Abridged report. In: Festschrift on the occasion of Prof.Dr. W. Bibel's 60th birthday. Kluwer, Dordrecht (1999)
11. Baget, J.F.: A datatype extension for simple conceptual graphs and conceptual graphs rules. In: Proc of ICCS 2007: Conceptual Structures: Knowledge Architectures for Smart Applications. LNCS, vol. 4604, pp. 83–96. Springer, Heidelberg (2007)
12. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. IJCAI 2009 (submitted, 2009)

Towards Extraction of Conceptual Structures from Electronic Health Records

Svetla Boytcheva[1] and Galia Angelova[2]

[1] State University of Library Studies and Information Technologies
119 Tzarigradsko Shose Blvd., 1784 Sofia, Bulgaria
svetla.boytcheva@gmail.com
[2] Institute for Parallel Processing, Bulgarian Academy of Sciences
25A Acad. G. Bonchev Str., 1113 Sofia, Bulgaria
galia@lml.bas.bg

Abstract. This paper presents the general framework and the current results of a project that aims to develop a system for knowledge discovery and extraction from the texts of Electronic Health Records in Bulgarian language. The proposed hybrid approach integrates language technologies and conceptual processing. The system generates conceptual graphs encoding the patient case history, which contains templates for the patient's diseases, symptoms and treatments. We describe simple inference in the generated graphs resource bank. Some experiments and their evaluation are presented in the article.

1 Introduction

The first known medical record was developed by Hippocrates in the fifth century B.C. He prescribed two goals when documenting the patient status in natural language: *(i)* A medical record should accurately reflect the course of the disease and *(ii)* A medical record should indicate the probable cause of the disease. These goals are still appropriate today and most of the patient documentation is still kept in natural language as free unstructured text. However, Electronic Health Records (EHR) systems provide additional functionality, such as interactive alerts to clinicians, interactive flow sheets and tailored order sets, automatic calculation of the price of the medical treatment etc., which cannot be supported in the paper-based archives [3].

Today most of the patient's information is available only in textual form. This makes its automatic processing a very difficult task. So to say, medical knowledge is "locked up" in paper documents, files or databases in formats which are not suitable for automated processing [7]. Great efforts have been made to translate this information into certain (semi-)structured representations; the activities always include manual or automatic information extraction from free texts. The main difficulties to structure medical information are the complexity of the domain as a whole, the complex medical language, and the variety of practices for including text descriptions in EHR, which are too specific for different countries and different languages. Many language processing systems, which extract and codify information from EHR in English have been developed. Most generally, they reflect at least two principally-different views to patient-related texts. The first approach is automatic extraction of information concerning patients diagnosis, treatment, manipulations etc. and automatic coding of this information with respect to some established classification schemes, which are provided by financing or statistical institutions. There are large terminology-based nomenclatures such as SNOMED (the Systematized Nomenclature of Medicine) and ICD (the International Classification of Diseases). These collections are unified classification systems, translated into many languages, and support the health management and health statistics. Recent critics to SNOMED explicate some conceptual shortcomings which prevent its application as medical ontology in semantic systems [16]. It remains unclear whether the same kind of terminology-based ontologies can support all principally different systems, which are built on top of medical information extraction. Regarding the extraction precision, the leaders in the fields report successful recognition of the complex medical terminology up to 80-85% even for English [12]. The second kind of prototypes is oriented to medical research and knowledge discovery in medicine. It reflects the AI view to text understanding: to translate the text to internal structured representations, to make inferences, to discover interconnections between facts and concepts which could remain unnoticed otherwise, and to spot previously unknown regularities. Most prototypes of this kind are developed for English. They benefit from the various language resources, available for English, among them large public archives of medical abstracts. Practically there are no significant developments for lesser spoken and minor languages.

Here we present the first steps towards building a system for automatic extraction of medical facts from patient-related texts in Bulgarian language. This research effort is made in a project, supported by the Bulgarian National Science Fund in 2009-2011. We discuss briefly the general ideas behind the project and present the results of its first steps - design and implementation of the Relations Analysis Module and the Conceptual Graphs Generator. The paper is structured as follows. Section 2 overviews some related research and discusses basic language technologies which are used for Information Extraction (IE) in the medical domain. Section 3 describes the general project ideas and sketches a view to the system architecture. Section 4 presents the Relations Analysis Module and the main types of relations which are automatically recognized at present. Section 5 presents the module for generation of logical forms of Conceptual Graphs (CG) using the templates that are filled in by the extracted EHR data. Examples and assessment figures describe the current experiments. Section 6 contains some discussions and the conclusion.

2 Related Work

An overview of the current EHR systems and their functionality can be found in [3]. We focus on the natural language texts in the EHRs assuming that they are available in certain integrated hospital information system. Another overview, comparison and evaluation of the language technologies which extract medical information is given in [4,5]. The white paper [12] presents recent industrial developments in the field.

Several language technologies are used to extract and codify medical information. The most successful applications run for English due to many reasons, among them the simple morphology. The tools for automatic natural language analysis have to lemmatize the text, i.e. to recognize the basic form of every wordform, and to group the separate wordforms into (complex) terms. Afterwards various relations between the sentence phrases have to be found to capture the medical semantics. The main approaches for sentence processing include:

– Partial analysis of sentence segments or local phrases in order to fill in predefined templates and to search for some specific relations and keywords, for instance:
  - using a shallow parser that captures relations between noun phrases (NPs) [9]. The parser extracts relations between all NPs regardless of their type. Then it searches for patterns in the text which are based on English closed-class words - i.e. prepositions (*by, of, in*), negation, conjunctions (*and, or*) and auxiliary or modal verbs. The extracted relations can contain up to five arguments: *relation negation*, *left-hand side*, *connector modifier*, *connector* and *right-hand side*;
  - searching for cause-effect relations within the sentence parse tree. This approach was used in [2] to identify and extract cause-effect information that is explicitly expressed in the Medline medical abstracts. The system is based on tree-like patterns that indicate the presence of certain causal relation in the sentences, and which parts of the sentence represent correspondingly the cause and the effect. The patterns are matched to the syntactic parse trees of the sentences. Thus parts of the parse tree are extracted as NPs referring to the cause or the effect;
  - searching for treatment relations [8] using linguistic patterns which enable the discovery of treatment relations. These patterns are constructed either semi-automatically or manually. Mining for 'association rules' is applied to sample sentences containing both a disease concept and a reference to drugs, to identify frequently occurring word patterns and evaluate whether these patterns could be used to identify treatment relations in sentences;

– Deep parsing of whole sentence in order to construct detailed parsing trees and to process further the sentence semantics;
– Combining several language technologies in a pipe-line environment - e.g. in MedLEE (A Medical Language Extraction and Encoding System [10]).

Specific natural language processing tools are developed to ensure the proper anonymisation of patient records [13] by removal of named entities and replacing them by pseudonyms. Some prototypes deal with the essential problem of negation in the patient records [6], among them there is a module for negation processing in Bulgarian medical texts [1].

3 Project Settings

The suggested system will work on EHRs collected in the Specialized University Hospital for active treatment in endocrinology, in the Clinical Centre of endocrinology and gerontology - Sofia. It will be tuned to the particular domain of diabetes (having in mind its importance). The extraction scenario reflects the specific way of collecting patient information in this hospital. For instance, information about the relatives identity is not systematically supported in the EHRs although there are citations of relevant family diseases. This is due to the fact that the documents are stored in electronic form in the recent years only. Information is kept about the patients' case history and there are links between the different patient's visits to the hospital. From the perspective of the medical experts, the most urgent task is to analyze the hospitalization effects: what happens to a patient when he or she enters the hospital in status A and leaves it in status B, i.e. how the hospital treatment affects the patient state. The prototype will extract the information needed for the automatic generation of a Patient's Chronicle - symptoms and diagnosis, hospital treatments and their results. Based on the ideas of granularity shift using CG type definitions, type contraction and type expansion [18] and applying inference rules, some more general statements regarding the patient status will be produced, which will describe the medications effect given certain patient status. These 'general' graphs will not deal with the single words and concepts in the personal EHRs but will allow for summarizations of the patient information in more general terms which are used by medical professionals when they describe medical knowledge. The whole conceptual archive will support knowledge discovery in medicine. Today we see it as a hypercube of conceptual graphs, corresponding to patients' EHRs and their generalizations. There will be connections between the nodes of different patient graphs which correspond to different visits to the hospital. This very challenging and ambitious task includes much research to be performed in several years. At the present moment we can discuss only the Information Extraction solutions, which concern the words in the particular EHRs, and the generation of conceptual graphs which capture the factology of the individual patient records.

As usual in natural language processing of raw documents, the input medical resources are really problematic - texts with specific abbreviations, numerical values of analyses and clinical test data, medical terminology in Bulgarian in Cyrillic and in Latin (using both the Cyrillic and the Latin alphabets), numerous synonyms of the medical terms, spell-errors with one or two wrong symbols per word, specific language style of the medical professionals and so on. All these obstacles together are not easy to overcome. Another essential problem is the rich temporal structure of the patient descriptions which prevents the application of standard language processing techniques. Fortunately, we rely on stable modules for morphological analysis, very large morphological dictionaries of Bulgarian and well-studied technologies for corrections of spelling errors, which encourages us to approach the automatic processing of raw medical texts as they are stored in the hospital information system. The test corpus contains about 8000 words and most of them are included in the very large lexicons of general Bulgarian vocabulary which supports the morphological analysis. Previous achievements in processing Bulgarian morphology enable chunking of sentence phrases and recognition of the ICD-10 medical terms with precision higher than 50%, which will be improved for the narrow domain of diabetes. A representative corpus of epicrises facilitates the semi-automatic extraction of linguistic templates which support the identification of important medical facts. So in principle the project is equipped with basic background resources and tools for natural language processing. Previous research of the negation in medical patient records in Bulgarian was carried out. It has revealed some typical language constructions, specific features of the negation scope and solutions for their processing [1]. The available components for processing the negation are extended and integrated in the current system. Needless to say, the expectation is that the automatic IE will work with partial success and many details (expressed indirectly or by wrong words) will be missed in the texts. But we believe that IE success of more than 75-80% will enable the development of an useful conceptual archive which will provide a good basis for knowledge discovery and conceptual search. Having in mind this ultimate project objective, we start by a narrow domain where we can progress more quickly with the natural language processing activities.

Comparing the IE tasks for Bulgarian and English, we notice that there are prototypes for English which are very successful in narrow domains - see for instance [15], where the patient smoking status is identified automatically in more than 92% of the cases. This is done by analysis of individual sentences in the patient record. These sentences are selected due to the presence of predefined keywords which occur in the text. Since we have no principal difficulties to tackle the Bulgarian morphology and to perform automatic text lemmatization, we believe that by the project end we can achieve comparable scores for IE success in a narrow domain (where we have to extract more templates, however, possibly from overlapping text fragments).

The design of our IE system is strongly influenced by the EHR structure. The textual part of the EHR in Bulgarian has average length of 2-3 pages and 11 predefined and ordered sections: Personal data, Anamnesis, Status, Examinations, Consultations, Debate, Treatment, Treatment results, Recommendations, Working abilities, and Diagnosis.

The architecture of the IE component is shown at Fig.1. It contains the following modules: Annotation analysis and Chunking; Patients' Data Module; Patients' Relations Module; Templates Selection Module; Post Processing Module; Extractor; Filling Templates Module; Relation Analysis Module; Logical Form Generator / Conceptual Graphs Generator; Template and Relations Refinement.

At the first step, each EHR is split to its 11 sub-topics by the Annotation analysis and Chunking module. The annotation process is based on morphological analysis using a lexicon of 30 000 lexemes. The common Bulgarian vocabulary is expanded by medical terminology and specific words which are met in the available EHR corpus. For each wordform, the module finds its basic form (lexeme)

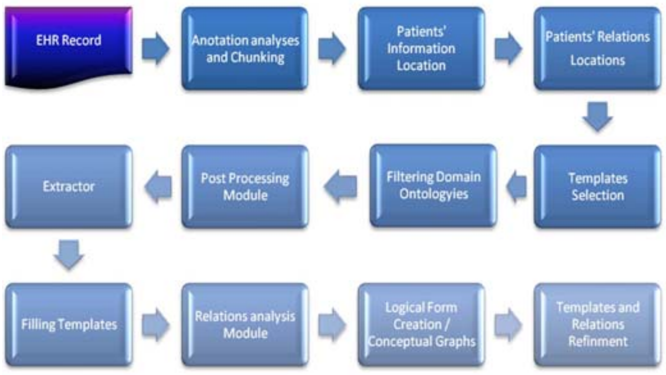

EHR Record
Anotation analyses and Chunking
Patients' Information Location
Patients' Relations Locations
Templates Selection
Filtering Domain Ontologyies
Post Processing Module
Extractor
Filling Templates
Relations analysis Module
Logical Form Creation / Conceptual Graphs
Templates and Relations Refinment

Fig. 1. Pipe-line architecture of modules for information extraction from medical texts

with the associated lexical and grammatical features. Chunks are sequences of words that build complex terms, syntactic groups or sentence phrases. They are recognized by rules defined as regular expressions, which take into account the morphological features of the lexemes and their mutual position. Mostly nominal chunks (NPs) and Prepositional Phrases are recognized at present. The module outputs tagged text.

The Patients' Data Module extracts personal data from the corresponding EHR section (taking into account the pseudonymisation). The Patients' Relations Module creates a Patients' Chronicle graph with nodes, which are slots of templates full of patient information collected in different time periods. The system searches for data about the same patient in the hospital information system. If any is found, the module includes a pointer to the previous records according to the case history. Otherwise the system generates a new graph.

After identifying the topics included in the particular EHR and the possible connections between the patient visits to the hospital, the system needs to decide which templates fit for the representation of the patient data. The possible templates are stored in a resource bank, which contains templates for common information as well as specific templates for the particular medical sub-domain. We have studied representative amounts of EHRs together with the medical experts and have created manually about 50 templates for different diabetes-related facts that need to be tracked in the patient records. A sample template is shown in Fig. 2. To narrow down the search while choosing a suitable template, the system uses domain ontology. It is very helpful that every EHR is split into 11 sections and the medical experts have some well-established practices how to write down the epicrises. The chosen template is included in the graph node for the current patient's EHR. The system maintains four types of ontologies - of symptoms, of diagnosis, of drugs and a shallow ontology of body parts. More details about the templates and their filling are given in Section 4.

The Post Processing Module recognizes important NP and VP chunks using the lexicon and partial grammar rules. Some efforts are needed to determine

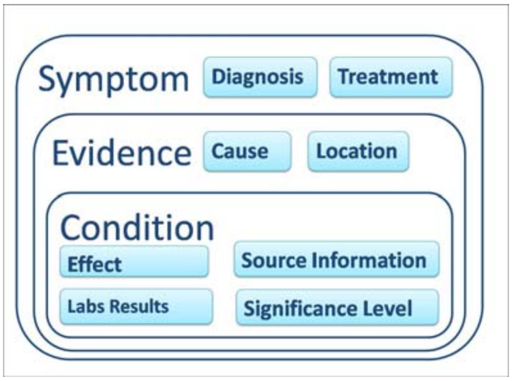


Fig. 2. Patient Status Template

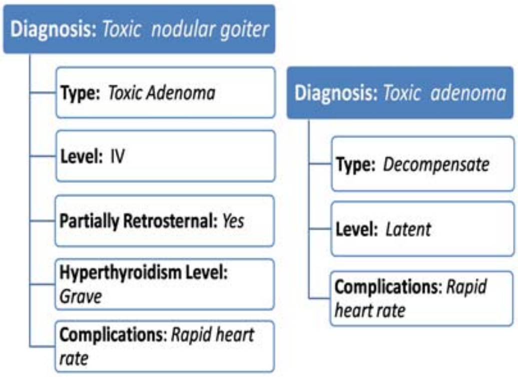

Diagnosis: Toxic nodular goiter
Type: Toxic Adenoma
Level: IV
Partially Retrosternal: Yes
Hyperthyroidism Level: Grave
Complications: Rapid heart rate
Diagnosis: Toxic adenoma
Type: Decompensate
Level: Latent
Complications: Rapid heart rate

Fig. 3. Templates for diagnoses with specific patient values

the VP chunks due to the telegraphic style of the medical reports which rearly contain complete sentences.

The Extractor determines the patient's symptoms, diagnoses and treatment which are reported in the current EHR. The module for Filling Templates tries to fill in the information for each node that is foreseen by the chosen template (Fig. 3). Due to the narrow domain, the expected values of each node can be prelisted. Analyzing the representative corpus of epicrises, we expect to be able to identify all words which are the surface verbalization of the respective medical notions. Some template slots might become empty but others are obligatory.

The Relation Analysis Module identifies three types of relations: *is-a relations*; *Cause-Effect relations*; *Internal relations* between symptoms, diagnosis and treatment in one node of the patient case history and *External Relations* (Fig. 4) between the different nodes of the patient's chronicle graph.

The Logical Form Generator creates CGs represented in first order logic, using the identified relations and the information which is already present in the templates (disregarding the empty slots). The last step is to check again whether

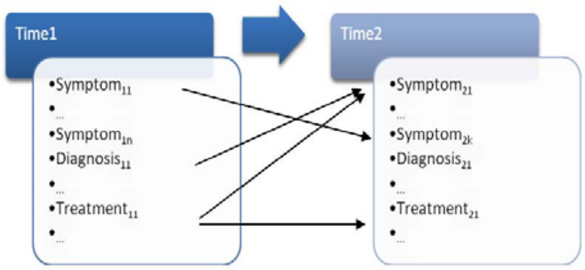


Fig. 4. Relations within the Patient's Chronicle-each node contains slots-templates

some empty slots in the templates can be filled in, given the context of all extracted information and the inference rules.

4 Relation Analysis Module

There are many kinds of relations between the concepts in medical texts but we shall classify the relations in two general classes: *internal* (between the slots of one EHR) and *external* (between the slots of two different EHRs, e.g. two records for one patient in the chronicle graph). At present we are working actively on the internal relations which are extracted with better accuracy.

Recognition of relations is crucial for proper text processing. For instance, the causal relation has significant importance in medicine, which deals with treatments and drugs that can affect or cure a disease. Due to this reason the causal relation is often explicitly indicated in EHRs using linguistic means (i.e. words such as *result, effect, cause* etc.). In some cases the specific phrasal structure helps to identify cue patterns, which work as indicators of the location of desired knowledge [17]. Unfortunately not all the cause-effect relations can be identified by keywords and phrasal patterns. There are more complex relations for which it is necessary to process several discourse sentences and to make inference in order to determine them. Khoo et al. [2] attempted to identify the location of causal relationship description using dependency subtree patterns. One very important task is to find a set of effective cue patterns suitable for the domain and the mining goal. Usually the systems use cue patterns given a priori, presumably devised by domain experts for the prescribed tasks or collected by statistical studies.

In our initial experiment we use about 150 EHRs in Bulgarian for diabetic patients. We have investigated the specific verbalizations of the symptoms-diseases relations in the corpus. The selected cue expressions are ranked by frequency and include the most frequent adjectives, prepositions, adverbs and verbs: "оплаквания от" (complaints) - 73% of its occurrences in the texts signal for symptoms-diseases relations; "данни за" (there exists evidences for) - it appears at least 2 times per each EHR and 100% of the occurrences denote symptoms-diseases relations; "поради" (because of) - 49.2% of the occurrences in EHRs encode a symptom or a disease; "по повод на" (reason for) - 74.6% of its occurrences in EHRs refer to symptoms and diseases, "съобщава" (inform) - 100% of its occurrences signal symptoms-diseases relations (but this cue is tricky because it appears mostly in combination with negation and it is not easy to identify the negation scope). All above mentioned cue phrases mark that the patient has some symptoms and diseases. Shallow ontologies for symptoms, diagnosis and body parts supported the process of cue patterns extraction.

At present the Relation Analysis Module recognizes the following types of cause-effect relations for patient status: *(i)* Between slots in one template (Symptom - Diagnose; Diagnose - Treatment); *(ii)* Between slots in two different templates (Diagnose - Symptom; Treatment - Symptom; Diagnose - Diagnose; Treatment - Treatment; Symptom - Symptom);

We take into account three major types of cue patterns: *(i)* Symptoms and conditions of diseases; *(ii)* Verb expressions representing a relationship, interaction, or action; *(iii)* Symptoms and conditions of diseases - for this type of patterns we use templates with predefined relations and empty slots for the concepts (symptoms, diseases), as well as slots for characteristics representing the condition.

4.1 Example for a Diabetic Patient

Постъпва за 1 път в клиниката, по повод на обща отпадналост, ацетонурия, високи стойности на кръвното налягане, а от няколко дни има повръщане. Заболяването е установено преди 4 години при измерване на кръвна захар, поради обрив на лицето. Въпреки назначеното лечение с манинил и диапрел няма подобрение.

This is the 1st visit of the patient to the clinic with complaints of general weakness, acetonoria, high blood pressure, and sickness since few days. The disease was detected 4 years ago by the high blood sugar measurement, made because of a face rash. Despite of the treatment with Maninil and Diaprel there are no changes for better.

After analysis and chunking of the first sentence we obtain:

Постъпва{Постъпва.V+IPF+I:R3s:E2s:E3s} за{за.PREP} 1{gb} път {път.N+M:s} в{в.PREP} клиниката{клиника.N+F:sd}, по{по.PREP,по.PC} повод {повод.N+M:s}на{на.PREP}обща{общ.A+GR:sf} отпадналост {отпадналост.N+F:s} ацетонория{} високи{висок.A+GR:p} стойности {стойност.N+F:p} на{на.PREP} кръвното {кръвното .A:sn, кръвно.ADV+MNN} налягане {налягам.V+IPF+T:VNs, налягане.N+N:s}, а{a.CONJ}от{от.PREP}няколко{няколко.PRO+IDF:ms}дни {ден.N+M:p:c} има {имам.V+IPF+T:R3s:E2s:E3s} повръщане {повръщам.V+IPF+T:VNs, повръщане.N+N:s}

The Extractor uses a cue pattern (Fig. 5) for each symptom in order to locate in the text as many words and phrases as possible and to send them to the Templates Filling module. There are minimal requirements to fill in the obligatory slots of any template, which is chosen as a relevant one, and in this case they are only:

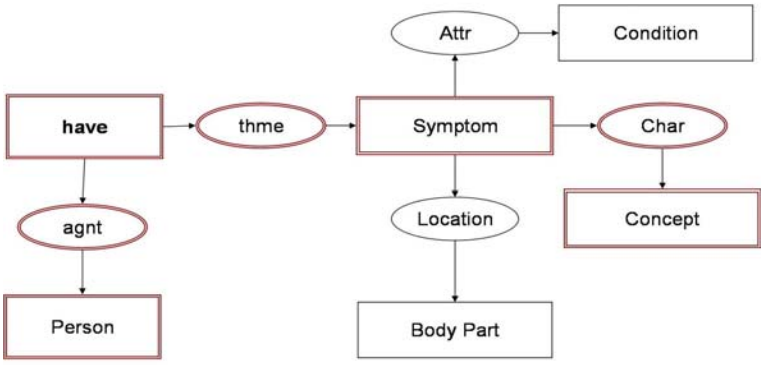


Fig. 5. Cue pattern for a symptom

```
 [HAVE]->(AGNT)->[PERSON]
       ->(THME)->[SYMPTOM]->(CHAR)->[CONCEPT]
```

The remaining slots are optional and they are filled in when additional information is present. The extractor generates the following CGs for the sample sentence 1:

```
[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[SYMPTOM]->(CHAR)->[Weakness]->(ATTR)->[General]

[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[SYMPTOM]->(CHAR)->[Acetonoria]

[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[SYMPTOM]->(CHAR)->[Blood pressure]->(ATTR)->[High]

[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[SYMPTOM]->(CHAR)->[Sickness]-
                                  ->(ATTR)->[since few days ago]
```

In this way the “elementary” cue patterns enable to fill in the templates by the words and phrases which are encountered in the text. Implicit relations are found in this way - e.g. AGNT, THeME, CHAR, ATTR, LOC. They do not correspond to specific words in the EHR. Another type of cue patterns - “Verb expressions representing a relationship, interaction, or action” - support the discovery of relationships between the patients’ template slots as well as relations among several slots in the patient’s chronicle.

Most generally, the relations between the slots in the different sections of one EHR connect each Symptom with the corresponding Diagnosis and each Diagnosis with the corresponding Treatment. To discover such relations we apply statistically collected cue phrases like *effect, results, influence, changes, achievement* etc. For instance, the cause-effect relation representing the result after the treatment "лечение - подобрение" (treatment - improvement) can be found in the sample sentence above with a specific negation: Въпреки назначеното лечение с манинил и диапрел няма подобрение. (*Despite of the treatment with Maninil and Diaprel there are no changes for better.*)

To discover relations between slots in different patient's nodes of the Patients' Chronicle graph we use the rich temporal information in the EHR. The most frequent cue phrases include time, dates, years, months, temporal adverbs like *after*, *before*, etc. The main task is to find the last and the first occurrence of each symptom and diagnose and to connect them to the corresponding Treatment. The example contains typical temporal information which has to be taken into consideration and kept for future monitoring: Заболяването е установено преди 4 години поради измерване на кръвна захар, поради обрив на лицето. (*The disease was detected 4 years ago by the high blood sugar measurement, because of a face rash.*)

The simplest templates are filled in with 92% correctness. However, the more complex cue patterns extract too many irrelevant phrases and the results needs manual human revision. On the other hand the too specific cue patterns generate only few results. The process of relations identification should be iterative in order to improve step by step the IE results, which is our main task at present.

5 Generation of CG in Logical Form

The CG generator collects all the information from the patient's node templates. Here we briefly introduce the CG generation algorithm:

- STEP 1: For each template T_i , construct one graph G_i using maximal join operation for the corresponding common concepts in the template slots.
- STEP 2: For each internal relation R^1, R^2, ..., R^k between slots in the template T_i, add consequently relations to the graph G_i between the corresponding concepts.
- STEP 3: Cluster the set of all p templates $\{T_1,T_2, ..., T_p\}$ in subsets depending on whether the templates are linked by external relations. A given template T_k belongs to a cluster C_m if and only if there exists a template T_n from C_m and an external relation between T_k and T_n. The resulting clusters contain interlinked templates.
- STEP 4: For each external relation between slots in the different templates T_i and T_j, construct a relation between the corresponding concepts in G_i and G_j and generate a new graph G_{ij}'
- STEP 5: Join all new graphs G_{ij}' belonging to one cluster
- STEP 6: Represent all constructed CGs as Logical forms (LF).
- STEP 7: The EHRs archive contains as many LFs as the number of clusters.

This algorithm ensures the production of connected conceptual structures, which encode interlinked information in the EHRs. Several issues have to be mentioned here. The bottom elements in the construction are the system patterns - like the one at Fig. 5 - which shape the extracted words/concepts into conceptual structures. The pattern relations are either present in the text explicitly, or are introduced by default as thematic roles like CHAR, ATTR, AGNT, THeME. Joining the simple patterns at step 1, we obtain one conceptual graph:

```
[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[SYMPTOM]-
                 ->(CHAR)->[Weakness]->(ATTR)->[General]
                 ->(CHAR)->[Acetonoria]
                 ->(CHAR)->[Blood preasure]->(ATTR)->[High]
                 ->(CHAR)->[Sickness]->(ATTR)->[since few days ago]
```

The internal relations between the templates enable joining of conceptual structures which correspond to separate sentences and paragraphs. The relations correspond to referential links between text fragments. We assume that there is no referential ambiguity since the domain language is very specific. Step 2 connects conceptual structures that are linked because of certain linguistic evidences in the EHR text. For instance, if for the sample patient above it is mentioned in the same EHR paragraph that he/she was diagnosed with diabetes, then in the same template we would find the following graph:

```
[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[Disease]->(CHAR)->[Diabetes]
```

After steps 1 and 2 we would obtain the following graph:

```
[HAVE]->(AGNT)->[PERSON]
      ->(THME)->[Disease]->(CHAR)->[Diabetes]
      ->(THME)->[SYMPTOM]-
                 ->(CHAR)->[Weakness]->(ATTR)->[General]
                 ->(CHAR)->[Acetonoria]
                 ->(CHAR)->[Blood preasure]->(ATTR)->[High]
                 ->(CHAR)->[Sickness]->(ATTR)->[since few days ago]
```

Steps 3 and 4 enable to build groups of interlinked templates and graphs in case of external links (reflecting multiple patient records in the hospital information systems). Steps 5 and 6 juxtapose one logical statement to conceptual structure which encodes connected facts.

In general, the join operation may unify different unspecified instances of the same concept type, which is problematic from a knowledge representation perspective. However, studying EHRs we discover that most often each word occurrence refers to one instance - e.g., the blood pressure of the patient. Also, we assume that there are no ambiguous words which denote simultaneously individuals and concept types. In fact the lexical semantics of the nouns in natural language allows them to refer to classes or instances [11] but in a narrow domain all words can be examined in advance and the important semantic distinctions can be marked in the system lexicon. We are currently investigating this issue, in order to provide solid motivation behind our algorithms for construction and unification of patient-related conceptual structures. This on-going work will continue by tests and elaborations of our empirical approach which is tailored to the specific domain.

6 Conclusion and Future Work

The research task presented here aims at the extraction of medical facts from unstructured text in natural language. Since language technologies operate on words and phrases, the atomic extracts are knowledge chunks corresponding to domain-specific templates. The suggested scenario is based on the typical IE settings; for instance all words, which remain outside the templates, are considered as unimportant. Another typical issue for IE is the explication of the implicit text relations via relation names defined in the template slots. At present we evaluate the precision and recall of the initial experiments. Obviously, the implementation of all extraction and modelling components will be iterative with several development cycles.

Conceptual graphs are well-suited to serve as primary patterns because they are adjusted to natural language applications. They also provide a well-defined join operation, assuming that graphs can be "merged" on their common concept instances. Our intuition and the present text examinations show that very often the words occurring in the EHR text point to single instances. This referential particularity is another important issue to be studied in the near future.

Acknowledgement. The research work presented in this paper is partly supported by grant №DO 02-292/December 2008 "Effective search of conceptual information with applications in medical informatics", funded by the Bulgarian National Science Fund in 2009-2011.

References

1. Boytcheva, S., Strupchanska, A., Paskaleva, E., Tcharaktchiev, D.: Some Aspects of Negation Processing in Electronic Health Records. In: Proc. of International Workshop Language and Speech Infrastructure for Information Access in the Balkan Countries, 2005, Borovets, Bulgaria, pp. 1–8 (2005)
2. Khoo, C.S.G., Chan, S., Niu, Y.: Extracting Causal Knowledge from a Medical Database Using Graphical Patterns. In: Proceedings of 38th Annual Meeting of the ACL, Hong Kong (2000)
3. Electronic Health Records Overview, National Institutes of Health and National Center for Research Resources, The MITRE Corporation, Center for Enterprise Modernization McLean, Virginia, USA (2006)
4. Friedman, C., Hripcsak, G.: Natural language processing and its future in medicine. Academic Medicine 74(8), 890–895 (1999)
5. Friedman, C., Hripcsak, G., Shablinsky, I.: An evaluation of natural language processing methodologies. In: Chute, C.G. (ed.) Proceedings 1998 AMIA Annual Symposium, pp. 855–859. Hanley and Belfus, Phil (1998)

6. Gindl, S.: Negation Detection in Automated Medical Applications, a Survey. Vienna University of Technology, Institute of Software Technology & Interactive Systems, Asgaard-TR-2006-1 (October 2006)
7. Hripscak, G., Friedman, C., Alderson, P., DuMouchel, W., Johnson, S., Clayton, P.: Unlocking clinical data from narrative reports: a study of natural language processing. Annals of Internal Medicine 122, 681–688 (1995)
8. Lee, C.H., Khoo, C., Na, J.C.: Automatic identification of treatment relations for medical ontology learning: An exploratory study. In: McIlwaine, I.C. (ed.) Knowledge Organization and the Global Information Society: Proc. of the Eighth International ISKO Conference, pp. 245–250. Ergon Verlag, Germany (2004)
9. Leroy, G., Chen, H., Martinez, J.D.: A Shallow Parser Based on Closed-class Words to Capture Relations in Biomedical Text. Journal of Biomedical Informatics (JBI) 36, 145–158 (2003)
10. MedLEE - A Medical Language Extraction and Encoding System, `http://lucid.cpmc.columbia.edu/medlee/`
11. Miller, G., Hristea, F.: WordNet Nouns: Classes and Instances. Computational Linguistics 32(1), 1–3 (2006)
12. Natural Language Processing in Medical Coding. White paper of Language and Computing (April 2004), `http://www.landcglobal.com`
13. Neubauer, T., Riedl, B.: Improving Patients Privacy with Pseudonymization. In: eHealth Beyond the Horizon - Get IT There, pp. 691–696. IOS Press, Amsterdam (2008)
14. Ruch, P., Baud, R.H., Geissbühler, A.: Using lexical disambiguation and named-entity recognition to improve spelling correction in the electronic patient record. Artificial Intelligence in Medicine 29(1-2), 169–184 (2003)
15. Savova, G., Ogren, P., Duffy, P., Buntrock, J., Chute, C.G.: Mayo Clinic NLP System for Patient Smoking Status Identification. Journal of Americal Medical Information Association 15, 25–28 (2008)
16. Schulz, S., Suntisrivaraporn, B., Baader, F.: SNOMED CT's Problem List: Ontologists and Logicians Therapy Suggestions. In: Proc. of the Medinfo 2007 Congress, Studies in Health Technology and Informatics (SHTI-series) (2007)
17. Shimbo, M., Tamamori, S., Matsumoto, Y.: Finding cue expressions for knowledge extraction from scientific text early results. In: Pacific Knowledge Acquisition Workshop 2004, Auckland, New Zealand (2004)
18. Sowa, J.: Conceptual Processing in Mind and Machines, Reading, MA (1984)

Algorithm Design Using Traversals of the Covering Relation

Andrew Burrow

RMIT University, Melbourne, Australia
andrew.burrow@rmit.edu.au

Abstract. Where posets are used to represent taxonomies, concept lattices, or information ordered databases there is a need to engineer algorithms that search, update, and transform posets. This paper demonstrates an approach to designing such algorithms. It presents a picture of covering relation traversals that characterises these in terms of up-set and down-set expressions involving union, intersection, and difference. It then provides a detailed analysis of three types of covering relation traversal. The approach is demonstrated by describing a suite of derived algorithms. The intention is to express a manner of decomposing mathematical problems into poset traversals, and to provide context to the selection a particular traversal algorithm. This line of work has previously been pursued by [1]. However, the success and influence of Formal Concept Analysis [2] has shifted the emphasis from posets to lattices, and from algorithms that operate on the graph of the partial order to the formal context. This paper contributes a methodology for the renewed investigation of poset algorithms, with the potential to lead to improvements in algorithms such as the online completion to a lattice.

1 Introduction

Given its close relationship to knowledge representation through information orderings and domain theory, order theory is the subject of several computational efforts. Much of the ground work for the approach presented in this paper was laid by [1]. In particular, [1] explicitly considers the covering relation as the basis of the implementation, and is unusual in emphasising the poset over the lattice. However, the community that previously existed around conceptual graphs [3] has merged with that of formal concept analysis [2]. Thus, the emphasis has shifted to concept lattices, as the basis for both the formalism and the implementation. Hence, many order theory algorithms are now published in terms of formal concept analysis. For example, the question of minimal insertion into a lattice received attention in the general case [4], but has since been largely considered in terms of formal concept analysis. See [5] for a survey.

This paper emphasises the role of the underlying covering relation, and explicitly represents it. Furthermore, it makes no presumption about the knowledge of the order, whereas formal concept analysis techniques are predicated on the structure of the formal context. For example, [6] defines a strict lexicographic ordering on subsets that is particularly useful when computing over formal contexts. In doing so we are able to put into practice many standard results of algorithms and data structures based on graph theory, in particular breadth-first and depth-first search. Hence, this paper presents generic poset traversal algorithms parametrised by pruning and visiting procedures, and demonstrates how a broad collection of important algorithms are expressed in terms of these generic algorithms.

1.1 Notation

This paper assumes familiarity with the basic definitions and results of order theory as presented in [7]. At the centre of the enquiry is the partial order relation $\leq$ with the familiar shorthand $x < y$ for $x \leq y$ and $x \neq y$.

Definition 1 (Partially Ordered Set). *A binary relation $\leq$ on a set P is a* partial order *on P if it is reflexive, antisymmetric, and transitive. In which case, we say P is a* partially ordered set *or* poset.

Traversals of the poset are examined in the context of the covering relation. It is the smallest relation $\prec$ that is a subset of $\leq$ such that $\leq$ is the transitive closure of $\prec$. Upward traversals visit elements that are greater than a starting element, and downward traversals visit elements that are less than a starting element. Hence, we define up-sets, down-sets, up-closures, and down-closures where P is a poset, $A \subseteq P$, and $x, y \in P$.

Definition 2 (Covering Relation). *If $x < y$ and there is no other $z \in P$ with $x < z < y$, then x is* covered by *y, written $x \prec y$.*

Definition 3 (Up-Sets and Down-Sets). *A is an* up-set *if $x \in A$ and $x \leq y$ implies $y \in A$. Dually, A is a* down-set *if $x \in A$ and $y \leq x$ implies $y \in A$.*

Definition 4 (Up-Closure and Down-Closure). *The* up-closure *of A is the up-set $\{x \in P \mid \exists y \in A : y \leq x\}$ written $\uparrow A$. Dually, the* down-closure *of A is the down-set $\{x \in P \mid \exists y \in A : x \leq y\}$ written $\downarrow A$.*

Note the important role played by the Duality Principle. For each statement in order theory, there is a dual statement in which the sense of every comparison is reversed. If the original statement is true of posets, then so is the dual. As a consequence, structures are defined in dual pairs, and dual results are supplied without explicit proof. In particular, where upwards traversals are defined, downwards traversals follow by duality.

2 Covering Relation Traversals

The central claim of this paper is that traversals of the covering relation are a sound and expressive mechanism for engineering poset algorithms. For the sake of brevity, I contract the term *covering relation traversal* to *poset traversal.* The justification of the claim has two parts. This section proves soundness by showing that poset traversals visit sequences whose extension and order are provably described by order theoretic expressions. Sec. 3 demonstrates the expressiveness by describing a representative collection of poset algorithms.

The examination considers in turn the representation of a poset, the subset visited by a traversal, the effects of pruning by element comparison, and finally the effects of filtering the elements before processing. The combination of poset traversals, pruning by comparison, and filtering generates a collection of algorithms that is further expanded by consideration of the operational differences between traversal algorithms as considered in the subsections that follow. The intention is to express a manner of decomposing mathematical problems into traversals, and to provide context to the selection a particular traversal.

The poset representation used in this paper is expressed in terms of the covering relation. Objects are denoted by conventional mathematical variables, and their data structure components are denoted by calls to *accessor* functions typeset in a fixed-width font. In order to support traversal, the accessor functions identify the maximal and minimal elements in the poset, and the adjacencies in the covering relation expressed via the functions $\mathrm{over}(P, x) = \{y \in P \mid x \prec y\}$ and $\mathrm{under}(P, x) = \{y \in P \mid y \prec x\}$. Each accessor returns a subset of the poset. They are defined with respect to a poset P and an element $x \in P$.

$$\begin{aligned} \texttt{heads}(P) &= \{x \in P \mid \forall y \in P : x \not\prec y\} & \qquad \texttt{over}(P, x) &= \mathrm{over}(P, x) \\ \texttt{tails}(P) &= \{x \in P \mid \forall y \in P : y \not\prec x\} & \qquad \texttt{under}(P, x) &= \mathrm{under}(P, x) \end{aligned}$$

Given the above representation, it is possible to traverse the covering relation in either direction. By analogy to the definitions for up-closure and down-closure, I refer to these directions as *upwards* and *downwards*, hence *upwards poset traversal* and *downwards poset traversal*. A downwards covering relation traversal in a poset is equivalent to an upwards covering relation traversal in its dual. Hence, each traversal algorithm is accompanied by a dual algorithm enjoying dual results. I do not present these dual algorithms, nor their dual results, except to note their name and construction.

Given a poset P and a starting subset $A \subseteq P$, an upwards traversal of P from A is limited to the up-closure of A. Dually, a downwards traversal of P from A is limited to the down-closure of A. This is a simple consequence of transitivity — reachability in the covering relation is equivalent to comparison in the partial order. These simple facts are the initial outline in a picture of poset traversals.

To limit the extent of a traversal pruning tests are introduced. In order to be susceptible to general analysis, tests in an upwards poset traversal must decide *up-conditions* defined as follows. The dual conditions are *down-conditions* required for downwards poset traversal.

Definition 5 (Up-Conditions). *An* up-condition *in a poset P is a formula C in one free variable x such that $\{x \in P \mid C(x)\}$ is an up-set in P.*

Now an upward poset traversal on P that starts at $A \subseteq P$ visits the up-closure of A in P, namely $\uparrow A$. When pruned by an up-condition C_{up} the traversal visits the up-closure of A in P minus those elements belonging to the up-set described by C_{up}, namely $\uparrow A \setminus \{x \in P \mid C_{\mathrm{up}}(x)\}$. The dual applies for downward poset traversals with down-conditions. This simple formulation generates a number of possibilities, because of the complementarity between up-sets and down-sets, and the opportunity to select starting points for either upwards or downwards traversal.

This is best illustrated by directed Venn diagrams. These informal extensions of Venn diagrams depict up-sets by regions that progressively widen upwards, and dually for down-sets. For the following examples illustrated in Fig. 1, let P be a poset, let $A, B \subseteq P$, let $C_{\mathrm{up},B}$ be the up-condition formed by the disjunction of comparisons $b \leq x$ such that $b \in B$, and let $C_{\mathrm{down},B}$ be the down-condition formed by the disjunction of comparisons $x \leq b$ such that $b \in B$. Also note that we can invert the up-condition by writing not $C_{\mathrm{up},B}$, and the down-condition by writing not $C_{\mathrm{down},B}$.

Given the above assignments, it is possible to compute the images in Fig. 1. The union $\uparrow A \cup \uparrow B$ is computed by unpruned upwards traversal from $A \cup B$, and the difference $\uparrow A \setminus \uparrow B$ by upwards traversal from A pruned by $C_{\mathrm{up},B}$. The convex subset $\uparrow A \cap \downarrow B$ is computed by upwards traversal from A pruned by not $C_{\mathrm{down},B}$, or vice-versa. In fact, the direction in which to compute an up-set or down-set

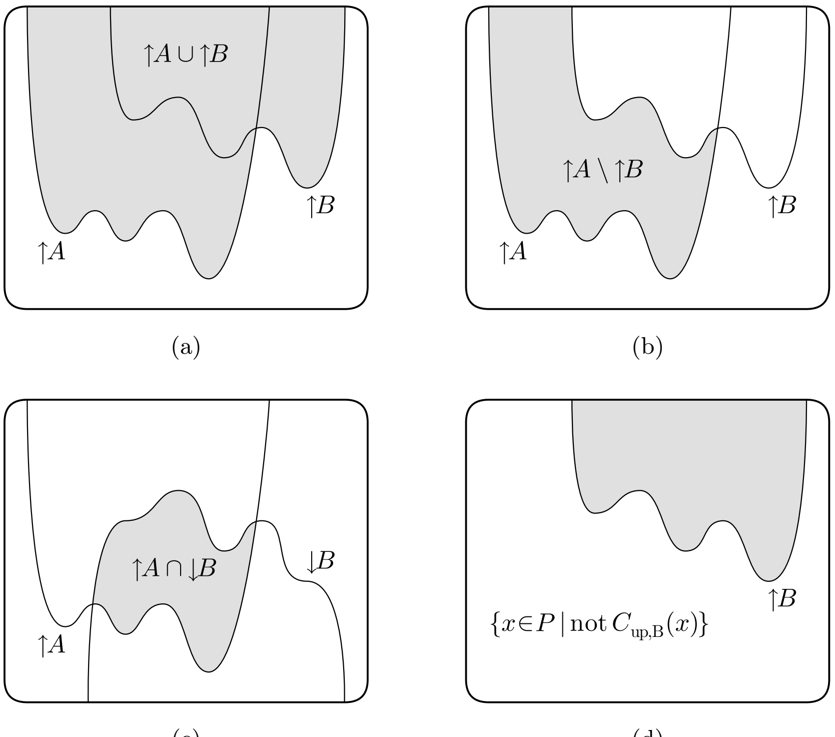

$\uparrow A \cup \uparrow B$
$\uparrow A$
$\uparrow B$
(a)
$\uparrow A \setminus \uparrow B$
$\uparrow A$
$\uparrow B$
(b)
$\uparrow A \cap \downarrow B$
$\downarrow B$
$\uparrow A$
(c)
$\uparrow B$
$\{x \in P \mid \text{not}\, C_{\mathrm{up,B}}(x)\}$
(d)

Fig. 1. Directed Venn diagram constructions via pruned poset traversal

can be elected. So $\uparrow B$ is computed by unpruned upwards traversal from B, or downwards traversal from heads(P) pruned by not $C_{\mathrm{up},B}$. The Duality Principle ensures the observations hold in the dual.

The ability to visit an up-set or down-set by either traversal from a starting set or traversal pruned by a condition is important. A subset of interest may provably be an up-set or down-set, yet we may lack an expression for its minimal or maximal elements respectively. In this case, we can visit the subset by a traversal pruned by the membership condition. For example, to search for the elements that would be covered by an element e if it were inserted into a poset P, visit the down-set $\{x \in P \mid x \leq e\}$ with up-condition $x \not\leq e$.

The problem of visiting an up-set specified by a membership condition rather than its minimal elements contains a second interesting case. Namely, we may hold the membership condition C_A of an up-set A and a starting set S such that $A \subseteq \uparrow S$. In particular, there might be good reason to suspect $\uparrow S \setminus A$ is small, and hence to prefer an upwards traversal. Consider again the example of the insertion of e into a poset P. Once the elements covered by e are computed, the elements covering e are computed. These antichains determine updates required to the representation of P for it to become a representation of $P \cup \{e\}$. The covering elements are contained in the up-closure of the covered elements, and hence we may wish to proceed by starting a new upwards traversal the covered elements.

The technique of *filtering* the visited elements to select a a subset to be processed deserves further consideration. In the above example, the covered elements are a subset of the pruned elements. However, the technique is general and can be applied to extract subsets that are neither up-sets nor down-sets. In addition, it completes the collection of directed Venn diagrams, by including a technique for recovering the intersections during traversal. In this case, it is used to compute upper and lower bounds. The idea by [1] is to represent the starting set by a bit vector, and then during the traversal to propagate information about the reachable elements of the starting set. In short, for each visited element the bit vector is the bitwise OR of the bit vectors of the elements it covers. The visited elements are then filtered according to whether their bit vector is complete.

The above discussion considers *which* elements are visited, but not *when* they are visited. Secs. 2.1, 2.2, and 2.3 investigate this question for breadth-first, depth-first, and topological traversals. The order of visits affects the implementation of the procedures to decide the pruning condition and operate on visited elements, and their suitability as the basis for other poset algorithms.

Given the role of up-conditions and down-conditions in this approach, the traversals of Secs. 2.1, 2.2, and 2.3 are described by algorithms parametrized by a procedure to decide the pruning condition, and a procedure to operate on the visited element. For example, breadth-first traversal is parametrised by the procedures `Prune` and `Pre`. This raises issues in the proofs of correctness for the algorithms, because we must consider the behaviour of these procedures to reason about the operation of the algorithms.

Clearly neither procedure should exhibit side effects in the traversal algorithm. Namely, the pruning and visiting procedures should not alter the structure being traversed, nor the state of the traversal. A more subtle problem concerns the correct behaviour of the pruning decision procedure.

It is tempting to demand that the pruning decision be independent of the position in the traversal. However, in Sec. 2.3 we note that this restriction would rule out many useful algorithms that exploit the properties of topological traversal. In particular, it rules out the propagation of reachability information as discussed. A more sophisticated assumption can be stated for the case of topological traversal by exploiting the properties of the algorithm.

The next objective is to bridge the terminology of graph theory and order theory in the context of poset traversal, in particular, to relate membership in the visited subset with characteristic paths in the covering relation. It is helpful to define a term denoting the collection of paths that extend from a starting set A under pruning. This notational convenience allows us to succinctly state the characteristics of the paths in a poset traversal. An *ascendible path* in the covering relation extends from an element of the starting subset and terminates at an unpruned element. The *descendible paths* are defined dually.

Definition 6 (Ascendible Paths). *Let P be a poset, let $A \subseteq P$, and let C_{up} be an up-condition. Then, the* ascendible paths *are the $\prec$ paths denoted as follows.*

$$\text{ascendible}(P, A, C_{\text{up}}) = \left\{ \begin{array}{r} \langle x_0, x_1, \ldots, x_m \rangle \in P^* \mid x_0 \prec x_1 \prec \cdots \prec x_m \\ \text{and } x_0 \in A \\ \text{and not}\, C_{\text{up}}(x_m) \end{array} \right\}$$

The connection between order theoretic properties and graph theoretic properties can be made explicit via the traversable paths of Definition 6. Given a poset P and a visited subset $V = \{x \in {\uparrow}A \mid \text{not}\, C_{\text{up}}\}$, membership in V is equivalent to the existence of a path in $\text{ascendible}(P, A, C_{\text{up}})$. Furthermore, every such path is contained within V, so that the traversal of V does not visit elements outside of V. The results are a simple consequence of the fact that membership to an up-set is monotonic with respect to the sequence of elements in a covering relation path.

Remark 1. Let P be a finite poset, let $A \subseteq P$, let C_{up} be an up-condition, and let $V = \{x \in {\uparrow}A \mid \text{not}\, C_{\text{up}}(x)\}$.

$$v \in V \iff \exists \langle x_0, x_1, \ldots, x_m \rangle \in \text{ascendible}(P, A, C_{\text{up}}) : x_m = v \tag{1}$$

$$\text{ascendible}(P, A, C_{\text{up}}) \subseteq V^* \tag{2}$$

2.1 Breadth-First Traversal in Posets

The breadth-first traversal of a poset uniformly expands the set of visited elements across the breadth of a boundary. This ensures properties relevant to some applications of graph theory but less relevant to the role of the covering relation in order theory.

```
Traverse-Up-Breadth-First ($P, B_{\text{init}}$, Prune, Pre)
  A1: let $S_{\text{visited}} \subseteq P \longleftarrow \emptyset$
  A2: let $B \subseteq P \longleftarrow \{x \in B_{\text{init}} \mid \text{not}\, \texttt{Prune}(P, x)\}$
  A3: while $B \neq \emptyset$
  A4:     for all $x \in B$
  A5:         Pre($P, x$)
  A6:         $S_{\text{visited}} \longleftarrow S_{\text{visited}} \cup \{x\}$
  A7:     $B \longleftarrow$ Traverse-Up-Breadth-First-Succ ($P, B$, Prune)

Traverse-Up-Breadth-First-Succ ($P, B$, Prune)
  B1: let $B_{\text{succ}} \subseteq P \longleftarrow \emptyset$
  B2: for all $x \in B$
  B3:     for all $v \in \texttt{over}(P, x)$
  B4:         if $v \notin S_{\text{visited}}$ and not Prune($P, v$)
  B5:             $B_{\text{succ}} \longleftarrow B_{\text{succ}} \cup \{v\}$
  B6: return $B_{\text{succ}}$
```

Fig. 2. Perform an upwards breadth-first traversal of poset P from $B_{\text{init}} \subseteq P$. The traversal is pruned of elements for which `Prune` is true, and `Pre` is applied to unpruned elements during pre-order visits.

The principal and well know property of breadth-first traversal is that the sequence of boundaries is ordered by path length. Namely, if B_0 is the set of initial elements and B_i is the i^{th} boundary, then for every element $x \in B_i$, the length of the shortest path from an element in B_0 to x is i. In other words, breadth-first traversal ranks each node according to its minimum distance from any of the initial nodes. Fig. 2 defines a procedure `Traverse-Up-Breadth-First` that implements upwards poset traversal. The dual procedure `Traverse-Down-Breadth-First` implements downwards poset traversal.

It is straightforward to prove that an upwards breadth-first poset traversal pruned by an up-condition visits the difference of the up-closure of the initial elements, and the elements selected by the up-condition. To begin, note that preceding values of B partition the visited subset S_{visited} by Lines A4–A6, and each subsequent value of B is disjoint from S_{visited} by Lines A7, and B1–B5. By induction on lengths of ascendible paths, an unpruned path of length m, from an initial element to an element x, guarantees x is visited in the m^{th} execution of the loop, so long as no shorter ascendible path exists. Namely, elements are visited according to the length of the shortest path from an initial element. Finally, the elements of $\{x \in {\uparrow}B_{\text{init}} \mid \text{not}\, C_{\text{up}}(x)\}$ enjoy ascendible paths by Remark 1. Hence, the elements are visited, because every ascendible path is of finite length in finite P, and the lengths of shortest ascendible paths form a sequence given each prefix is itself a shortest ascendible path.

The path-length properties of breadth-first traversal are of little relevance to order theory. Hence, `Traverse-Up-Breadth-First` has limited application, especially given the properties of depth-first and topological traversal detailed in Secs. 2.2 and 2.3. However, the algorithm exemplifies an approach to sequencing operations that will be the subject of further analysis.

```
Traverse-Up-Depth-First($P, S_{\text{init}}$, Prune, Post)
C1: let $S_{\text{visited}} \subseteq P \longleftarrow \emptyset$
C2: for all $x_{\text{init}} \in S_{\text{init}}$
C3:     if $x_{\text{init}} \notin S_{\text{visited}}$ and not Prune($P, x_{\text{init}}$)
C4:         Traverse-Up-Depth-First-Visit($P, x_{\text{init}}$, Prune, Post)

Traverse-Up-Depth-First-Visit($P, x$, Prune, Post)
D1: for all $v \in$ over($P, x$)
D2:     if $v \notin S_{\text{visited}}$ and not Prune($P, v$)
D3:         Traverse-Up-Depth-First-Visit($P, v$, Prune, Post)
D4: Post($P, x$)
D5: $S_{\text{visited}} \longleftarrow S_{\text{visited}} \cup \{x\}$
```

Fig. 3. Perform an upwards depth-first traversal of poset P from $S_{\text{init}} \subseteq P$. The traversal is pruned of elements for which `Prune` is true, and `Post` is applied to unpruned elements during post-order visits.

Two properties of the breadth-first traversal algorithm are of interest. First, breadth-first traversal processes elements in a sequence of subsets that partition the visited elements. While the subsets have no special order theoretic property, the idea has a natural extension in topological traversal, where the subsets are antichains. Second, breadth-first traversal is immediate in its processing of covering elements. By comparison, depth-first traversal visits an element only after visiting all greater elements. Thus, breadth first traversal is suggestive of lazy search, where boundaries are computed on demand.

2.2 Depth-First Traversal in Posets

The depth-first traversal of a poset extends the current path from the initial element wherever possible. Namely, it deepens the search first. In contrast to breadth-first traversal, it must backtrack to visit all elements reachable from the initial elements. Therefore, rather than maintain a set representing the boundary, the algorithm maintains a stack representing the current path.

The principal and well known property of depth-first traversal is that the post-order visit of a node x occurs only after the post-order visit of all nodes reachable from x. Thus, the sequence of post-order visits in a downwards depth-first traversal is a linear extension of the partial order, and the sequence of post-order visits in an upwards depth-first traversal is a linear extension of the dual of the partial order. This property is very useful, because it provides an invariant on the execution of `Post` that expresses the partial order, namely that every element greater than the current element has been processed. Fig. 3 defines a procedure `Traverse-Up-Depth-First` that implements upwards poset traversal. The dual procedure `Traverse-Up-Depth-First` implements downwards poset traversal.

The linear extension property of depth-first traversal is of particular importance to client algorithms. Hence, it is asserted on the execution state of `Traverse-Up-Depth-First`, and the execution of `Post` in particular.

Lemma 1. *Let* `Traverse-Up-Depth-First`$(P, S_{\text{init}},$ `Prune`, `Post`$)$ *invoke Fig. 3 where P is a finite poset, $S_{\text{init}} \subseteq P$,* `Prune` *decides an up-condition C_{up}, and* `Prune` *and* `Post` *are side effect free in* `Traverse-Up-Depth-First`*. Then,*

$$\{u \in P \mid x < u \text{ and not } C_{\text{up}}(u)\} \subseteq \{u \in P \mid \texttt{Post}(P, u) \textit{ is executed}\}$$

at `Post`(P, x).

Proof. Let $A(x)$ assert Lemma 1 is satisfied at `Post`(P, x). Assume $A(v)$ is true for all $v \in \text{over}(P, x) : \text{not } C_{\text{up}}(v)$. Note updates to S_{visited} are modeled by set union given Line D5. Then at `Post`(P, x)

$$\begin{aligned} S_{\text{visited}} &\supseteq \bigcup_{v \in \text{over}(P,x):\text{not } C_{\text{up}}(v)} \{u \in P \mid v < u \text{ and not } C_{\text{up}}(u)\} \cup \{v\} \\ &= \bigcup_{v \in \text{over}(P,x)} \{u \in P \mid v \leq u \text{ and not } C_{\text{up}}(u)\} \\ &= \uparrow \text{over}(P, x) \cap \{u \in P \mid \text{not } C_{\text{up}}(u)\} \\ &= \{u \in P \mid x < u\} \cap \{u \in P \mid \text{not } C_{\text{up}}(u)\} \end{aligned}$$

because $C_{\text{up}}(v)$ iff $C_{\text{up}}(u)$ for all $v \leq u$ by Definition 5, then by distributive laws and definition of up closure. Hence by mathematical induction because the assumption agrees with the recursion in Lines D1–D4. □

The next lemma confirms that an upwards depth-first traversal pruned by an up-condition visits the difference of the up-closure of the initial elements, and the elements selected by the up-condition. It is proved by showing the inclusion in both directions of the equation.

Lemma 2. *Let* `Traverse-Up-Depth-First`$(P, S_{\text{init}},$ `Prune`, `Post`$)$ *invoke Fig. 3 where P is a finite poset, $S_{\text{init}} \subseteq P$,* `Prune` *decides an up-condition C_{up}, and* `Prune` *and* `Post` *are side effect free in* `Traverse-Up-Depth-First`*.*

$$\{x \in P \mid \texttt{Post}(P, x) \textit{ is executed}\} = \{x \in \uparrow S_{\text{init}} \mid \text{not } C_{\text{up}}(x)\}$$

While depth-first traversal is recommended by its simplicity and order theoretic properties, a weakness is that the linear extension generated by post-order visits does not extend to the sequence of pruning tests. However, there are occasions when the fact that the pruning test is applied to all visited elements and all elements covering the visited elements is of specific use. In particular, information that is discovered in failing calls to `Prune` can be passed to the post order call to `Post`.

2.3 Topological Traversal in Posets

Topological traversal is distinguished by order theoretic properties of the sequence of visits, and its frugality with respect to element comparison. It also

```
Traverse-Up-Topological($P, A_{\text{init}}$, Prune, Pre)
  E1: let $S_{\text{invited}} \subseteq P \longleftarrow$ Up-Closure($P, A_{\text{init}}$)
  E2: let $S_{\text{visited}} \subseteq P \longleftarrow \emptyset$
  E3: let $A \subseteq P \longleftarrow \{x \in A_{\text{init}} \mid \text{not Prune}(P, x)\}$
  E4: while $A \neq \emptyset$
  E5:     for all $x \in A$
  E6:         Pre($P, x$)
  E7:         $S_{\text{visited}} \longleftarrow S_{\text{visited}} \cup \{x\}$
  E8:     $A \longleftarrow$ Traverse-Up-Topological-Succ($P, A$, Prune)

Traverse-Up-Topological-Succ($P, A$, Prune)
  F1: let $A_{\text{succ}} \subseteq P \longleftarrow \emptyset$
  F2: for all $x \in A$
  F3:     for all $v \in$ over($P, x$)
  F4:         if under($P, v$) $\cap S_{\text{invited}} \subseteq S_{\text{visited}}$ and not Prune($P, v$)
  F5:             $A_{\text{succ}} \longleftarrow A_{\text{succ}} \cup \{v\}$
  F6: return $A_{\text{succ}}$
```

Fig. 4. Perform an upwards topological traversal of poset P from the antichain $A_{\text{init}} \subseteq P$. The traversal is pruned of elements for which `Prune` is true, and `Pre` is applied to unpruned elements during pre-order visits.

shares properties with both breadth-first and depth-first traversal. Like breadth-first traversal, it iterates over a sequence of disjoint boundaries; and like depth-first traversal, it visits elements in a linear extension of the partial order.

Informally, topological traversal can be viewed as a variant of breadth-first traversal, where the graph theoretic path length property guarantees the order of visits is a linear extension of the partial order. An element $x \in P$ is a member of the i^{th} antichain if and only if the length of the *longest* ascendible path from the starting antichain A_{init} to x is i. Thus, an element is visited only once all lesser elements have been visited. Fig. 4 defines a procedure `Traverse-Up-Topological` that implements upwards poset traversal. The dual procedure `Traverse-Down-Topological` implements downwards poset traversal.

The initial analysis of `Traverse-Up-Topological` follows that of `Traverse-Up-Breadth-First`. Namely, the boundaries partition the traversal, and boundary membership is decided by the lengths of ascendible paths. Again, note that preceding values of A partition S_{visited} by Lines E5–E7, and each subsequent value of A is disjoint from S_{visited} by Lines E8, and F1–F6. Then, Lemma 3 asserts that an unpruned path of length m, from a minimal element in P to an element x, guarantees x is visited in the m^{th} execution of the loop, so long as no *longer* ascendible path exists. Namely, elements are visited according to the length of the longest path from an initial element. It is also shown by induction on lengths of ascendible paths.

Lemma 3. *Let* `Traverse-Up-Topological`(P, A_{init}, `Prune`, `Pre`) *invoke Fig. 4 where P is a finite poset, $A_{\text{init}} \subseteq P$ is an antichain,* `Prune` *decides an up-condition C_{up} under topological order, and* `Prune` *and* `Pre` *are side effect free in*

`Traverse-Up-Topological`. *Then,*

$$\begin{aligned} x \in A_n \iff & \ (\ \exists\langle x_0, x_1, \ldots, x_m\rangle \in \text{ascendible}(P, A_{\text{init}}, C_{\text{up}})\ : \\ & \qquad x_m = x \text{ and } m = n\ ) \\ & \text{and } (\ \forall\langle x_0, x_1, \ldots, x_m\rangle \in \text{ascendible}(P, A_{\text{init}}, C_{\text{up}})\ : \\ & \qquad x_m = x \implies m \le n\ ) \end{aligned}$$

where A_n is the value of A after Lines E5–E8 have executed n times.

The next lemma states that elements are visited only after all lesser elements have been visited. The distinction between Lemmata 1 and 4 is that `Traverse-Up-Topological` visits elements by iteration over antichains, and begins ordering the elements along the initial ascending paths rather than post-order. This second fact is noted by amending the proposition to assert the ordering at both `Prune`(P, x) and `Pre`(P, x). The result is a direct consequence of Lemma 3, because if $x \prec y$ then the longest ascendible path to y is greater than that to x.

Lemma 4. *Let* `Traverse-Up-Topological`$(P, A_{\text{init}},$ `Prune`, `Pre`$)$ *invoke Fig. 4 where P is a finite poset, $A_{\text{init}} \subseteq P$ is an antichain,* `Prune` *decides an up-condition C_{up} under topological order, and* `Prune` *and* `Pre` *are side effect free in* `Traverse-Up-Topological`. *Then,*

$$\begin{aligned} \{u \in {\uparrow}A_{\text{init}} \mid u < x\} &\subseteq \{u \in {\uparrow}A_{\text{init}} \mid \texttt{Pre}(P, u) \textit{ is executed}\} \\ &\subseteq \{u \in {\uparrow}A_{\text{init}} \mid \texttt{Prune}(P, u) \textit{ is executed}\} \end{aligned}$$

at `Prune`(P, x) *and also at* `Pre`(P, x).

The next lemma confirms that an upwards topological traversal pruned by an up-condition visits the difference of the up-closure of the initial elements, and the elements selected by the up-condition. However, this result extends to the elements visited by `Prune`. Note that the elements in this difference enjoy ascendible paths by Remark 1. Hence, Lemma 3 guarantees the elements are visited.

Lemma 5. *Let* `Traverse-Up-Topological`$(P, A_{\text{init}},$ `Prune`, `Pre`$)$ *invoke Fig. 4 where P is a finite poset, $A_{\text{init}} \subseteq P$ is an antichain,* `Prune` *decides an up-condition C_{up} under topological order, and* `Prune` *and* `Pre` *are side effect free in* `Traverse-Up-Topological`.

$$\begin{aligned} &\{x \in P \mid \texttt{Pre}(P, x) \textit{ is executed}\} \\ &\quad = \{x \in {\uparrow}A_{\text{init}} \mid \text{not}\, C_{\text{up}}(x)\} \end{aligned} \tag{3}$$

$$\begin{aligned} &\{x \in P \mid \texttt{Prune}(P, x) \textit{ is executed}\} \\ &\quad = \{x \in {\uparrow}A_{\text{init}} \mid \text{not}\, C_{\text{up}}(x)\} \cup \text{tails}(\{x \in {\uparrow}A_{\text{init}} \mid C_{\text{up}}(x)\}) \end{aligned} \tag{4}$$

Topological traversal is recommended by its frugality with respect to pruning tests. By Lemma 3, an element is not a candidate for testing until it has been discovered along a longest ascendible path. The implementation of this condition in Line F4 is straightforward — an element $x \in P$ is not a candidate until it has been discovered along every incoming edge. This is sound, because all ascendible paths occur wholly within the unpruned space by Remark 1. Thus, an element remains untested if a covered element is pruned or remains untested, and thus failure propagates by indefinite deferral — pruning a single element $x \in P$ ensures that all elements reachable from x are never tested. Consider an upward poset traversal from A. For breadth-first and depth-first traversals, $x \in P$ is tested if and only if $x \in A$ or there exists an incoming edge originating from a visited element. For topological traversal, $x \in P$ is tested if and only if $x \in A$ or every incoming edge originates from a visited element. Hence, it requires fewer pruning tests.

A further important property concerns the order of pruning tests. Superficially, both depth-first and topological traversal generate linear extensions of the partial order or its dual. However, in a topological traversal the pruning tests also enjoy this ordering, because depth-first traversal recovers the order in post-order visits while topological traversal recovers the order in pre-order visits. This has important consequences for algorithms based upon topological traversal. At $x \in P$ in an upwards topological traversal, the pruning of elements greater than x can be informed by the processing of elements less than x. This fact plays a key role in many algorithms. Hence, the pruning decision procedure can be granted greater freedom by demanding only that `Prune` decide an up-condition under *topological order*. This phrase denotes a sanction to prove the correctness of `Prune` by mathematical induction, so long as the induction is framed in terms of Lemma 4 concerning the order of visits.

A potential shortcoming of topological traversal is that it must distinguish between elements that have not yet been visited, and elements that are outside of the traversal. It is possible to avoid this issue if we search the poset as a whole. However, there are instances where the generalisation of topological traversal to include additional starting sets is important. For example, consider the case where we have $x \in P$ and wish to search for k given $x < k$. In this case, we can restrict the search to $\uparrow \{x\}$ rather than search P.

`Traverse-Up-Topological` naively resolves the problem of distinguishing between unvisited elements and elements outside of the traversal by first constructing $\uparrow A_{\text{init}}$. This somewhat negates the advantage that topological traversal generates the linear extension in pre-order visits. While less naive implementations clearly depend upon exploitable specifics, it is worth asking whether there are general considerations impacting on `Traverse-Up-Topological`.

It is tempting to propose that breadth-first and topological traversals be interleaved. Namely, that a breadth-first traversal operate as a *coroutine* of a topological traversal, in much the same way a lexical analyser supports a parser. However, each step of a breadth-first traversal generates the next rank of shortest ascendible paths, while the question in Line F4 is whether there exist *any* undiscovered, ascendible paths. Topological traversal must defer the visit if there exists an undiscovered, ascendible path. Breadth-first search cannot refute the existence of ascendible paths until it has exhausted the poset.

We cannot avoid these problems without resorting to sources of information outside the traversal. Two possibilities emerge: either we resort to applying a comparison to determine whether the unvisited elements belong to the up-closure; or we employ ranking information stored from previous topological traversals. The first solution annuls the comparison efficiency. The second solution requires additional mechanisms to keep the longest ascendible path ranking up to date. Techniques to solve these problems are not considered in this paper, because they do no affect the mathematical properties of the topological traversal algorithm.

3 Conclusion

In summary, this paper provides a detailed *framework* within which distinct classes of poset traversal are subsequently developed. This framework provides both a detailed sketch of the traversable subsets of a poset, and definitions and propositions to assist in realising this sketch under distinct types of poset traversal. This allows many problems concerning partial orders to be decomposed into traversals of the covering relation. Therefore, I conclude by providing examples that demonstrate the most important techniques to realise an abstract data type for posets. These cases exemplify the application of the generic poset traversal algorithms to important poset algorithms, and employ the techniques depicted in Fig. 1 and discussed in Sec. 2. Together Sec. 2 and this list justify the claim that traversals of the covering relation are a sound and expressive mechanism for engineering poset algorithms.

Covered and Covering Elements. Topological traversal is well suited to searching a poset for a match with a key. It also recovers the elements covered by the key in case no match exists. Dually, downwards topological search recovers the elements that cover the key. These two sets are required to update the covering relation for element insertion. Upwards topological search traverses the elements less than or equal to the key by starting at the minimal elements and pruned elements that are not less than or equal to the key.

Extremal Upper and Lower Bounds. The computation of extremal upper and lower bounds by topological traversal exemplifies two important techniques exploiting the linear extension property of Lemma 4. The first is the recovery of the extremal elements of the pruned subset, by simply recording elements as they are pruned. By Lemma 4, if $x \in P$ is pruned then elements greater than x cannot be visited by `Prune`, so that `Prune` is successfully applied to exactly the minimal elements in $\{x \in P \mid C_{\text{up}}(x)\}$. The second is the propagation of reachability information during traversal. In the case of reachability information, the ordering on visited elements allows us to prove, by induction, that each visited element enjoys complete information about the reachability of its covered elements.

Element Removal. The first task of element removal is to determine reachability after erasure. Namely, given $\langle u, v\rangle \in \text{under}(P, e) \times \text{over}(P, e)$, we require an algorithm to determine the existence of an ascendible path from u to v that bypasses e. One approach is to erase e and adjacent edges, and then test reachability. This approach is inelegant in that the modified covering relation is inconsistent with both $P \setminus \{e\}$ and P. An alternative is to compute the reachability on the unmodified poset P by a topological traversal that computes reachability in `Prune` given Lemma 4 with the exception that `Prune` discards reachability information at e. `Prune` also returns false at an element of `over`(P, e), because `over`(P, e) is an antichain and hence the elements are pairwise unreachable, and returns false once all elements of `over`(P, e) have been discovered.

Join and Meet Operator Updates. The family of algorithms associated with the incremental maintenance of join or meet operators provides excellent examples of algorithms using depth-first traversal. The technique of representing the binary join or meet operator of a lattice by a table, and then retrieving the partial order via the Connecting Lemma is useful for type lattices.

The algorithms are based upon nested depth-first traversals of the lattice. This is because the join and meet operators are composed of triples, in which the elements can be distinguished as either operands or results. For example, the equation $x \vee y = z$ is represented by a triple $\langle x, y, z\rangle$ in which x and y are operands, and z is a result. A lattice traversal enumerates values for a single element in the triple. By nesting traversals we are able to enumerate values for a pair of elements. For example, to generate all pairs $\langle x, y\rangle \in K \times K$ we traverse the lattice to enumerate values for x, and for each such value we traverse the lattice again to enumerate values for y. By careful application of the Connecting Lemma and judicious use of pruning tests, we can develop traversals restricted to the triples requiring update.

References

1. Ellis, G.: Managing Complex Objects. PhD thesis, Department of Computer Science, University of Queensland, Brisbane, Australia (1995)
2. Wille, R.: Concept lattices and conceptual knowledge systems. Computers and Mathematics with Applications 23(6-9), 493–515 (1992)
3. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison Wesley, Reading (1984)
4. Valtchev, P.: An algorithm for minimal insertion in a type lattice. Computational Intelligence 15(1), 63–78 (1999)
5. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoretical Artificial Intelligence 14, 189–216 (2002)
6. Ganter, B., Kuznetsov, S.O.: Stepwise construction of the Dedekind-MacNeille completion. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS, vol. 1453, pp. 295–302. Springer, Heidelberg (1998)
7. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1990)

Representing and Reasoning about Different Viewpoints: An Agronomy Application

Madalina Croitoru[1] and Rallou Thomopoulos[1,2]

[1] LIRMM (CNRS and Univ. Montpellier II), F–34392 Montpellier cedex 5, France
croitoru@lirmm.fr
[2] INRA, UMR1208, F–34060 Montpellier cedex 1, France
rallou.thomopoulos@supagro.inra.fr

Abstract. Real-world applications are often complex systems where several ways of analysing a given situation can be expressed, depending on actors' viewpoints. This paper proposes a semantically sound syntactic extension to Conceptual Graphs, namely Conceptual Graph Assemblies (CGAs), that allows the representation of multiple viewpoints on the same situation. Several reasoning mechanisms, based on the projection operation, corresponding to different strength levels and adapted to multi-viewpoints situations are then demonstrated. Several modelling scenarios are then proposed and our work is put in the context of real world examples from the agri-food domain.

1 Introduction

Quality control within agri-food chains relies on numerous criteria: nutritional, functional, sanitary, environmental, economical, etc. The management of food quality has to reconcile several facets constituted by these criteria. Moreover, the objectives of quality are based on several actors: technicians, managers, users, scientists, professional associations, public communities, etc. The importance attached to the different quality criteria varies according to the considered actors. These elements lead to the following open research questions: "how to represent, within a knowledge representation model, these contradictory viewpoints?"; "how to take into account, by the reasoning mechanisms, the interests of the different involved actors?"

The current structure of chains is questioned as for system perenniality, protection of the environment, public health issues, cost and energy. The actors' viewpoints are divergent, hence it is necessary to define representational and reasoning mechanisms able to model and take into account the balance between viewpoints, and the risks and benefits they imply. Our general objective is the conception of a decision support tool for the actors of an agri-food chain, in presence of contradictory viewpoints and priorities.

In this context, as a first step, we built a knowledge-based system able to represent the different kinds of knowledge needed, initially provided with consistency checking, querying and symbolic simulation mechanisms. Given that the information sources are both experimental data extracted from the domain literature and expert statements, the intuitiveness and proximity to natural language of the representation language are essential features. Moreover, the experts should be able to understand the reasoning on their modelling and to validate it, thus reasoning should be done directly on the knowledge representation and feedback intuitive. Finally, a logical semantics is desirable as a foundation for reasoning and the language should be flexible enough to be easily extended to new features. For these reasons, conceptual graphs were initially chosen as the knowledge representation and reasoning language for this specific application.

However, conceptual graphs cannot easily represent different, potentially contradictory viewpoints, and moreover, rigorous mechanisms for reasoning about this type of knowledge have not been put into place. In this paper we present a formalism that allows the representation of such contradictory, inconsistent type of knowledge for this application along with sound and complete syntactic operation for manipulation.

A simple case of this problem has been addressed by Puder [9] who considered alternative descriptions for one concept. He built a tree with this concept as a root node, and used this structure for service trading in the AI-Trader project[1]. This work is not sufficient in the context of the agronomy domain where whole sentences could be debated and argued upon. In [10], an approach for viewpoint representation is proposed in the framework of the conceptual graph model, however it concerns the expression of facets of concepts in an ontology, i.e. the terminological part of the model (the support), and does not treat the commensal representation of several viewpoints in the assertional knowledge. Another approach for representing viewpoints in the conceptual graph model is based on nested graphs. They have been introduced at a descriptive level by Sowa [12] as a way of representing contexts by structuring knowledge by levels, and studied in further works such as [7,8]. Typed nestings were introduced by [2], which allows to specify the relationship (description, explanation, etc.) between the surrounding vertex and one of its descriptions and thus to explicitly attach several descriptions to the same vertex. Each description can then be viewed as a viewpoint, as proposed in [13] which more specifically focuses on how to associate specific vocabularies with contexts. A drawback of the nested graph approach is that it does not allow inter-viewpoint reasoning, such as inter-viewpoint projection or detection of contradictions between viewpoints. In [5,6] an extension to Conceptual Graphs was proposed to further address the above mentioned modelling needs. However, the proposed formalism was lacking in rigorousness by the fact that the combinatorial structures proposed were not complete with respect to the proposed semantics [5]. While this problem has been partially solved in [6] the lack of a concrete practical framework to address the concrete modelling needs of the agri-domain was still to be addressed.

In this paper we extend this formalism by showing different combinatorial structures of defining sound and complete viewpoints as well as demonstrating their applicability for the above mentioned problem in the agronomy domain. Section 2 presents a motivating example, Section 3 introduces the formalism,

[1] http://www.puder.org/aitrader/

Section 4 shows how CGAs can be used in conceptual modelling, finally Section 5 concludes with some perspectives.

2 Motivating Example

Conceptual graphs [11,12] (CGs) are a logical, graph-based approach to knowledge representation that introduce a clear distinction between ontological and asserted knowledge. More specifically, a Conceptual Graph represents knowledge as a support and an associated bipartite graph. The support encodes the ontological, background information. It consists of a concept and a relation taxonomy along with the markers used to denote instances or generic concepts. The factual information is depicted as a bipartite graph where one partition class, the concepts, is represented using square nodes, and the other, the relations, is represented using ovals. An example of a Conceptual Graph is depicted in the figure below:

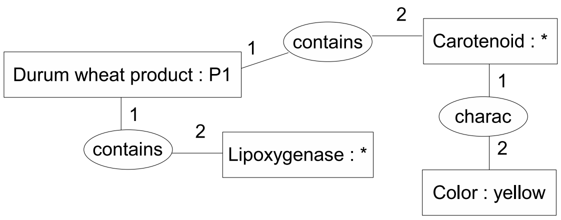


Fig. 1. Example of a Conceptual Graph

The conceptual graph in this figure states that the durum wheat product P1 contains a lipoxygenase and carotenoid which is characterised by the yellow color.

Reasoning with Conceptual Graphs means translating the Conceptual Graph into FOL (First Order Logic) formulae and employing FOL deduction. Another method looks at finding a homomorphism (projection) between two graphs defined on the same support. These two methods have been proven equivalent [1,12].

However, Conceptual Graphs can only represent static, "snapshot" facts about the world. Indeed, the support encodes the hierarchies which classify the entities and relations we need to describe a certain scene, while the bipartite graph represents that scene. We do not have a proper built-in mechanism to describe alternative scenes (e.g. as viewed from different / inconsistent view points). An example of the expressivity needed for this application is depicted in figure 2.

In this figure two viewpoints are represented about the information given by a conceptual graph: the scientist viewpoint (denoted "Sc.") and the marketing viewpoint (denoted "Mk."). The scientist view indicates that the durum wheat product P1 contains carotenoid characterised by the yellow color, lipoxygenase

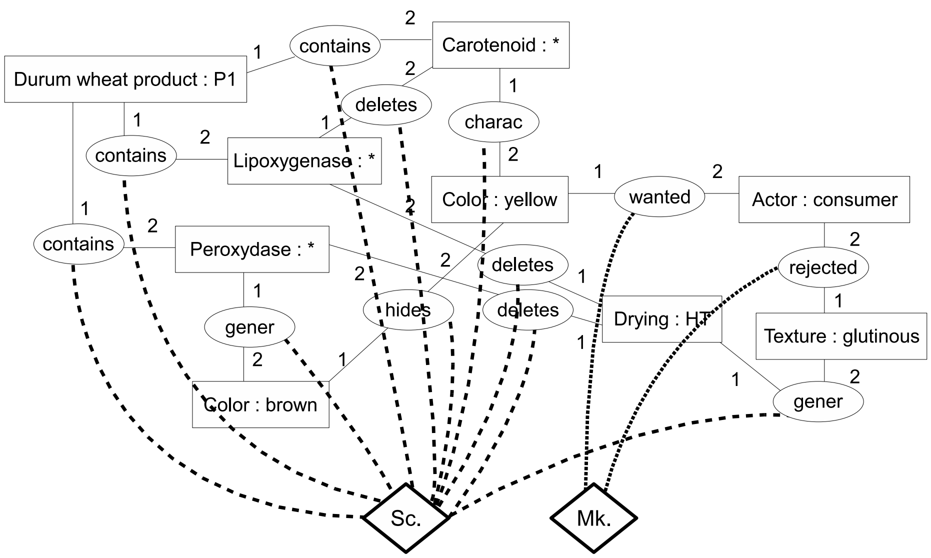


Fig. 2. Multiple viewpoints in the agri-food application

that deletes carotenoid, and peroxydase that generates a brown color which hides the yellow one. It also indicates that the HT (High Temperature) drying deletes lipoxygenase and peroxydase and generates a glutinous texture. The marketing view indicates that the yellow color is wanted by the consumer and that the glutinous texture is rejected by the consumer.

In this paper we propose a syntactic, semantically sound mechanism for representing the expressivity needs mentioned above. Viewpoints are represented using different combinatorial grouping in the Conceptual Graphs Assemblies (CGA). Reasoning about viewpoints is done using an extension of the projection mechanism that respects the combinatorial structure induced by the CGA. In the following example the query is searched in all of the viewpoints therefore denoting a consensus.

For example, consider the following simple query composed of a single concept vertex:

Fig. 3. Example of a simple query

The meaning of searching for this query in all of the viewpoints is highlighting product properties that are of interest for all of them, e.g. in figure 2 for both scientists and marketing. In the example of figure 2, there are two answers to this query that appear in all the viewpoints. These answers correspond to the concept vertices represented in greyed out shade in figure 4.

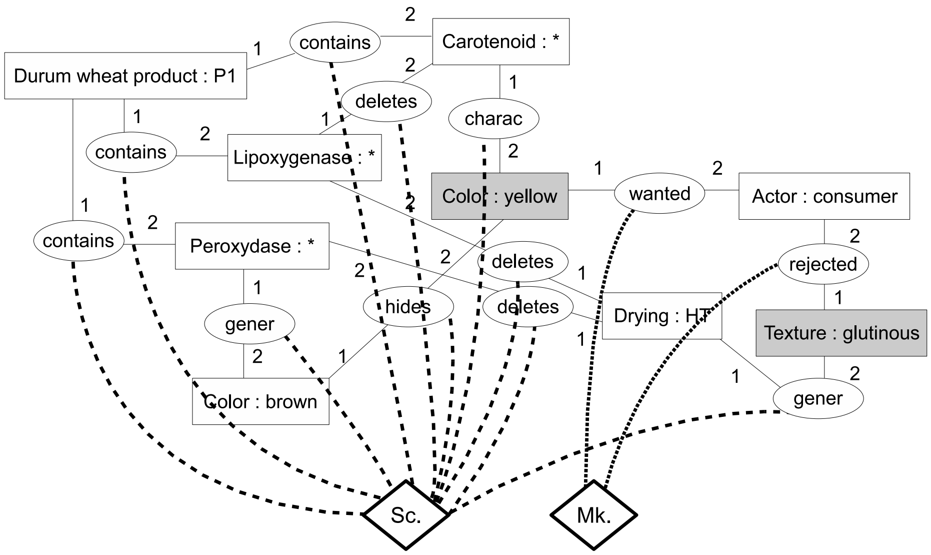


Fig. 4. Answers to the query

3 Formalism

An ordered bipartite graph is a triple which consists of a set of concept nodes, a set of relation nodes and a set of mappings between the relation nodes and nonempty finite sequences over concept nodes. An ordered bipartite graph with just one relation node is called a star graph. We consider a special kind of subgraphs for our modelling purposes, namely spanned subgraphs. A spanned subgraph induced by a set of relation nodes consists of the set of relation nodes, the edges incident with these and the corresponding concept nodes.

Definition 1. (Ordered Bipartite Graph)
A triple $G = (V_C, V_R, N_G)$ is called an **ordered bipartite graph** *if*
- V_C and V_R are finite disjoint sets, ($V_G := V_C \cup V_R$ is the vertices set *of G), and*
- $N_G : V_R \to V_C^+$ is a mapping; V_C^+ is the set of all finite nonempty sequences over V_C.

For $r \in V_R$ with $N_G(r) = c_1 \dots c_k$, $d_G(r) := k$ is the **degree** *of r in G and $N_G^i(r) := c_i$ is the i-***neighbour** *of r in G. The set of (distinct) neighbours of r is denoted $\overline{N}_G(r)$.*

The multiset E_G of edges *of G is $E_G = \big(\{c, r\} | c \in V_C, r \in V_R$ and $\exists i$ such that $N_G^i(r) = c\big)$.*

We further assume that for each $c \in V_C$ there is $r \in V_R$ and $i \in \mathbb{N}$ such that $c = N_G^i(r)$ (G has no isolated vertices).

An ordered bipartite graph $G = (V_C, V_R, N_G)$ with $|V_R| = 1$ is called a **star graph**.

If $G = (V_C, V_R, N_G)$ is an ordered bipartite graph and $A \subseteq V_R$, the **subgraph spanned by** A **in** G *is the graph* $\mathbf{G[A]} := (V_C^1, A, N_G^1)$*, where N_G^1 is the restriction of N_G to A and $V_C^1 = \{c \in V_C | \exists r \in A$ and $\exists i \in \mathbb{N}$ such that $c = N_G^i(r)\}$.*

If $A = \{r\}$, then we simply write $G[r]$, which is referred to as the ***star subgraph spanned by*** r ***in*** G*. Clearly, the graph G can be expressed as the union of its star subgraphs: $G = \cup_{r \in V_R} G[r]$.*

Ordered bipartite graphs are appropriate tools to represent and visualize (directed) hypergraphs. Visually, an ordered bipartite graph $G = (V_C, V_R, N_G)$ can be represented using boxes for vertices in V_C, ovals for vertices in V_R and integer labelled simple curves (edges) connecting boxes and ovals: if c and r are such that $c = N_G^i(r)$, then we have an edge with label i connecting the box labelled c to the oval labelled r (the labels of the vertices are depicted outside of the corresponding shape, and are used as visual marks only if it is necessary to make the diagram more clear).

We also need some additional graph theoretical notations. If $D = (V, E)$ is a DAG (Directed Acyclic Graph), then a **source (sink)** in D is any node v of D such that there is no entering (leaving) arc in (from) v.

A hypergraph is a pair $H = (V, \mathcal{P}(H))$, where V is a nonempty finite set (the vertices set of H), and $\mathcal{P}(H)$ is a family of nonempty subsets of V. Each member P of $\mathcal{P}(H)$ is a hyperedge of H.

The next two definitions, following the line of [1], introduce the concepts of support and Conceptual Graphs. A support is a structure that provides the background knowledge about the information to be represented in the Conceptual Graphs. It consists of a concept type hierarchy, a relation type hierarchy, a set of individual markers that refer to specific concepts and a generic marker, denoted by *, which refers to an unspecified concept.

Definition 2. (Support)
A support *is a 4-tuple $S = (T_C, T_R, \mathcal{I}, *)$ where:*

- T_C is a finite partially ordered set (poset), $(T_C, \leq)$, of concept types*, defining a type hierarchy (specialization hierarchy: $\forall x, y \in T_C$ $x \leq y$ means that x is a subtype of y) and which has a greatest element $\top_C$, the universal type.*

- T_R is a finite set of relation types *partitioned into k posets $(T_R^i, \leq)_{i=1,k}$ of relation types of arity i $(1 \leq i \leq k)$, where k is the maximum arity of a relation type in T_R. Each $(T_R^i, \leq)_{i=1,k}$ has a greatest element, the universal type $\top_{T_R^i}$.*

- $\mathcal{I}$ is a countable set of individual markers*, used to refer specific concepts.*

- $*$ is the generic marker *used to refer to an unspecified concept (having, however, a specified type).*

- The sets T_C, T_R, $\mathcal{I}$ and $\{*\}$ are mutually disjoint and $\mathcal{I} \cup \{*\}$ is partially ordered by $x \leq y$ iff $x = y$ or $y = *$.

A Conceptual Graph is a structure that depicts factual information about the background knowledge contained in its support. This information is presented in a visual manner as an ordered bipartite graph, whose nodes have been labelled with elements from the support. The label $\lambda(v)$ is inserted in the shape representing the node v.

Definition 3. (Conceptual graph) *A* (simple) Conceptual Graph *(*CG*) is a triple $SG = [S, G, \lambda]$, where:*

- *$S = (T_C, T_R, \mathcal{I}, *)$ is a support;*
- *$G = (V_C, V_R, N_G)$ is an ordered bipartite graph;*
- *λ is a labelling of the vertices of G with elements from the support S: $\forall r \in V_R,\ \lambda(r) \in T_R^{d_G(r)}$; $\forall c \in V_C,\ \lambda(c) \in T_C \times \big(\mathcal{I} \cup \{*\}\big)$.*

We introduce now the notion of a Conceptual Graph Assembly (CGA) as a structure which consists of a Conceptual Graph (CG) and a hypergraph on the CG's relation nodes. Each hyperedge defines a CG subgraph which is a member of the CGA.

Definition 4. (Conceptual Graph Assembly)
Let $S = (T_C, T_R, \mathcal{I}, *)$ be a support, $SG = [S, G, \lambda]$ a Conceptual Graph without isolated concept vertices, and let $H = (V_R, \mathcal{P}(H))$ be a hypergraph on the set V_R of all relation vertices of G.

The pair $CGA = (SG, H)$ is called a **Conceptual Graph Assembly***. The* ***members*** *of $CGA = (SG, H)$ are the Conceptual Graphs $SG_P = (S, G[P], \lambda_P)$, where $P \in \mathcal{P}(H)$ is any hyperedge of H, $G[P]$ is the subgraph of G spanned by the hyperedge P, and λ_P is the restriction of λ on the set of vertices of $G[P]$.*

Note that any simple Conceptual Graph SG (without isolated concept vertices) can be viewed as a Conceptual Graph Assembly with a single member, by taking H as a hypergraph with a single hyperedge $\mathcal{P}(H) = (V_R)$, containing all relation vertices of G.

Each hyperedge can be considered as a given world; each CG member of a CGA thus provides information available in this world. The hypergraph H can be given explicitly or implicitly, by a combinatorial property of its hyperedges. The latter case will be developed in Section 4.

Explicitly, the hypergraph H can be represented as a bipartite graph with one node class V_R, and the other (disjoint) class having a node v_P for each hyperedge $P \in \mathcal{P}(H)$, connected by edges to the relation nodes belonging to P. In the visual representation of the Conceptual Graph SG, this new set of vertices, V_P, representing the members of the CGA, can be designated as diamonds. This was illustrated in the example of figure 2.

From a knowledge representation point of view, this tripartite graph structure has the property that the information is well organized in order to facilitate reasoning (inferences) and, at the same time, presents itself as a visual medium of expression.

CGs are provided with logical semantics; more precisely, an operator θ is considered, which assigns to every support S a set of FOL formulas $\theta(S)$ and maps each simple Conceptual Graph G to a conjunctive, existential closed FOL formula $\theta(G)$. θ can be the well-known Sowa's operator Φ, or a variant of it Ψ discussed in [3]. A logical semantics of CGAs can be defined as follows.

Definition 5. (CGA logical semantics)
Let θ be a logical semantics for CGs, and $CGA = (SG, H)$ a Conceptual Graph Assembly. Then, $\theta(CGA)$ is the disjunction of the formulas assigned by θ to the members of CGA, that is,

$$\theta(CGA) = \vee_{P \in \mathcal{P}(H)} \theta(SG_P).$$

For example, consider the CGA $CGA_{grouping}$ described in Figure 5 and $\theta = \Phi$. Then, $\Phi(CGA_{grouping}) = [Color(yellow) \wedge Color(brown) \wedge hides(brown, yellow)] \vee [Color(yellow) \wedge Actor(consumer) \wedge Texture(glutinous) \wedge wanted(yellow, consumer) \wedge rejected(glutinous, consumer)]$.

Projection [12] is the fundamental operation on simple Conceptual Graphs since it can be used to define a preorder on the set of CGs based on the same support. If $SG = (G, \lambda_G)$ and $SF = (F, \lambda_F)$ are two CG's defined on the same support S, then a *projection from SG to SF* is a mapping $\Pi : V_C(G) \cup V_R(G) \rightarrow V_C(F) \cup V_R(F)$ such that

- $\Pi(V_C(G)) \subseteq V_C(F)$ and $\Pi(V_R(G)) \subseteq V_R(F)$;
- $\forall c \in V_C(G),\ \forall r \in V_R(G)$ if $c = N^i_G(r)$ then $\Pi(c) = N^i_F(\Pi(r))$
- $\forall v \in V_C(G) \cup V_R(G)\ \lambda_G(v) \geq \lambda_F(\Pi(v))$.

If there is a projection from SG to SF then *SG subsumes SF*, which is denoted $SG \geq SF$. This *subsumption relation* is a preorder on the set of all CG's defined on the same support. Subsumption checking is an NP-complete problem [1].

The notion of projection can be extended to CGAs. In the next definition we consider three forms of projection (weak, mild and strong) under two scenarios: projecting a CGA to a CG or projecting a CGA to a CGA. Intuitively we need different projection mechanisms to account for the very nature of Conceptual Graph Assemblies: commensalism. More precisely when trying to project a Conceptual Graph Assembly in a simple conceptual graph we need to distinguish from the case when the information encoded in the simple conceptual graph is contained in at least one world, all of worlds, or the conceptual graph itself represented by the Conceptual Graph Assembly. Similarly when projecting two Conceptual Graph Assemblies (which is a generalization of the previous case) we have to consider the same three possible situations.

Definition 6. (CGA Projection)
I. Let $CGA^1 = (SG^1, H^1)$ be a CGA and SG^2 a CG. Then
- **weak projection :** *$CGA^1 \geq_w SG^2$ if there is $P^1 \in \mathcal{P}(H^1)$ such that $SG^1_{P^1} \geq SG^2$.*
- **mild projection :** *$CGA^1 \geq_m SG^2$ if $SG^1_{P^1} \geq SG^2$ for each $P^1 \in \mathcal{P}(H^1)$.*
- **strong projection :** *$CGA^1 \geq_s SG^2$ if $SG^1 \geq SG^2$.*

II. Let $CGA^1 = (SG^1, H^1)$ and $CGA^2 = (SG^2, H^2)$ be two CGAs . Then
- **weak projection :** *$CGA^1 \geq_w CGA^2$ if there are $P^1 \in \mathcal{P}(H^1)$ and $P^2 \in \mathcal{P}(H^2)$such that $SG^1_{P^1} \geq SG^2_{P^2}$.*
- **mild projection :** *$CGA^1 \geq_m CGA^2$ if for each $P^1 \in \mathcal{P}(H^1)$ there is $P^2 \in \mathcal{P}(H^2)$ such that $SG^1_{P^1} \geq SG^2_{P^2}$.*

- **strong projection :** *$CGA^1 \geq_s SG^2$ if there is a projection Π from SG^1 to SG^2 such that the restriction of Π to the relation vertices of SG^1 is a homomorphism from H^1 to H^2.*

Note that in the case when $CGA^1 = (SG^1, H^1)$ is a simple conceptual graph (i.e. it contains a single member), weak, mild and strong projection are identical.

The following theorem can be easily deduced from the above definitions and further explains the need for different combinatorial degrees of subsumption. Note that strong and mild projection can give extra information with regard to the associated CGA logical semantics. When defined from a CGA to a CG, weak projection preserves the soundness and completeness of CG projection; when defined on two CGAs, it only preserves the soundness.

Theorem 1. *I. Let $CGA^1 = (SG^1, H^1)$ be a CGA and SG^2 a SCG. Then the following implications hold:*

$$CGA^1 \geq_s SG^2 \;\Rightarrow\; CGA^1 \geq_m SG^2 \;\Rightarrow\; CGA^1 \geq_w SG^2.$$

Furthermore, if θ is a logical semantics for CGs such that SG projection is sound and complete with respect to θ then

$$CGA^1 \geq_w SG^2 \;\Leftrightarrow\; \theta(S), \theta(SG^2) \models \theta(CGA^1).$$

II. Let $CGA^1 = (SG^1, H^1)$ and $CGA^2 = (SG^2, H^2)$ be two CGAs. Then the following implications hold:

$$CGA^1 \geq_s CGA^2 \;\Rightarrow\; CGA^1 \geq_m CGA^2 \;\Rightarrow\; CGA^1 \geq_w CGA^2.$$

If θ is a logical semantics for CGs such that SG projection is sound and complete with respect to θ then

$$CGA^1 \geq_w CGA^2 \;\Leftrightarrow$$

$$\textit{there is } P^2 \in \mathcal{P}(H^2) \textit{ s.t. } \theta(S), \theta(SG^2_{P^2}) \models \theta(CGA^1).$$

Proof
Part I. Suppose that $CGA^1 \geq_s SG^2$. It follows that $\Pi_{G^1 \to G^2} \neq \emptyset$. Let $\pi \in \Pi_{G^1 \to G^2}$ and $P^1 \in \mathcal{P}(H^1)$. π_1, the restriction of π to the vertices of $[P^1]_{G^1}$, is a projection from $SG^1_{P^1}$ to SG^2. Therefore $SG^1_{P^1} \geq SG^2$ for each $P^1 \in \mathcal{P}(H^1)$, that is $CGA^1 \geq_m SG^2$. The implication $CGA^1 \geq_m SG^2 \;\Rightarrow\; CGA^1 \geq_w SG^2$ is obvious by the definition of CGA projection.

Let θ be a logical semantics for CGs such that SG projection is sound and complete with respect to θ.

If $CGA^1 \geq_w SG^2$, then there is $P^1 \in \mathcal{P}(H^1)$ such that $SG^1_{P^1} \geq SG^2$. By the soundness of θ, we have $\theta(S), \theta(SG^2) \models \theta(SG^1_{P^1})$. Now, by the definition of CGA semantics, $\theta(SG^1_{P^1}) \models \theta(CGA^1)$, and therefore $\theta(S), \theta(SG^2) \models \theta(CGA^1)$.

If $\theta(S), \theta(SG^2) \models \theta(CGA^1)$, it follows that there is a term in the disjunction $\theta(CGA^1)$, say $\theta(SG^1_{P^1})$, where $P^1 \in \mathcal{P}(H^1)$, such that $\theta(S), \theta(SG^2) \models \theta(SG^1_{P^1})$. By the completeness of θ, we obtain that $SG^1_{P^1} \geq SG^2$. We have obtained that there is $P^1 \in \mathcal{P}(H^1)$ such that $SG^1_{P^1} \geq SG^2$ and, by the definition of CGA projection, $CGA^1 \geq_w SG^2$ holds.

Part II. Suppose that $CGA^1 \geq_s CGA^2$. It follows, by the definition of CGA projection, that there is π a projection from SG^1 to SG^2 such that $P^2 = \pi(P^1) \in \mathcal{P}(H^2)$, for each $P^1 \in \mathcal{P}(H^1)$. Obviously, $SG^1_{P^1} \geq SG^2_{P^2}$. Therefore, for each $P^1 \in \mathcal{P}(H^1)$ there is $P^2 \in \mathcal{P}(H^2)$ such that $SG^1_{P^1} \geq SG^2_{P^2}$, that is, $CGA^1 \geq_m CGA^2$.

If $CGA^1 \geq_w CGA^2$, then there is $P^1 \in \mathcal{P}(H^1)$ and $P^2 \in \mathcal{P}(H^2)$ such that $SG^1_{P^1} \geq SG^2_{P^2}$. By the soundness of θ, we have $\theta(S), \theta(SG^2) \models \theta(SG^1_{P^1})$. Now, by the definition of CGA logical semantics, $\theta(SG^1_{P^1}) \models \theta(CGA^1)$, and therefore $\theta(S), \theta(SG^2) \models \theta(CGA^1)$.

Conversely, if there is $P^2 \in \mathcal{P}(H^2)$ s.t. $\theta(S), \theta(SG^2_{P^2}) \models \theta(CGA^1)$, then it follows that there is a term in the disjunction $\theta(CGA^1)$, say $\theta(SG^1_{P^1})$, where $P^1 \in \mathcal{P}(H^1)$, such that $\theta(S), \theta(SG^2_{P^2}) \models \theta(SG^1_{P^1})$. By the completeness of θ, we obtain that $SG^1_{P^1} \geq SG^2_{P^2}$. Therefore, we have obtained that $CGA^1 \geq_w CGA^2$.

4 Conceptual Modelling Using CGAs

As mentioned in the previous section, the hypergraph H defined on the Conceptual Graph Assemblies relation nodes can be given *explicitly* (a list of the hyperedges of H is provided) or *implicitly*. This section details the latter technique and shows how CGAs can be effectively used for knowledge modelling.

H is given by specifying a property of its hyperedges. In this way, it is possible to represent, in a succinct manner, an exponential number of members in the CGA. However, if it is necessary, the explicit list of the hyperedges can be generated. Let us give some interesting ways to specify the above property.

(Di)graphs. On the set V_R of all relation vertices of the CG G, a graph (or digraph) HG is provided. The edges (or directed edges) of the graph HG express some links between their extremities. $\mathcal{P}(H)$ is described as a usual family of subsets of the vertices set of HG having graph theoretical significance.

For example, if D is an arbitrary DAG on the set V_R of all relation vertices of G, we can take $\mathcal{P}(H)$ as the family of vertices of all paths in D starting from a source and ending in a sink. The acyclicity condition assures that each path starting in a source must reach a sink. If it is necessary, a dummy source and a dummy sink are added in order to increase the visual quality of the digraphs considered (this fictive nodes are not considered when the hyperedges of H are constructed).

Another example can be obtained if we consider a graph HG on the set V_R of all relation vertices of G with the set of edges expressing a compatibility relation. For example, an edge $\{v_{r^1}, v_{r^2}\} \in E(HG)$ means that the facts expressed by v_{r^1} and v_{r^2} in the CG G can be considered in the same time in order to describe a complex factual information. Taking $\mathcal{P}(H)$ as the family of vertices of all maximal (w.r.t. set inclusion) cliques in HG, we obtain a CGA with an exponential number of members, which could be an elegant and efficient representational mechanism.

Conceptual Grouping. Let $S = (T_C, T_R, \mathcal{I}, *)$ be a support, $SG = [S, G, \lambda]$ a Conceptual Graph without isolated concept vertices and $CGA^1 = (SG, H^1)$ a CGA. Let $TH \subseteq T_C$ be a given set of *threshold concept types* and let $V_{TH} \subseteq V_C$ the set of all concept vertices v_c of the graph G, with the property that if $\lambda(v_c) = (type_{v_c}, ref_{v_c})$ then $\exists t \in TH$ such that $type_{v_c} \geq t$ (V_{TH} contains the vertices of G designating objects having the type "above" the prescribed threshold TH).

Taking $\mathcal{P}(H)$ as the family of all maximal (w.r.t. inclusion) subsets P of the members P^1 of CGA^1 such that $N_G(P) \subseteq V_{TH}$, we obtain a new CGA whose members describe only the facts about objects having a type above the threshold set TH in the hierarchy given by S.

For example, starting from the CGA of figure 2, with $TH = \{$*Food product, Color, Texture, Person*$\}$, we obtain the new CGA shown in figure 5. This CGA contains two members P_1 and P_2. P_1 indicates that the brown color hides the yellow one. P_2 indicates that the yellow color is wanted by the consumer and that the glutinous texture is rejected by the consumer. The interest of such a transformation can be, for instance, to determine parts of the knowledge base that can be easily understood by a wide public, due to the non-specific vocabulary used in the concepts.

Transversal Methods. Let $CGA^1 = (SG, H^1)$ be a CGA. Taking $\mathcal{P}(H)$ as the family of all subsets P of V_R with the property that $P \cap P^1 \neq \emptyset$, for each $P^1 \in \mathcal{P}(H^1)$, we obtain a new CGA $CGA = (SG, H)$ with interesting combinatorial connections with the first one. For example, let us suppose that $CGA^1 = (SG, H^1)$ satisfies the property that $\forall P^1 \in \mathcal{P}(H^1)$ and $\forall P^2 \in \mathcal{P}(H^1)$, if $P^1 \subseteq P^2$ then $P^1 = P^2$; if we take in $\mathcal{P}(H)$ only minimal transversal (that is minimal subsets, w.r.t. sets inclusion, of V_R intersecting all members of H^1) then

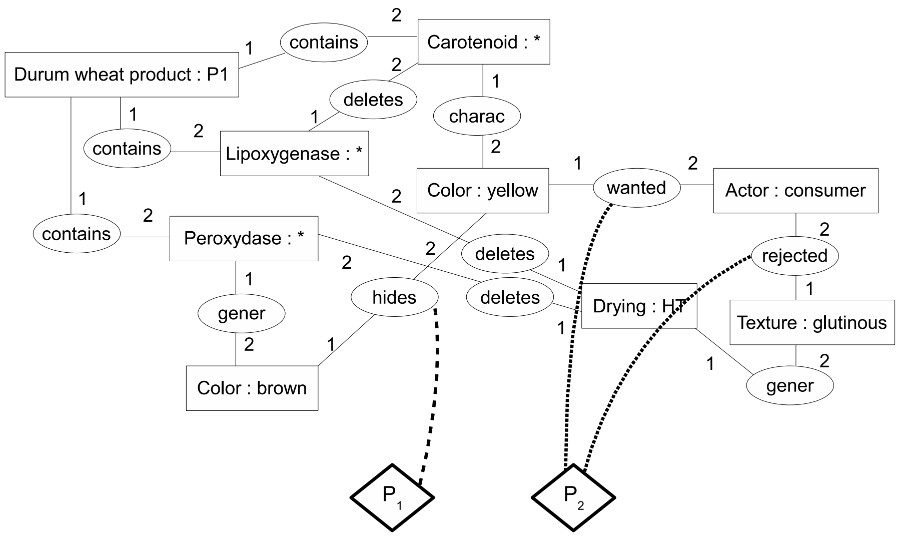


Fig. 5. The new obtained CGA

the hypergraph H has the property that its minimal transversals are precisely the members of the initial hypergraph H^1.

Another example in this category comes from the integration framework described in [4]. Let $SG^i = [S^i, G^i, \lambda^i]$, $i \in [0, n]$, be a set of $n+1$ CGs. For each relation node $r_j^0 \in V_R^0 = \{r_1^0, \ldots, r_m^0\}$, a set $\mathbb{R}(r_j^0)$ of triples is provided. Each such triple, $(i, A, w) \in \mathbb{R}(r_j^0)$ specifies a *rewriting rule* of r_j^0 in SG^i: r_j^0 is translated in the spanning subgraph $[A]_{G^i}$, and the $k = deg_{G^0}(r_j^0)$ neighbors of r_j^0 in G^0 are represented by the sequence $w = w^1, \ldots, w^k$ of concept nodes in $[A]_{G^i}$: $N_{G^0}^1(r_j^0)$ is represented by w^1, ..., $N_{G^0}^k(r_j^0)$ is represented by w^k.

Now, for each transversal T of the hypergraph $\mathbb{R} = (\mathbb{R}(r_1^0), \ldots, \mathbb{R}(r_m^0))$, a hyperedge is added to the hypergraph $\mathcal{T}^i$ by considering the union of the relation nodes sets of T contained in V_R^i. In this way, n CGAs, $CGA^i = (SG^i, \mathcal{T}^i)$, are obtained.

Assisting Reasoning. Let $SG = [S, G, \lambda_G]$ and $SQ = [S, Q, \lambda_Q]$ be two CGs defined on the same support S such that $SQ \geq SG$.

If $\mathbf{\Pi}_{Q \to G} = \{\pi | \pi$ is a projection from SQ to $SG\}$, then we can consider $Occ(Q, G) = (\pi(V_R^Q) | \pi \in \mathbf{\Pi}_{Q \to G})$. Taking $H = (V_R^G, Occ(Q, G))$ we obtain a CGA $CGA = (SG, H)$ which gives all the occurences of the query SQ in SG. For some usual query SQ this CGA can be pre-computed in order to have fast response time. With the same goal of efficiency, the following CGA can be considered.

Let $SG = [S, G, \lambda]$ be a Conceptual Graph and $\mathcal{M}$ a model for the support $S = (T_C, T_R, \mathcal{I}, *)$. Suppose that $\mathcal{M} \not\models SG$ and let us consider the CGA $CGA = (SG, H)$, $H = (V_R^G, \mathcal{P}(H))$, where $\mathcal{P}(H) = (I | I \subset V_R^G$ and $\mathcal{M} \models [I]_G)$. It is easy to see that H is an *independence system* on V_R^G, that is, if $I \in \mathcal{P}(H)$ and $I_1 \subseteq I$, then $I_1 \in \mathcal{P}(H)$.

5 Conclusions

In this paper we proposed a semantically sound syntactic extension to Conceptual Graphs: Conceptual Graph Assemblies (CGAs), and defined several reasoning mechanisms, based on the projection operation. We showed that CGAs provide increased representational power. We proposed several modelling scenarios and illustrated through an example in the agri-food domain the applicability of this extension in practice, in particular for the representation of multiple viewpoints on the same situation.

Conceptual Graph Assemblies are a flexible, versatile way of representing interrelated facts, concurrent events or possible scenarios. In future work we plan to explore two directions of modelling with CGAs: modelling temporal information, by attaching a temporal value to the relation nodes of the conceptual graph prior to defining the CGA by the means of this "stamp" value; modelling multi-viewpoints reasoning such as conflict detection – that can be viewed e.g. as the projection of a negative constraint in the conceptual graph represented by a CGA –, and resolution proposals through argumentation and decision methods.

We believe these are promising directions of work which will further demonstrate CGAs applicability.

References

1. Chein, M., Mugnier, M.-L.: Conceptual graphs: Fundamental notions. Revue d'Intelligence Artificielle 6(4), 365–406 (1992)
2. Chein, M., Mugnier, M.-L.: Positive nested conceptual graphs. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 95–109. Springer, Heidelberg (1997)
3. Chein, M., Mugnier, M.-L., Simonet, G.: Nested graphs: A graph-based knowledge representation model with FOL semantics. In: Proc. of the 6th Int'l Conf. on the Principles of Knowl. Repres. and Reasoning (KR 1998), pp. 524–535. Morgan Kaufmann, San Francisco (1998)
4. Croitoru, M., Compatangelo, E.: Hierarchical knowledge-oriented specification for information integration. In: Research and Development in Intelligent Systems XXII - Proc. of the 25th Int'l Conf. of the Brit. Comp. Soc. Specialist Group on Artif. Intell. (AI 2005), pp. 60–73. Springer, Heidelberg (2005)
5. Croitoru, M., Compatangelo, E.: Conceptual graph assemblies. In: Hitzler, P., Scharfe, H., Ohrstrom, P. (eds.) Contributions to ICCS 2006, 14th International Conference on Conceptual Structures, pp. 15–28. Aalborg University Press (2006)
6. Croitoru, M., Compatangelo, E.: Extending conceptual graphs for representing partial knowledge. In: The 7th IJCAI International Workshop on Nonmonotonic Reasoning, Action and Change (2007)
7. Esch, J.: Contexts and Concepts, Abstraction Duals. In: Tepfenhart, W.M., Dick, J.P., Sowa, J.F. (eds.) ICCS 1994. LNCS, vol. 835, pp. 175–184. Springer, Heidelberg (1994)
8. Mineau, G.W., Gerbé, O.: Contexts: A Formal Definition of Worlds of Assertions. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 80–94. Springer, Heidelberg (1997)
9. Puder, A., et al.: Service Trading Using Conceptual Structures. In: 3rd International Conference on Conceptual Structures, University of California, pp. 59–73. Springer, Heidelberg (1995)
10. Ribière, M., Dieng-Kuntz, R.: A viewpoint model for cooperative building of an ontology. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS, vol. 2393, pp. 220–234. Springer, Heidelberg (2002)
11. Sowa, J.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co. (2000)
12. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
13. Thomopoulos, R., Mugnier, M., Leclère, M.: Mapping contexts to vocabularies to represent intentions. In: ECAI 2006 Workshop on Contexts and Ontologies: Theory, Practice and Applications (c&o-2006), pp. 44–46 (2006)

Access Policy Design Supported by FCA Methods[⋆]

Frithjof Dau and Martin Knechtel

SAP AG, SAP Research CEC Dresden, Germany
{frithjof.dau,martin.knechtel}@sap.com

Abstract. Role Based Access Control (RBAC) is a methodology for providing users in an IT system specific permissions like *write* or *read* to users. It abstracts from specific users and binds permissions to user roles. Similarly, one can abstract from specific documents and bind permission to document types.

In this paper, we apply Description Logics (DLs) to formalize RBAC. We provide a thorough discussion on different possible interpretations of RBAC matrices and how DLs can be used to capture the RBAC constraints. We show moreover that with DLs, we can express more intended constraints than it can be done in the common RBAC approach, thus proving the benefit of using DLs in the RBAC setting. For deriving additional constraints, we introduce a strict methodology, based on attribute exploration method known from Formal Concept Analysis. The attribute exploration allows to systematically finding unintended implications and to deriving constraints and making them explicit. Finally, we apply our approach to a real-life example.

1 Introduction

1.1 Access Control Matrix, RBAC, Description Logics

An access control matrix M, first introduced by Lampson [1], is an abstract formal computer security model which consists of a set of objects O, subjects S and actions A. Each matrix row represents a subject and each column represents an object. Each matrix element $M[s, o] \subseteq A$ is the set of actions which a subject $s \in S$ is allowed to perform on object $o \in O$. For any type of access control system it can model the static access permissions, ignoring further definitions of a policy like rules and dynamic behavior in time. One type of access control system is Role Based Access Control (RBAC) [2], which abstracts from specific users and binds permissions to user roles. The permission set of a specific user is the union of all permissions of the roles he is assigned to. Flat RBAC comprises a set of users U, a set of roles R and a set of permissions P. Users are assigned to roles via a relation $UA \subseteq U \times R$, and permissions are assigned to roles via

⋆ This research was funded by the German Federal Ministry of Economics and Technology under the promotional reference 01MQ07012 and the German Federal Ministry of Education and Research under grant number 01IA08001A.

a relation $PA \subseteq R \times P$. One extension to this simple model is Hierarchical RBAC, which introduces a hierarchy of user roles for permission inheritance. The partial order $\geq_R \subseteq R \times R$ defines the role dominance relation. If a senior role $r_s \in R$ dominates a junior role $r_j \in R$, it inherits all permission from it (i.e. $\forall p \in P : (r_j, p) \in PA \wedge (r_s, r_j) \in \geq_R \rightarrow (r_s, p) \in PA$).

The relationship between RBAC and other access control systems which can be modeled with the access control matrix has been elaborated in [3]. For our paper we straightforwardly interpret the set of user roles as the set of subjects ($S = R$) and we define permissions as a set of tuples of action and object ($P \subseteq A \times O$). We call this an *RBAC matrix.* An *RBAC policy* can not completely be described by an RBAC matrix, since it contains further constraints, e.g. rules, dynamic behavior in time, user role hierarchy, implications between the allowed actions etc. Objects do not need to be individuals but could also be abstract groups. As an example for the RBAC matrix, each row represents a user role and each column a document type.

Description Logic (DL) [4] systems are formal knowledge representation and reasoning systems which provide inference services that deduce implicit knowledge from the explicitly represented knowledge. For these inference services to be feasible the underlying inference problems must at least be decidable, since DL is a decidable fragment of First Order Logic this is provided. Some proposals are available to model an RBAC policy with a DL ontology, in order to reduce authorization decision to standard reasoning services. Some of these approaches contained modeling flaws which we discussed in [5] and [6].

1.2 Our Contributions

Our paper discusses how FCA can be applied in order to provide services to a security policy designer. In our approach, a role-based access control matrix is formalized as triadic formal context $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}} := (\mathsf{R}, \mathsf{D}, \mathsf{P}, I)$, with a set R of $\underline{\text{role names}}$, a set D of $\underline{\text{document type names}}$ and a set P of $\underline{\text{permission names}}$.

Although it is quite straightforward to use an access control matrix as a model for RBAC, the interpretation of the matrix is not a priori clear. The paper contains a *discussion of three interpretations of an RBAC matrix.*

Up to now the DL modeling was done with ad hoc approaches. In [6] we discussed a flawed approach and proposed a reworked version. In this paper, we show how in each of the three possible interpretations, the information contained in the RBAC matrix is correctly modeled by DL general concept inclusions (GCIs). The used DL is $\mathcal{ALEROI}$ which is a subset of $\mathcal{SROIQ}$, the basis for the coming W3C OWL 2 standard. This DL is required to simulate the concept product expressions $\mathsf{R}^\mathcal{I} \times \mathsf{D}^\mathcal{I} \subseteq \mathsf{P}^\mathcal{I}$ and $(\mathsf{R}^\mathcal{I} \times \mathsf{D}^\mathcal{I}) \cap \mathsf{P}^\mathcal{I} = \emptyset$.

Using DLs, it will turn out that we can model additional constraints which are intended by the RBAC engineer, but which cannot be modd in role-based access control matrix alone. For example for a review process, it is not allowed that the same person who writes a document also approves it. The inclusion of axiom $\mathit{mayWrite} \sqcap \mathit{mayApprove} \sqsubseteq \bot$ in the DL model defines that both permissions are disjoint. The DL model allows *consistency checks of the RBAC* *policy* with given additional restrictions. Both the higher expressiveness of a DL based modelling approach and the consistency check clearly show the benefit of using DLs for RBAC.

Generally, one can say that ontology editors provide reasoning facilities, where for example the consistency of an DL knowledge base can be checked. Roughly speaking, ontology editors support checking the *soundness* of a DL knowledge base. In this paper, we do not only target the soundness of the DL formalization of an RBAC matrix, but also the *completeness* (compare to [7]). We introduce a *strict methodology*, based on the attribute exploration method of FCA, for deriving additional constraints in RBAC setting. Our methodology derives such constraints not explicitly contained in the RBAC matrix in a computer supported dialog with the RBAC engineer. This helps the engineer to create the policy as DL model based on the matrix.

The paper is structured as follows: In Sec. 2, all relevant notions are formally defined, and the running example we will use is introduced. Moreover, in this section the tree possible interpretations of an RBAC matrix are discussed. In Sec. 3, we show how the information of an RBAC matrix can be expressed by means of DL GCIs. In Sec. 4, we thoroughly discuss how attribute exploration can be used in order to obtain additional constraints from the RBAC matrix, and how these constraints are then modeled with DL GCIs. In Sec. 4, we apply our approach to a real-life example. Finally, in Sec. 6 we summarize this paper and discuss future research.

2 Basic Definitions

In this section, all relevant notions which will be used in this paper are formally defined, and our working example is introduced.

Vocabulary: As already mentioned, our starting point is a three-dimensional matrix, where the three dimensions are the *roles*, *document types* and *permissions*. In order not to mix up user roles and DL roles, with "role" we always refer to a user role, whereas we use the OWL terminology "property" for a DL role. In our ongoing formalization, both roles and document types will be modeled as concept names of a (appropriately chosen) DL, and each permission will be modeled as a property between roles and document types. That is, we consider a DL vocabulary which consists of a set **R** of role names, a set **D** of document type names, and of a set **P** of permission names. The vocabulary of these names will be denoted **V**. We will use a working example with specific roles, document types and permissions. We consider the permissions mayApprove, mayWrite and mayOpen, which are abbreviated by MA, MW and MO, respectively. The document types are user manual, marketing document, customer contract document, term of use document, installation guide, external technical interface document, design document and rating entry, abbreviated by UM, MD, CCD, ToUD, IG, ETID, DD, RE. The roles are marketplace visitor, service consumer, software development engineer, service vendor, legal department employee, service provider, marketing employee, technical editor and customer service

Table 1. Our example RBAC matrix

	mayOpen								mayWrite								mayApprove							
	UM	MD	CCD	ToUD	IG	ETID	DD	RE	UM	MD	CCD	ToUD	IG	ETID	DD	RE	UM	MD	CCD	ToUD	IG	ETID	DD	RE
MV		×		×				×																
SC	×	×	×	×		×		×								×			×	×				
SDE	×	×		×	×	×	×	×	×				×	×	×			×						
SV	×	×	×	×	×	×	×	×									×	×	×	×	×	×	×	
LDE	×	×	×	×	×	×	×	×			×	×												
SP	×	×		×	×	×		×																
ME	×	×		×	×	×	×	×		×														
TE	×	×		×	×	×	×	×	×				×	×	×									
CSE	×	×	×	×	×	×	×	×			×													

employee, abbreviated by MV, SC, SDE, SV, LDE, SP, ME, TE and CSE. This example stems from the research project Theseus/Processus from a scenario where documents describe aspects of services offered in the Internet of Services. The documents are accessible by different roles with different permissions.

Formal Contexts: The three-dimensional matrix of roles, document types and permissions is formalized as a triadic formal context $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}} := (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$. The example we use in this paper is provided in Tab. 1.

Our aim is to conduct an attribute exploration in order to explore dependencies between different roles, different document types, or different permissions. As attribute exploration is applied to dyadic contexts, we have do derive such contexts from the given triadic context. This can be done in several ways.

1. First, we can consider "slices" of the triadic context. For our goal, it is most useful to consider the "slice" for each $\mathsf{P} \in \mathbf{P}$. That is, for a given $\mathsf{P} \in \mathbf{P}$, we consider $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}} := (\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$, where $(\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} :\Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$.
2. Next, we can consider the dyadic contexts, where the set of attributes is one of the sets $\mathbf{R}$, $\mathbf{D}$, $\mathbf{P}$, and the set of objects is the cross-product of the remaining two sets. E.g. we can consider the context $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}} := (\mathbf{R} \times \mathbf{P}, \mathbf{D}, I^{\mathbf{R}\times\mathbf{P},\mathbf{D}})$ with $((\mathsf{R}, \mathsf{P}), \mathsf{D}) \in I^{\mathbf{R}\times\mathbf{P},\mathbf{D}} \Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. This is a straight-forward transformation. To simplify notations, we will denote the incidence relation again by I, thus writing $(\mathbf{R} \times \mathbf{D}, \mathbf{P}, I)$. We can construct six dyadic contexts this way, namely $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$, $\mathbb{K}_{\mathbf{P}\times\mathbf{R},\mathbf{D}}$, $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$ and the respective named variants with identical cross table $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$, $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$, $\mathbb{K}_{\mathbf{P}\times\mathbf{D},\mathbf{R}}$.
3. For a given context $\mathbb{K} := (G, M, I)$, when attribute exploration is conducted, sometimes it is sensible to add an additional attribute $\perp$ (which satisfies $\neg\exists g \in G : (g, \perp) \in I$) to M. We use $\mathbb{K}_{\perp} := (G, M \cup \{\perp\}, I)$ to denote this context (again, we simply 'reuse' the symbol 'I' for the incidence relation). In our example no agent will be allowed to write and approve the same document, thus $\mathsf{mayApprove} \wedge \mathsf{mayWrite} \rightarrow \perp$.

As each of the formal context only deals with *names* for roles, document types, and permissions, but not with instances of these names (in some DL interpretations, see below), all these formal contexts are called $\mathcal{T}$-context.

Interpretations: The DL-interpretations for RBAC matrices are straightforwardly defined: For our setting, a DL-interpretation for $\mathbf{V}$ is a pair $(\Delta, \cdot^{\mathcal{I}})$ with a non-empty universe (of discourse) Δ and an interpretation function $\cdot^{\mathcal{I}}$ which satisfies:

- $\mathsf{R}^{\mathcal{I}} \subseteq \Delta$ for each $\mathsf{R} \in \mathbf{R}$. Moreover, we set $\mathbf{R}^{\mathcal{I}} := \bigcup_{\mathsf{R}\in\mathbf{R}} \mathsf{R}^{\mathcal{I}}$. The elements $r \in \mathbf{R}^{\mathcal{I}}$ are called agents.
- $\mathsf{D}^{\mathcal{I}} \subseteq \Delta$ for each $\mathsf{D} \in \mathbf{D}$. Moreover, we set $\mathbf{D}^{\mathcal{I}} := \bigcup_{\mathsf{D}\in\mathbf{D}} \mathsf{D}^{\mathcal{I}}$.
- $\mathsf{P}^{\mathcal{I}} \subseteq \mathbf{R}^{\mathcal{I}} \times \mathbf{D}^{\mathcal{I}}$ for each $\mathsf{P} \in \mathbf{P}$
- $\mathbf{R}^{\mathcal{I}} \cap \mathbf{D}^{\mathcal{I}} = \emptyset$ (nothing is both agent and document)
- $\mathbf{R}^{\mathcal{I}} \cup \mathbf{D}^{\mathcal{I}} = \Delta$ (everything is either agent or document)

Note that the first two conditions are standard conditions for DL interpretations, whereas the last 3 condition are additional constraints.

Permissive, Prohibitive and Strict Interpretations: As each formal object and attribute of $(\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$ stands in fact for a whole class of agents resp. documents, it is not a priori clear what the semantics of the incidence relation I with respect to an interpretation $(\Delta, \cdot^{\mathcal{I}})$ is. So we have to clarify the meaning of I. First we might assume that a relationship $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$ means that *each* agent $r \in \mathsf{R}^{\mathcal{I}}$ has the permission $\mathsf{P}^{\mathcal{I}}$ for *each* document $d \in \mathsf{D}^{\mathcal{I}}$. So a cross in the cross-table of the context $(\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ grants permissions to agents on documents, and we can read from the context which permissions are *at least* granted to agents. Vice versa, we might assume that a missing relationship $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I$ means that *no* agent $r \in \mathsf{R}^{\mathcal{I}}$ has the permission $\mathsf{P}^{\mathcal{I}}$ for *any* document $d \in \mathsf{D}^{\mathcal{I}}$. So a missing cross in the cross-table of the context $(\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ prohibits that permissions are granted to agents, and we can read from the context which permissions are *at most* granted to agents. And finally, we could of course assume that both conditions hold. That is, we can read from the context which permissions are *precisely* granted to agents.

These three understandings lead to the notion of permissive, prohibitive and strict interpretations (with respect to the formal context) summarized in Tab. 2. They are formally defined as follows:

- An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called permissive (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$), and we write $(\Delta, \cdot^{\mathcal{I}}) \models_{+} (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$, iff. for all role names $\mathsf{R} \in \mathbf{R}$, all document

Table 2. Variants how to interpret a cross in the context

interpretation	cross	no cross
strict	permission for all individuals	prohibition for all individuals
permissive	permission for all individuals	permission for some individuals
prohibitive	permission for some individuals	prohibition for all individuals

type names $\mathsf{D} \in \mathbf{D}$ all permission names $\mathsf{P} \in \mathbf{P}$ we have:

$$(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I \quad \Longrightarrow \quad \forall r \in \mathsf{R}^{\mathcal{I}}\ \forall d \in \mathsf{D}^{\mathcal{I}} : (r, d) \in \mathsf{P}^{\mathcal{I}}$$

In other words, if $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$, we have $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$.

– An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called <u>prohibitive (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$)</u>, and we write $(\Delta, \cdot^{\mathcal{I}}) \models_{-} (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$, iff. for all role names $\mathsf{R} \in \mathbf{R}$, all document type names $\mathsf{D} \in \mathbf{D}$ all permission names $\mathsf{P} \in \mathbf{P}$ we have:

$$(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I \quad \Longrightarrow \quad \forall r \in \mathsf{R}^{\mathcal{I}}\ \forall d \in \mathsf{D}^{\mathcal{I}} : (r, d) \notin \mathsf{P}^{\mathcal{I}}$$

In other words, if $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I$, we have $(\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}}) \cap \mathsf{P}^{\mathcal{I}} = \emptyset$.

– An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called <u>strict (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$)</u>, iff. it is both permissive and prohibitive.

We say that we use the <u>permissive approach</u> (<u>prohibitive approach</u>, <u>strict approach</u>), if we assume that each interpretation is permissive (prohibitive, strict).

Instantiations of Contexts: As already said in the introduction, it will turn out that for running attribute exploration on the context, it is reasonable not to consider the <u>$\mathcal{T}$-context</u>, but contexts where on the side of the objects, roles are replaced by "real" users resp. document types are replaced by "real" documents. Essentially, instantiations of a context contain at least all rows of the given context, and there might be more rows, but these additional rows must be extensions of rows in the given context. These contexts are now introduced.

Let one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}} := (\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ $(\mathsf{P} \in \mathbf{P})$ be given. An <u>instantiation</u> of $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ is a context $(R, \mathbf{D}, J^{\mathsf{P}})$, where R is a set of agents such that

Table 3. The context $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$ and one possible instantiation

	UM	MD	CCD	ToUD	IG	ETID	DD	RE
MV								
SC								×
SDE	×				×	×	×	
SV								
LDE			×	×				
SP								
ME		×						
TE	×				×	×	×	
CSE			×					

	UM	MD	CCD	ToUD	IG	ETID	DD	RE
$agent_1$								
$agent_2$								×
$agent_3$	×				×	×	×	
$agent_4$								
$agent_5$			×	×				
$agent_6$								
$agent_7$		×						
$agent_8$	×				×	×	×	
$agent_9$			×					
$agent_{10}$			×	×	×			
$agent_{11}$	×		×		×	×	×	×
$agent_{12}$		×	×					

– $\forall \mathsf{R} \in \mathbf{R}\ \exists r \in R\ \forall \mathsf{D} \in \mathbf{D} : (\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} \Leftrightarrow (r, \mathsf{D}) \in J^{\mathsf{P}}$
– $\forall r \in R\ \exists \mathsf{R} \in \mathbf{R}\ \forall \mathsf{D} \in \mathbf{D} : (\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} \Rightarrow (r, \mathsf{D}) \in J^{\mathsf{P}}$

Such a context will be denoted $\mathbb{K}^{\mathsf{P}}_{R,\mathbf{D}}$. We define similarly the instantiations $\mathbb{K}_{R\times\mathbf{P},\mathbf{D}}$ of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$, and $\mathbb{K}_{\mathbf{P}\times R,\mathbf{D}}$ of $\mathbb{K}_{\mathbf{P}\times\mathbf{R},\mathbf{D}}$ (where again the role names are replaced by agents), as well as the instantiations $\mathbb{K}^{\mathsf{P}}_{D,\mathbf{R}}$ of $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ ($\mathsf{P} \in \mathbf{P}$), $\mathbb{K}_{D\times\mathbf{P},\mathbf{R}}$ of $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$, and $\mathbb{K}_{\mathbf{P}\times D,\mathbf{R}}$ of $\mathbb{K}_{\mathbf{P}\times\mathbf{D},\mathbf{R}}$ (where now the document type names are replaced by documents).

Instantiations of the contexts where the permissions are the attributes, i.e. instantiations $\mathbb{K}_{D\times R,\mathbf{P}}$ of $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$ (resp. $\mathbb{K}_{R\times D,\mathbf{P}}$ of $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$) are defined similarly (where on the side of the objects, both document type names and role names are replaced by "real" documents and "real" agents, respectively).

An example for an instantiation of $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$ is given in Tab. 3.

3 Expressing the Cross-Table by GCIs

In this section, we scrutinize how the information of the context $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ can be expressed by means of DLs. Besides the standard DL quantifications $\exists R.C$ (the set of entities which stand in relation R *to at least one* instance of C) and $\forall R.C$ (the set of entities which stand in relation R *only to* instances of C), we will use the non-standard constructor $\forall C.R$ (the set of entities which stand in relation R *to all* instances of C). This constructor can be expressed by means of negation of relations, as $\forall C.R$ is equivalent to $\forall\neg R.\neg C$ (see [8] for a thorough discussion of the constructor). Adding it to $\mathcal{ALC}$ still yields a decidable DL, but as this constructor is certainly non-standard, is it not supported by common DL reasoners.

For the permissive approach, we have to capture the condition $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$. The left expression is a *concept product*. It can not be expressed in $\mathcal{SHOIN}$(D), which is the underlying DL of OWL DL. In OWL 2.0, there does not exist a native language construct for the concept product, but Krötzsch, Rudolph, Hitzler provide in [9] a workaround to express it in OWL 2.0. Using the constructor $\forall C.R$, the condition $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$ can be expressed with the GCIs

$$\mathsf{R} \sqsubseteq \forall \mathsf{D}.\mathsf{P} \quad \text{(i.e. } \mathsf{R} \sqsubseteq \forall\neg\mathsf{P}.\neg\mathsf{D}) \qquad \text{and} \qquad \mathsf{D} \sqsubseteq \forall \mathsf{R}.\mathsf{P}^{-1} \quad \text{(i.e. } \mathsf{D} \sqsubseteq \forall\neg\mathsf{P}^{-1}.\neg\mathsf{R})$$

For the prohibitive approach, the condition $(\mathsf{R}^{\mathcal{I}} \times \mathsf{D}^{\mathcal{I}}) \cap \mathsf{P}^{\mathcal{I}} = \emptyset$ has to be captured. This can be expressed by the two GCIs

$$\mathsf{R} \sqsubseteq \forall \mathsf{P}.\neg\mathsf{D} \qquad \text{and} \qquad \mathsf{D} \sqsubseteq \forall \mathsf{P}^{-1}.\neg\mathsf{R}$$

Note that this condition is precisely the condition for the permissive approach, when we replace each permission P by its complement $\neg\mathsf{P}$. This duality principle will be discussed in the next section.

If we knew that $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is correct, and if we know which type of approach (permissive, prohibitive, strict) we use, then we can describe the information of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ by DL GCIs. We first set $\mathsf{R}_{\text{all}} := \bigsqcup_{\mathsf{R}\in\mathbf{R}} \mathsf{R}$ and $\mathsf{D}_{\text{all}} := \bigsqcup_{\mathsf{D}\in\mathbf{D}} \mathsf{D}$. Now we define the following knowledge base:

$KB_0 := \{\mathsf{R}_{\text{all}} \sqsubseteq \forall \mathsf{P}.\mathsf{D}_{\text{all}}\,, \mathsf{D}_{\text{all}} \sqsubseteq \forall \mathsf{P}^{-1}.\mathsf{R}_{\text{all}} \mid \mathsf{P} \in \mathbf{P}\} \cup \{\mathsf{R}_{\text{all}} \sqsubseteq \neg \mathsf{D}_{\text{all}}\} \cup \{\mathsf{R}_{\text{all}} \sqcup \mathsf{D}_{\text{all}} \equiv \top\}$

Obviously, a general DL-interpretation $(\Delta, \cdot^{\mathcal{I}})$ is a DL-interpretation of $\mathbf{V}$ iff. it satisfies KB_0. According to the chosen approach, we can now capture the information of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ as follows:

$$\begin{aligned} KB_+ &:= KB_0 \cup \{\mathsf{R} \sqsubseteq \forall \neg \mathsf{P}.\neg \mathsf{D}\,,\ \mathsf{D} \sqsubseteq \forall \neg \mathsf{P}^{-1}.\neg \mathsf{R} \mid (\mathsf{R},\mathsf{D},\mathsf{P}) \in I\} \\ KB_- &:= KB_0 \cup \{\mathsf{R} \sqsubseteq \forall \mathsf{P}.\neg \mathsf{D}\,,\ \mathsf{D} \sqsubseteq \forall \mathsf{P}^{-1}.\neg \mathsf{R} \mid (\mathsf{R},\mathsf{D},\mathsf{P}) \notin I\} \\ KB_\pm &:= KB_+ \cup KB_- \end{aligned}$$

Again, a DL-interpretation is obviously an permissive (prohibitive, strict) interpretation of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, if it satisfies KB_+ (KB_-, $KB_\pm$).

4 Using Attribute Exploration for RBAC Matrices

In this section, we discuss how attribute exploration can be utilized in order to create a DL knowledge base which captures as best as possible the dependencies between roles, documents, and permissions. It is crucial which approach (permissive, prohibitive, strict) we use, thus we first elaborate the differences between these approaches with respect to attribute exploration. In the second and third part of this section, we go into the details of an attribute exploration for instantiations of contexts in the permissive approach.

4.1 General Discussion

We first compare the permissive and the prohibitive approach. In the permissive approach, the crosses in a cross-table carry information, whereas missing crosses are not informative. In the prohibitive approach, the situation is converse: Missing crosses carry information, and crosses are not informative. Missing crosses in a relation correspond to crosses in the complement of the relation. Thus if we replace in the prohibitive approach the relations mayOpen,mayWrite and mayApprove by their complements $\mathsf{mayOpen}^c = \mathsf{mustNotOpen}$, $\mathsf{mayWrite}^c = \mathsf{mustNotWrite}$, $\mathsf{mayApprove}^c = \mathsf{mustNotApprove}$, we have a situation similar to the permissive approach. That is, we have the following duality principle: Any account to the permissive approach can be turned into an account to the prohibitive approach (and vice versa) by replacing each permission by its complement.[1] For this reason, we do not target the prohibitive approach in this paper.

We assume that the set of role names, document type names, and permission names is fixed. Conducting an attribute exploration on one of the $\mathcal{T}$-contexts seems for this reason to some extent pointless, as we cannot add new objects (counterexamples for implications which do not hold). We can still use attribute exploration in order to check that the information in $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is *correct*, but this

[1] But keep in mind that switching between the permissive and prohibitive approach requires changing the underlying DL-language, including the need for non-standard constructors in the permissive approach.

idea does not tap the full potential of attribute exploration and will not be carried out in this paper (we assume that the matrix $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ is correct). But notice that this check for correctness would have avoided the inconsistency between role hierarchy, DL model and access matrix discussed in [6]. Anyhow, we emphasized that in the formal context, the formal objects (the elements of $\mathbf{R}$) and attributes (the elements of $\mathbf{D}$) stand in turn for complete classes (of agents and documents). This can be used to apply attribute exploration to RBAC matrices. Assume we stick to the permissive approach. Assume moreover that we consider a permissive interpretation $(\Delta, \cdot^{\mathcal{I}})$ with respect to $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$. Then for a given permission $\mathsf{P} \in \mathbf{P}$, agent $r \in \mathsf{R}^{\mathcal{I}}$ for a role $\mathsf{R} \in \mathbf{R}$, and document $d \in \mathsf{D}^{\mathcal{I}}$ for a document type $\mathsf{D} \in \mathbf{D}$, we might have that r has permission P to d (i.e., $(r, d) \in \mathsf{P}^{\mathcal{I}}$), though we do not have $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. That is, it is sensible to run an attribute exploration on the *instantiations* of the $\mathcal{T}$-contexts. As we will see in the next section, with attribute exploration we can in fact infer constraints for the dependencies between roles, documents and permissions which are not captured by $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$.

In the strict approach, the situation is different. If we consider a strict interpretation $(\Delta, \cdot^{\mathcal{I}})$ with respect to $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$, then for a given permission $\mathsf{P} \in \mathbf{P}$, agent $r \in \mathsf{R}^{\mathcal{I}}$ and document $d \in \mathsf{D}^{\mathcal{I}}$, we have $(r, d) \in \mathsf{P}^{\mathcal{I}} \Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. That is, based on the given assumption that the sets of roles, documents and permissions are fixed, all possible constraints for the dependencies between these entities are already captured by $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$. This observation has two consequences: First, no DL formalization of the strict approach can extend the information of $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$, i.e., a DL formalization of $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ is somewhat pointless. Second, the instantiations of $\mathcal{T}$-context are nothing but the $\mathcal{T}$-context themselves (instantiations might duplicate some rows, but this is of course of no interest), thus conduction attribute exploration in the strict approach is pointless as well.

To summarize: As the permissive and prohibitive approach are mutually dual, and as addressing the strict approach with DLs or attribute exploration is pointless, it is sufficient that we here address only the permissive approach.

4.2 Attribute Exploration for Instantiations of $\mathcal{T}$-Contexts

In the last part we argued why we will run attribute exploration on instantiations of $\mathcal{T}$-contexts. Before doing so, we first have to discuss how implications in $\mathcal{T}$-contexts and their instantiations are read, and then we will scrutinize some peculiarities for applying attribute exploration in our setting. In fact, due to the fact that the objects and attributes of $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ stand for whole classes, the existing approaches for conducting attribute explorations on triadic contexts (e.g, [10]) cannot be applied to our framework.

Reading Implications. We consider the two contexts of Tab. 3. In both contexts, $\mathsf{term\ of\ use\ document} \rightarrow \mathsf{customer\ contract\ document}$ holds. For the $\mathcal{T}$-context $\mathbb{K}_{\mathsf{R},\mathsf{D}}^{\mathsf{mayWrite}}$, the objects are classes, thus this implication is read as follows:

> **$\mathcal{T}$-reading:** For each role we have that whenever every agent of that role may write all terms of use documents, then every agent of that role may write all customer contract documents as well.

For the instantiation of $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$, the objects are now instances instead of classes, thus we have a different reading of the implication. It is:

> **$\mathcal{I}$-reading:** Whenever every agent may write all terms of use documents, then the agent may write all customer contract documents as well.

Implications like this cannot be read from any $\mathcal{T}$-context, thus running attribute exploration on instantiations can indeed be used to obtain new knowledge.

Please note that none of the above readings conforms to the concept inclusion $\mathsf{term\ of\ use\ document} \sqsubseteq \mathsf{customer\ contract\ document}$. This is due to in both implications we quantify over *all* term of use documents and *all* customer contract documents. For the latter reading, we now show how it is correctly translated into a GCI. The implication means that for any permissive interpretation $(\Delta, \cdot^{\mathcal{I}})$, we have that $\forall r \in \mathbf{R}^{\mathcal{I}} : (\forall d \in \mathsf{ToUD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}} \to \forall d \in \mathsf{CCD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}})$ holds. This condition is now transformed into a GCI as follows:

$$\begin{aligned}
&\forall r \in \mathbf{R}^{\mathcal{I}} : \Big(\forall d \in \mathsf{ToUD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}} \to \forall d \in \mathsf{CCD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}}\Big)\\
\Longleftrightarrow\ &\forall r \in \mathbf{R}^{\mathcal{I}} : \big(r \in (\forall \mathsf{ToUD.MW})^{\mathcal{I}} \to r \in (\forall \mathsf{CCD.MW})^{\mathcal{I}}\big)\\
\Longleftrightarrow\ &(\Delta, \cdot^{\mathcal{I}}) \models \forall \mathsf{ToUD.MW} \sqsubseteq \forall \mathsf{CCD.MW}
\end{aligned}$$

(we have to emphasize that the direction "$\rightarrow$" of the last equivalence is only valid if we assume that $dom(\mathsf{MW}^{\mathcal{I}}) \subseteq \mathbf{R}^{\mathcal{I}}$ holds, but we assume that out interpretation satisfies KB_0, which models this additional condition).

In general, any implication of the form $\mathsf{D}_1 \wedge \ldots \wedge \mathsf{D}_{n-1} \to \mathsf{D}_n$ in an instantiation of one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ can be translated into the following GCI:

$$\forall \mathsf{D}_1.\mathsf{P} \sqcap \ldots \sqcap \forall \mathsf{D}_{n-1}.\mathsf{P} \sqsubseteq \forall \mathsf{D}_n.\mathsf{P}$$

Similarly, any implication of the form $\mathsf{R}_1 \wedge \ldots \wedge \mathsf{R}_{n-1} \to \mathsf{R}_n$ in an instantiation of one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ can be translated into the following GCI:

$$\forall \mathsf{R}_1.\mathsf{P}^- \sqcap \ldots \sqcap \forall \mathsf{R}_{n-1}.\mathsf{P}^- \sqsubseteq \forall \mathsf{R}_n.\mathsf{P}^-$$

If we consider an instantiation of a context where the attributes of the context are neither document type names nor role names, but instead permission names, the situation is different, as now the attributes do not stand for classes of instances, but for properties between instances. In Sec. 5.1, we consider a context $\mathbb{K}_{D\times R,\mathbf{P}}$. In this context, $\mathsf{mayWrite} \to \mathsf{mayOpen}$ holds. The reading of this implication is

> Whenever some agent has the permission to write some document, then this agent may open this document as well.

So we see that in this case, the implication can be translated to a simple inclusion axiom between properties, namely $\mathsf{mayWrite} \sqsubseteq \mathsf{mayOpen}$.

4.3 Conducting Attribute Exploration on Instantiations

We consider the instantiation of a $\mathcal{T}$-context, where we want to run attribute exploration on. Obviously, for any $\mathcal{T}$-context $\mathbb{K}$, there exists a smallest instantiation $\mathbb{K}_{\min}$, which is isomorphic to $\mathbb{K}$, and a largest instantiation $\mathbb{K}_{\max}$. The basic idea is that we start the attribute exploration with $\mathbb{K}_{\min}$, and for implications which do not hold, we add (as usual) counterexamples to the context, until we finally reach a context $\mathbb{K}_{\text{ae}}$. Anyhow, in this process, we cannot add counterexamples in an arbitrary manner, as the context $\mathbb{K}_{\text{ae}}$ we obtain must still be an instantiation. The question is how this additional constraint can be captured by attribute exploration. First of all, we trivially have the following subset relations between the implications which hold in the contexts:

$$Imp(\mathbb{K}_{\max}) \subseteq Imp(\mathbb{K}_{\text{ae}}) \subseteq Imp(\mathbb{K}_{\min})$$

So if we run an attribute exploration on $Imp(\mathbb{K}_{\min})$, we could use $Imp(\mathbb{K}_{\max})$ as a set of additional background implications. Anyhow, a closer observation yields that $Imp(\mathbb{K}_{\max})$ only contains all implications of the form $\emptyset \to m$, where m is an attribute of $\mathbb{K}_{\min}$ which applies to all objects. This can easily be seen as follows: Let $\mathbb{K}_{\min} := (O_{\min}, M, I_{\min})$, let $\mathbb{K}_{\max} := (O_{\max}, M, I_{\max})$, let $M_1 := \{m \in M \mid \forall o \in O_{\min} : (o, m) \in I_{\min}\}$ and $M_2 := M - M_1$ be the complement of M_1. First of all, we obviously have that $\emptyset \to m_1$ holds in $\mathbb{K}_{\min}$, thus in $\mathbb{K}_{\max}$ as well, for each $m_1 \in M_1$. Now let $m_2 \in M_2$. Then there exists an object $o \in O_{\max}$ with $(o, m) \in I_{\max} \Leftrightarrow m \neq m_2$ for all $m \in M$. That is, there cannot exist any (nontrivial) implication in $Imp(\mathbb{K}_{\max})$ with m_2 in its conclusion.

4.4 Choice of Instantiation Contexts for Attribute Exploration

Theoretically, we could conduct an attribute exploration on the minimal instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$. Anyhow, we observe that any instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$ is the subposition of instantiations of the contexts$\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$, $\mathsf{P} \in \mathbf{P}$. Generally, for any contexts $\mathbb{K}_1, \ldots, \mathbb{K}_n$ with identical attribute sets, an implication holds in each context $\mathbb{K}_1, \ldots, \mathbb{K}_n$ if and only if it holds in the subposition of these contexts. Thus if the RBAC engineer runs an attribute exploration on the minimal instantiation of all contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$, $\mathsf{P} \in \mathbf{P}$, there is no need to run an attribute exploration on the minimal instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$.

The discussion above applies to the context $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$ as well. To summarize: For a *complete* investigation of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, the RBAC engineer should run an attribute exploration on the minimal instantiations of the following contexts:

- $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ for each permission $\mathsf{P} \in \mathbf{P}$ to infer document implications
- $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ for each permission $\mathsf{P} \in \mathbf{P}$ to infer role implications
- $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$ to infer permission implications

For the context $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$, one could add the additional attribute $\perp$ in order to obtain constraints which express the disjointness of some permissions.

5 Evaluation of the Approach for a Real-Life-Example

In this section, we apply our approach to the example introduced in Tab. 1. Due to space limitations, we do not conduct a complete attribute exploration: Instead we consider only the contexts $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$ and $\mathbb{K}^{\mathsf{MO}}_{\mathbf{D},\mathbf{R}}$.

	MO	MW	MA	$\perp$
$(mv1, um1)$				
$(sc1, um1)$	$\times$			
$(sde1, um1)$	$\times$	$\times$		
...				
$(sv1, ig1)$	$\times$		$\times$	
...				
$(cse1, re1)$	$\times$			

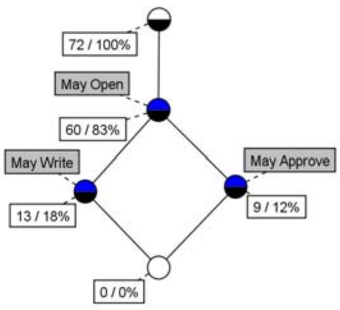


Fig. 1. The instantiation context $\mathbb{K}_{D\times R,\mathsf{P}}$ and its concept lattice

5.1 Attribute Exploration for $\mathbb{K}_{D\times R,\mathsf{P}}$.

In this section, we conduct the attribute exploration on the minimal instantiation $\mathbb{K}_{\min}$ of $\mathbb{K}_{D\times R,\mathsf{P}}$. For this exploration, as discussed in Sec. 2, we added an additional attribute $\perp$ to the set of attributes. An excerpt of $\mathbb{K}_{\min}$, together with its concept lattice, is provided in Fig. 1. This is the context the RBAC engineer starts the attribute exploration on. In $\mathbb{K}_{D\times R,\mathsf{P}}$, thus in $\mathbb{K}_{\min}$, we have the following implications:

1. $\mathsf{MW} \to \mathsf{MO}$
2. $\mathsf{MA} \to \mathsf{MO}$
3. $\perp \to \mathsf{MO} \wedge \mathsf{MW} \wedge \mathsf{MA}$
4. $\mathsf{MO} \wedge \mathsf{MW} \wedge \mathsf{MA} \to \perp$.

The first implication is read: Whenever some agent can write some document, then this agent can open this document as well. It can easily be verified that this implication should indeed hold in any interpretation $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$, so we add the property inclusion mayWrite$\sqsubseteq$mayOpen to our DL knowledge base. This is the first example of a statement which can be modeled with a DL statement, but not with matrix $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ alone.

The next implication can be handled analogously, and we add the inclusion mayApprove$\sqsubseteq$mayOpen to the knowledge base.

The third implication trivially holds due to the definition of $\perp$.

The last implication can, due to the first two implications, be simplified to $\mathsf{MW} \wedge \mathsf{MA} \to \perp$. Due to the definition of $\perp$, this is read: No agent can both write and approve some document. Again, the engineer decides that this implication is valid. Thus she adds the disjoint property axiom $\mathsf{MW} \sqcap \mathsf{MA} \sqsubseteq \perp$ to the knowledge base.

If it is later verified that the complete RBAC policy is consistent, which can be done with a DL reasoner, then each document which can be written or can be approved has to be readable and furthermore no document can be written and approved by the same agent. These are constraints which have not been contained in the matrix but where derived by our methodology.

5.2 Attribute Exploration for $\mathbb{K}_{\boldsymbol{D},\mathsf{R}}^{\mathsf{mayOpen}}$.

For a second example, attribute exploration is performed on the minimal instantiation context $\mathbb{K}_{\min}$ of $\mathbb{K}_{\mathsf{D},\mathsf{R}}^{\mathsf{mayOpen}}$. The context $\mathbb{K}_{\min}$ looks like the left third of the cross table in Tab. 1 despite that it is transposed and document types are replaced by documents (columns are roles, rows are documents). Due to space limitation, we do not conduct a complete attribute exploration on $\mathbb{K}_{\min}$, but only provide an example for an valid and an invalid implication.

Let us first note that in $\mathbb{K}_{\mathsf{D},\mathsf{R}}^{\mathsf{mayOpen}}$, the attributes SV, LDE and CSE apply to all objects. So, according to the discussion in the implications $\emptyset \to \mathsf{SV}$, $\emptyset \to \mathsf{LDE}$ and $\emptyset \to \mathsf{CSE}$ hold in all instantiations of $\mathbb{K}_{\mathsf{D},\mathsf{R}}^{\mathsf{mayOpen}}$, thus we can add the GCIs $\top \sqsubseteq \forall \mathsf{SV}.\mathsf{mayOpen}^-$, $\top \sqsubseteq \forall \mathsf{LDE}.\mathsf{mayOpen}^-$ and $\top \sqsubseteq \forall \mathsf{CSE}.\mathsf{mayOpen}^-$ to our knowledge base.

A example for an implication (of the stem base) of $\mathbb{K}_{\min}$ is $\mathsf{TE} \to \mathsf{ME}$. During the attribute exploration, the RBAC engineer has to decide whether this implication holds in all desired interpretations of $\mathbb{K}_{\mathsf{D},\mathsf{R}}^{\mathsf{mayOpen}}$. In fact there might be a contract document in preparation by a technical editor which is not allowed to be opened by a marketing employee. Thus the RBAC engineer adds a counterexample to the context $(\mathsf{CCD\_in\_prep}, \mathsf{TE}, \mathsf{MO}) \in I$ and $(\mathsf{CCD\_in\_prep}, \mathsf{ME}, \mathsf{MO}) \notin I$.

Another example for an implication (of the stem base) of is $\mathsf{MV} \to \mathsf{SC}$. In fact, the RBAC engineer realizes that this implication must hold: Any document which can be opened by a marketplace visitor can be opened by a service consumer as well. So she adds the GCI $\forall \mathsf{MV}.\mathsf{mayOpen}^- \sqsubseteq \forall \mathsf{SC}.\mathsf{mayOpen}^-$ to the knowledge base. This is again an example which cannot be derived from $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ alone.

6 Conclusion and Future Research

In this paper we used the access control matrix as basic model for the behavior of RBAC and called this an RBAC matrix. We discussed three interpretations of an RBAC matrix and described that for the permissive approach additional constraints can be derived which are not contained in the RBAC matrix. This additional information was added to a so called RBAC policy, modeled in DL.

For obtaining a *complete* RBAC policy, we introduced a strict methodology, based on FCA. The general approach was to derive different dyadic context from RBAC matrix context $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ and conduct an attribute exploration on them. The attribute exploration allowed finding unintended implications and to derive constraints and make them explicit.

Our ongoing work comprises several directions. First, we are seeking a smaller DL fragment which meets our modeling requirements. This is particularly for the permissive approach essential, as the DL modelling we used so far is based on some non-standard DL constructors. Next, we want to support positive and negative authorizations in one policy. That is, we want to combine the permissive and prohibitive approach, so we have to investigate how our approach has to be extended in order to do so. Finally, recall that our approach was based on the assumption that sets of roles resp. document types are fixed. In some applications this might be too strict. The three interpretations would have to be adapted and even attribute exploration for the strict approach might make sense if we drop this assumption. This is subject of future research as well. In the long run, we target at a comprehensive methodology for utilizing DLs for RBAC.

References

1. Lampson, B.: Protection. In: Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems, pp. 437–443 (1971)
2. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control: towards a unified standard. In: RBAC 2000: Proceedings of the fifth ACM workshop on Role-based access control, pp. 47–63. ACM Press, New York (2000)
3. Saunders, G., Hitchens, M., Varadharajan, V.: Role-based access control and the access control matrix. SIGOPS Oper. Syst. Rev. 35(4), 6–20 (2001)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications, 2nd edn. Cambridge University Press, Cambridge (2007)
5. Knechtel, M., Hladik, J.: RBAC authorization decision with DL reasoning. In: ICWI 2008: Proceedings of the IADIS Int. Conf. WWW/Internet (2008)
6. Knechtel, M., Hladik, J., Dau, F.: Using OWL DL reasoning to decide about authorization in RBAC. In: OWLED 2008: Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions (2008)
7. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: Proceedings of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI 2007). AAAI Press, Menlo Park (2007)
8. Lutz, C., Sattler, U.: Mary likes all cats. In: Baader, F., Sattler, U. (eds.) Proceedings of the 2000 Int. Workshop in Description Logics (DL 2000), Aachen, Germany, August 2000. CEUR-WS, vol. 33, pp. 213–226. RWTH Aachen (2000), http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/
9. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: Proceedings of the 21st International Workshop on Description Logics (DL 2008) (2008)
10. Ganter, B., Obiedkov, S.A.: Implications in triadic formal contexts. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS, vol. 3127, pp. 186–195. Springer, Heidelberg (2004)

Using VRML Technology for Visualization of Relations between the Main Concepts of an Educational Course

Evgeny Eremin

Perm State Pedagogical University,
Sibirskaya, 24, Perm, 614990, Russia
eremin@pspu.ac.ru

Abstract. This paper proposes to use the VRML technology to save and display visual schemes of Topic Maps representing the structures of educational subjects. As learning purposes always require the careful selection of the most pertinent concepts from the total scientific knowledge, the rendering of Topic Maps for educational scope need not be too large. Special software has been developed to convert the original TM files into corresponding VRML 3D scenes. Processing of several computer science courses has already produced satisfactory images, and work to improve the visualization will continue.

Keywords: VRML, Topic Map, visualization, relation, concept, knowledge, structure, education.

1 Introduction

Every educational course has a number of conceptual notions and terms that must be compulsory learnt. Being interconnected, they form some kind of semantic *thesaurus*, which contains not only a list of terms, but, more importantly, the relations between them. Such course's "skeleton", formulated very close to computer presentation of the data, may be considered as a conceptual model of the selected knowledge domain. It is clear that visualization of the thesaurus structure can help both students and teachers in the learning process.

Like various other knowledge structures, interrelated educational concepts can be described using different computer technologies, for instance in the form of ontologies [1]. The author has found the Topic Map technology [2] to be very convenient for representing the thesaurus of a course. But while Topic Map structures, based on XML text, succinctly generalize a human experience, they obviously lack a pictorial representation.

The problem of visualizing Topic Maps has a long history. The comprehensive illustrated review by Le Grand and Soto [3] particularly describes the current state of the question. It shows the difficulty of the issue and numerous attempts to find appropriate solutions. Most of the reviewed works have a very general goal of visualizing millions of topic nodes with unlimited associations between them.

In contrast to this approach, educational Topic Maps tends to have few nodes and a limited number of associations, so we may try to depict them in their entirety on a computer display. From the sense of the learning process it follows that only the most essential concepts must be examined and bulk of the incidental detail must be removed to make the theories under consideration clear for students. To illustrate the difference, consider comparing the full specification of a passenger aircraft with a description of its organization for passenger transportation.

2 Statement of the Problem

This work is devoted to the visualization of Topic Maps representing educational thesauruses. The free software TM4l [4], [5] has been used to build the structures of several computer science courses, such as Introduction to Computer Architecture [6]. In order to get more demonstrative visual images of these structures, we have added a depth dimension to the conventional flat drawing. Special software developed by the author is used to specify a 3D graphic map of concepts in the well-known VRML language. Such technology provides the possibility to study the structure of educational materials by means of one of the numerous VRML viewers. The free Right Hemisphere Deep View [7] was recognized as the most suitable for our aims.

Let's consider a typical educational course and pick out its main concepts. Generally we expect to find no more than 100-150 topics after this procedure. Our analysis of the associations between concept terms includes standard "whole-part" and "class-subclass" relations, as well as specific ones defined for our course; typical custom associations for the computer science domain are "theoretical base", "connection" and "control". Building a set of the governing relationships is not predefined and self-evident procedure, but we have found that quite a small number of relational categories is enough to describe an educational domain. Their full table for our computer architecture course [6] consists of only 11 basic types of interrelations.

As a result we get the thesaurus, containing all of the fundamental terms and the interrelations between them. It can be directly entered into the computer in the form of a Topic Map, and is then ready for our analysis.

In addition to the development of Topic Map files for specific computer science courses, the contribution of this work is the creation of the software that reads these files, analyzes the character of the interrelations in the thesaurus, and specifies the results of topics' arrangement in VRML language. Using this software, suitable representation of the main concepts was obtained on the display screen.

3 Discussion of Results

It is clearly not a trivial task to produce an optimal rendering of the structure of an educational course, so here we briefly present the most general algorithm of the software. To generate a VRML scene with interdependent spatial objects, our program carries out the following operations:

- reads a selected XML file with Topic Map, parses it and builds tables of concepts and relations between them;
- sorts the concepts, arranging them according to their interrelations, in order to get some regularized groups of connected concepts;

- allocates the visual symbols of the concepts in space and associates them with graphical links, trying to optimize the visibility of all concepts;
- calculates coordinates for the graphical objects and fixes them in the VRML notation.

To arrange the concepts we tested several different algorithms, based mainly on the idea of *layers*. We may define a layer as a collection of concepts, that are equally distant from the basic concept of the whole map (TM4l editor demands to set such main topic mandatory). In the educational courses we have explored the typical number of layers ranges from five to nine. For instance, one version of the tested Topic Map for the course of Pascal language consists of eight layers, containing 3, 6, 13, 27, 18, 10, 6 and 2 terms.

Every layer can be split into sub-layers according to correlations between its topics. Our exploration shows that drawing of the internal links is the most difficult problem, so evaluation of object's position within the layer is extremely important.

We have considered several variations of the algorithm, such as combining all nodes with a common parent, splitting a layer into several parts on parallel planes, changing the coordinate system, and more. No variation produced an ideal method of visualization, although some combinations of strokes generated satisfactory images.

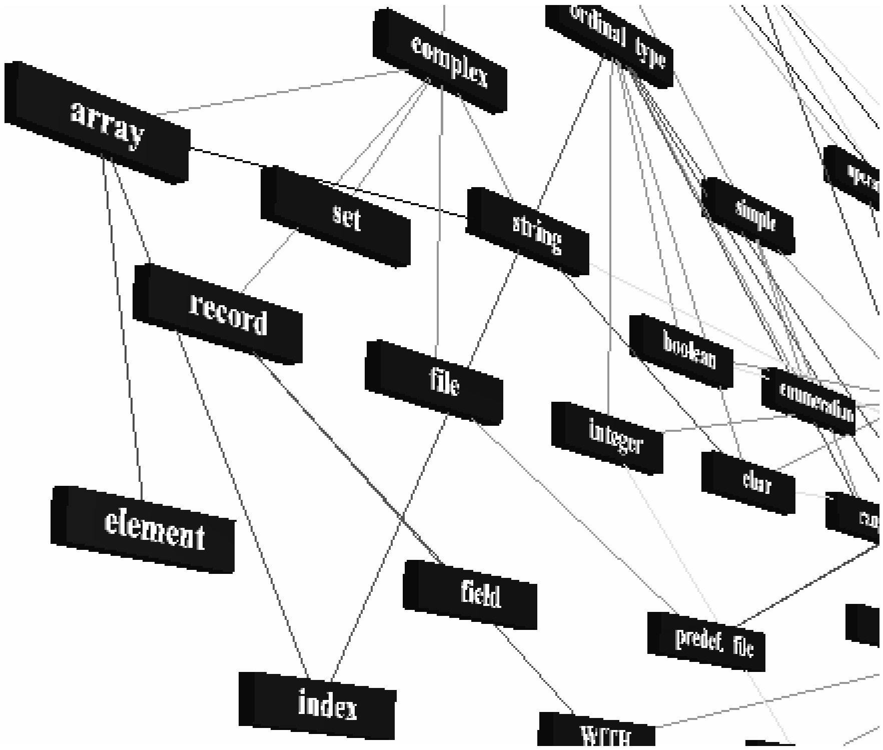


Fig. 1. Fragment of the VRML scheme for the main concepts of Pascal language, captured directly from computer screen

These explorations gave rise to the idea of creating an original software toolbox for processing Topic Maps, which was realized by the author, using Borland Turbo Delphi. Future work will include adding new methods of visualization to this toolbox.

Applying our software to prepared Topic Maps for several computer science disciplines confirmed that all of the conceptual notions could be automatically combined and displayed together on the computer screen. The results look satisfactory and certainly readable. The added ability to travel across the VRML scheme gives us a good tool to observe as an organic whole the foundations of an educational course.

Fig. 1 illustrates a fragment of the VRML 3D map describing the main concepts of Pascal. Every concept on this scheme is a parallelepiped labeled with a term from the knowledge domain. Multicolored lines, representing different types of relations, connect the concepts into a semantic network. The picture shows the teacher or student all of the basic interrelations of the course, for instance, *string-array-char*, *index-ordinal type*, *file-predefined file-input-output* and so on. Such a holistic view of the course can clearly be very useful for learning.

Developed method of representation has obvious practical application. From the educational point of view such a map of a course may be very helpful for both lecturer and students, as it clearly shows the relations between different parts of the material and demonstrates the role of every concept in the learning discipline.

Work will continue on improving the graphical VRML representation of educational courses.

References

1. Milam, J.: Ontologies in High Education. In: Metcalfe, A.S. (ed.) Knowledge Management and Higher Education: A Critical Analysis, pp. 34–62. Idea Group Publishing, Hershey (2005)
2. Park, J., Hunting, S.: XML Topic Maps: Creating and Using Topic Maps for the Web. Addison-Wesley, Boston (2003)
3. Le Grand, B., Soto, M.: TopicMaps, RDF Graphs, and Ontologies Visualization. In: Geroimenko, V., Chen, C. (eds.) Visualizing the Semantic Web: XML-Based Internet and Information Visualization, pp. 59–79. Springer, London (2006)
4. Topic Maps 4 e-Learning, `http://compsci.wssu.edu/iis/nsdl/`
5. Dicheva, D., Dichev, C.: TM4L: Creating and Browsing Educational Topic Maps. British Journal of Educational Technology 37(3), 391–404 (2006)
6. Eremin, E.A.: Using Topic Map Technology in the Planning of Courses from the CS Knowledge Domain. In: Lister, R., Simon (eds.) VII Baltic Sea Conference on Computing Education Research. CRPIT, vol. 88, pp. 179–182. ACS (2007)
7. Right Hemisphere Deep View, `http://www.righthemisphere.com/products/deeppub/DeepPub_View/index.html`

Efficient Browsing and Update of Complex Data Based on the Decomposition of Contexts

Sébastien Ferré

Université de Rennes 1, CNRS
Campus de Beaulieu, 35042 Rennes cedex, France
ferre@irisa.fr

Abstract. Formal concept analysis is recognized as a good paradigm for browsing data sets. Besides browsing, update and complex data are other important aspects of information systems. To have an efficient implementation of concept-based information systems is difficult because of the diversity of complex data and the computation of conceptual structures, but essential for the scalability to real-world applications. We propose to decompose contexts into simpler and specialized components: *logical context functors*. We demonstrate this allows for scalable implementations, updatable ontologies, and richer navigation structures, while retaining genericity.

1 Introduction

Formal Concept Analysis (FCA) [GW99] has been recognized as a good paradigm for browsing data sets [GMA93, CES03, FR04]. Besides browsing (querying and navigation), update is another important aspect of information systems. FCA is defined on binary relations between objects and attributes. Those relations are called *formal contexts*. In practice, data is generally more complex than the simple attributes of formal contexts: e.g., numbers and intervals, strings and patterns, valued attributes [GW89], vectors, trees, graphs [GK01]. Furthermore, logical dependencies may exist in complex data: e.g., if an object has the property $age = 23$, it implicitly has the more general property $age \in [20, 30]$. A first approach to handle complex data in FCA is *conceptual scaling* [GW89], which is a process that takes complex data as an input, and outputs a standard formal context, called the *scaled context*. In the scaled context, the attributes are abstraction of the original data, and the incidence relation reflects their internal logic. For example, Prediger et al [PS99] use description logics to define the meaning a finite set of chosen attributes; and Tane et al [TCH06] use the same description logics to compute scaled contexts as views over a complex knowledge base. A second approach [CM00, FR00, GK01] strives to keep complex data in its original form by generalizing the definition of a formal context as well as other FCA operations (e.g., Galois connection, concept lattice). For example, Logical Concept Analysis (LCA) [FR00] uses logical formulas instead of sets of attributes to represent and manipulate object descriptions, concept intents, queries, and navigation links between concepts. The first approach allows the reuse of FCA algorithms and tools on complex data, while the second approach keeps the original form of complex data, and entails no loss of information.

This paper discusses the efficient implementation of concept-based information systems. In the two above approaches, the existing implementations make no or little use of the specificities of complex data. For example, the data structure used to represent contexts in LCA, called the *logic cache* [FR04], makes use of the logical entailment between formulas, but makes no difference between a string, an interval or a graph. This entails the following problems: (1) the update of the context is not efficient enough to support scalability (10,000 objects at most), (2) any change in the logic requires the complete recomputation of the logic cache, and (3) the set of navigation links is not informative enough.

We propose to benefit from the nature of the original complex data to solve the above problems. For instance, there exist dedicated data structures and algorithms for strings, which can be used to build specialized implementations of contexts where objects are described by strings. The same can be done for other concrete domains, or taxonomies. Now, if objects are described by string-valued attributes, a specialized implementation can be composed from two specialized contexts: one for attributes, and the other for strings. Operations for composing contexts have been defined in FCA [GW99]: e.g., apposition, direct product. However, they apply to formal contexts only, and their implementation is not discussed. We detail in this paper the definition and specialized implementation of both primitive contexts and composition operations. Those are collectively called *logical context functors*, because they are functions (with zero, one or several arguments) from logical contexts to logical contexts. The term *functor* is taken from the domain of functional programming where it denotes functions from modules to modules [Mog89], which are precisely used to implement our functors.

Section 2 recalls the basics of LCA, and introduce the browsing and update operations. Section 3 explains the problems of the logic cache, the existing LCA implementation. Section 4 defines a *logical context functor* as an extension of a *logic functor*, and details five functors: string, taxonomy, product, disjoint union, and *root*. Section 5 illustrates the use of logical context functors on two real examples (string-valued attributes, and user-tag annotations), and demonstrates their efficiency by giving the complexity of operations compared to the logic cache. For instance, the addition of an object into a context of n objects is in $O(1)$ or $O(\ln(n))$ instead of $O(n)$.

2 Logical Contexts and Logical Information Systems

The LCA framework [FR04] applies to logics with a set-valued semantics similar to description logics [BCM$^+$03]. The logic is not fixed *a priori* so that it can be customized to different applications. Examples of logical formulas are binary attributes, attributes valued on different concrete domains (e.g., strings, intervals, dates), terms from taxonomies, and any combination of those such as lists, trees or graphs. It is sufficient here to define a logic (see [FR04] for a detailed presentation) as a pre-order of formulas. The pre-ordering is the logical entailment, called *subsumption*: e.g., an interval included in another one, a string matching some regular expression, a graph being a subgraph of another one.

Definition 1 (logic). *A* logic *is a pre-order $\mathcal{L}_T = (L, \sqsubseteq_T)$, where L is a set of formulas, T is a customizable parameter of the logic, and $\sqsubseteq_T$ is a* subsumption *relation that depends on T. The relation $f \sqsubseteq_T g$ reads "f is more specific than g" or "f is subsumed by g", and is also used to denote the partial ordering induced from the pre-order.*

The parameter T helps to take into account domain knowledge that may change over time: e.g., an ontology, a taxonomy. In the following, for simplicity, we designate this parameter as the "ontology", and consider it is a set of *subsumption axioms* $f \prec g$: e.g., $cat \prec animal$, $Quebec \prec Canada$. In addition to the logic and its ontology, the *logical context* constitutes the third level of knowledge. It defines a set of objects along with their logical description, and a finite subset of formulas, called *vocabulary*, that is used for navigation.

Definition 2 (logical context). *A* logical context *is a tuple $K = (\mathcal{O}, \mathcal{L}_T, X, d)$, where $\mathcal{O}$ is a finite set of objects, $\mathcal{L}_T$ is a logic, $X \subseteq L$ is a finite subset of formulas called the* navigation vocabulary*, and $d \in (\mathcal{O} \to \mathcal{L}_T)$ is a mapping from objects to logical formulas. For any object o, the formula $d(o)$ denotes the* description *of o.*

Each formula is described by a single formula for genericity reasons. Even if a description is often in practice a set of properties, it can also be a sequence of properties or any other data structure. The definition of a vocabulary is necessary because there is often an infinite set of formulas (e.g., intervals, strings). The choice of a relevant vocabulary depends on both the logic and object descriptions, and a contribution of this paper is precisely to show how it can be automatically generated in a logical context. The elements of the vocabulary are called *features*.

Logical contexts make up the core of Logical Information Systems (LIS) [FR04], so that we need both update and information retrieval operations on them. Update operations apply to the ontology, the objects, and the navigation vocabulary. For every formulas $f, g \in L$:

- $K.axiom(f, g)$ adds the axiom $f \prec g$ to the ontology T, which modifies the behaviour of the subsumption $\sqsubseteq_T$;
- $K.add(o, f)$ adds the new object o to the set of objects $\mathcal{O}$, and sets its description $d(o)$ to f;
- $K.show(f)$ adds the formula f to the navigation vocabulary X.

The *filling of a context* is the successive addition of a set of objects, defining the updatable part of the context. There are also update operations for removing axioms, modifying the description of objects, removing objects, and hiding formulas, but we do not detail them here.

A key feature of LIS, shared by other concept-based information systems [GMA93, DVE06], is to allow the tight combination of querying and navigation. The principle is that, instead of returning a ranking of all the answers to the query, the system returns a set of query *increments* that suggest to users relevant ways to refine the query, i.e., navigation links between concepts, until a manageable amount of answers is reached. They are two information retrieval operations on logical contexts: one to compute the query answers, and another to compute the query increments from those answers.

A query is a logical formula, and its answers are defined as the extent of this formula, i.e., the set of objects whose description is subsumed by this formula.

Definition 3 (extent). *Let K be a logical context, and $q \in \mathcal{L}$ be a query formula. The* extent *of q in K is defined by $K.ext(q) = \{o \in \mathcal{O} \mid d(o) \sqsubseteq_T q\}$.*

The increments of a query q are the features that are frequent in the extent of q, i.e., the features whose extent shares a pre-defined minimum m of objects with the extent of q: $\{y \in X \mid \ |K.ext(q) \cap K.ext(y)| \geq m\}$. Those increments are partially ordered by subsumption, and should be presented so to users because this gives a more comprehensive view. For instance, if the vocabulary contains continents, countries, and regions, it is better to display them as a tree rather than a flat list. Moreover, it is not necessary to compute and display all of them at once; continents should be displayed first, and could then be expanded on demand to display countries, etc. So, the navigation operation takes an increment x and returns its lower neighbours that are also increments. For technical reasons, we prefer to compute increments w.r.t. a set of objects $O \subseteq K.ext(x)$ instead of a query. Starting with a query q, that set O is defined as $K.ext(q) \cap K.ext(x)$. Each increment is returned with the extent of the concept that would be reached by using it as a navigation link.

Definition 4 (children increments). *Let K be a logical context, $x \in X$ be an increment, O be a set of objects s.t. $O \subseteq K.ext(x)$, and m be a frequency threshold. The* children increments *of x, called the* parent increment*, w.r.t. O at threshold m is defined by $K.incrs(x, O, m) =$*

$$Max_{\sqsubseteq_T}\{(y, O') \mid y \in X,\ y \sqsubseteq_T x\ ,\ x \not\sqsubseteq_T y\ , O' = O \cap K.ext(y), |O'| \geq m\},$$

where $\sqsubseteq_T$ is trivially extended to pairs (y, O').

These increments provide feedback on the current query and answers, as well as several forms of navigation [Fer09]. LIS have been applied to many kinds of data, and most noticeably to the management of a collection of > 5000 photos [Fer09], which can be browsed and updated in terms of time, location, event, persons, and objects.

3 Problems with Logic Caches

Logical information systems were designed to be generic, and so can make no assumption on the logic, except for the existence of a decision procedure for subsumption (decidability). Because this subsumption test is costly for some expressive logics, the choice was made to minimize its use in browsing operations, which are more frequent than update operations. Therefore the cost of computing subsumption is moved to update operations, in the form of an incremental preprocessing, whose result is called a *logic cache* [FR04].

There are a number of problems with logic caches as a generic implementation. The first problem is that efficient browsing has been achieved at the cost of a quadratic complexity for the building of a logic cache. This is more acceptable to users than lengthy response times in browsing operations, but this strongly limits the scalability of LIS. An appropriate complexity would be $O(n)$ or at most $O(n \ln(n))$. The second problem is the operation $K.axiom$ that requires a complete recomputation of the logic cache. This is because the impact of new axioms on the subsumption is only known by the logic, and not by the generic logic cache. In order to take real profit of changing ontologies, the ability to handle them incrementally is crucial. The third problem is in the design of the vocabulary (the set of scale attributes in conceptual scaling). There is a conflict between having a rich and progressive navigation that requires a large vocabulary, and the efficient filling of contexts that requires a small vocabulary. Consider the example where objects are documents, and object descriptions are titles, i.e. strings. If the vocabulary contains only full titles, the navigation structure is a flat list of titles, which is not very interesting. If the vocabulary contains all words occuring in the titles, the navigation structure is more interesting, but: (1) it still misses composed keywords such as "formal concept analysis"; (2) it contains unique words that are specific to one title, and in this case it is better to show the full title; (3) it is costly compared to the small vocabulary.

Consider another context where documents are described by a set of pairs (user, tag), and users/tags are organized in two taxonomies. The taxonomy of users defines various groups over users (possibly overlapping), and the taxonomy of tags defines a generalization ordering on tags. This context enables different kinds of queries: Which documents have been given this kind of tags by this kind of users ? Which tags have been given by this kind of users on this set of documents ? Which users have given this kind of tags on this set of documents ? Even if each taxonomy has a reasonable size, say 1000, the vocabulary will contain all pairs (user, tag) that have a non-empty extent, which can go up to 1 million pairs. Intuitively, it should be possible to keep each taxonomy on its side as a kind of partial logic cache, and to combine them on the fly, thus bounding the size of the total logic cache to the sum rather than the product of taxonomies sizes. In the next section, this decomposition is formalized by *logical context functors*.

4 Logical Context Functors

We introduce in this section *logical context functors*, i.e. functions that build logical contexts and their operations from simpler parts. We show how they solve the problems presented in the previous section: efficiency, ontology evolution, and selection of the navigation vocabulary. The problem of efficiency comes from the well-known trade-off between genericity and efficiency. Efficiency requires specific data structures and algorithms, which is *a priori* incompatible with the need for genericity. A solution, that has already been done and validated on logics [FR04], is to define specific components, called *functors*, and to allow their composition in more and more complex components, in which the correction of operations w.r.t. semantics is automatically verified [FR06]. The efficiency comes from the specificity of functors, and the genericity comes from the ability to compose them. This is similar to languages where a finite number of different words can be combined in an infinity of sentences. Of course, the expressivity is limited by the available functors, but new functors can always be added to the set. Another benefit of logic functors is to permit the choice of the *right* logic for each application, instead of having an all-purpose logic that is costly to use, and cannot cover the specific needs of all applications.

Definition 5 (logic functor). *A* logic functor *is a function $\mathcal{F}$ that takes logics $\mathcal{L}_1, ..., \mathcal{L}_n$ as arguments ($n \geq 0$), returns a composed logic $\mathcal{L}_T = \mathcal{F}(\mathcal{L}_1, ..., \mathcal{L}_n)$. As for any logic, one has $\mathcal{L}_T = (L, \sqsubseteq_T)$, but L (resp. $\sqsubseteq_T$) is function of the sets of formulas (resp. subsumption) of arguments logics.*

Examples of logic functors are *String* that takes no argument, and returns the set of all strings ordered by the subsumption relation "contains"; and $\mathit{Prod}(\mathcal{L}_1, \mathcal{L}_2)$ that takes two arguments, and returns the logic where formulas are pairs (f_1, f_2) of formulas from $\mathcal{L}_1$ and $\mathcal{L}_2$, and the subsumption test is naturally decomposed in the two subsumption tests. These examples and others are more formally defined below, along with logical context functors. Logical context functors are defined similarly to logic functors, as functions from contexts to contexts.

Definition 6 (logical context functor). *A* logical context functor *is a function F that takes logical contexts $K_1, ..., K_n$ as arguments ($n \geq 0$), returns a composed context $K = F(K_1, ..., K_n)$. As for any context, one has $K = (\mathcal{O}, \mathcal{L}_T, X, d)$, but each part of this context is function of the respective parts of argument contexts.*

Logical context functors and logic functors are implemented as *functors*[1] in the Objective Caml programming language. In order to reuse code from logics into contexts, logical context functors *inherit* (in the object-oriented sense) from their respective logic functors.

In the following we detail a few common logical context functors. For each functor we define the set of objects, the set of formulas, the subsumption, the vocabulary, and the description mapping as a function of argument contexts. Besides this mathematical point of view, each functor is also given data structures and algorithms for the implementation of logical context operations. Complexities are given in function of the number n of objects, the size x of the vocabulary, and the number i of children increments.

[1] Similar structures exist in other programming languages: e.g., parameterized modules (ML), generic classes (Java), templates (C++).

4.1 String Context

The logical context functor *String* represents the concrete domain of strings and substrings ordered by string inclusion. It has no argument, so that it constitutes a logical context that can be used alone or as part of a more complex logical context. This logical context could also be defined as $Seq(Char)$, where *Seq* and *Char* would be two functors respectively about sequences and characters.

The logic is simply defined as $\mathcal{L} = (\Sigma^*, \supseteq)$, where Σ^* stands for finite strings over an alphabet Σ, and $\supseteq$ is the "contains" relation on strings. It is not parameterized by an ontology because it is a concrete domain. It enables to describe objects by strings, and to query them with patterns like `contains "FCA"`. A previous work [Fer07] has shown that a concise and complete vocabulary X can be efficiently and automatically extracted from such a context: the set of *maximal substrings*. This vocabulary is complete because the set of formal concepts it generates is the same as when considering all possible substrings. It is concise because its size is bounded by the cumulated size kn of strings describing the n objects in K, where k is the average length of strings. It can be computed in $O(kn \ln(kn))$. In practice, however, the number of maximal substrings is generally much lower than kn (e.g., 3,816 instead of 52,360 [Fer07]).

The data structure that permits to store the string-description of objects, and to compute the vocabulary, is an extended *suffix tree* [Fer07]. Its complexity in space is in $O(kn \ln(kn))$. It naturally provides the efficient incremental addition of a string, and hence the addition of an object in the context: $K.add(o, s)$ takes $O(|s| \ln(kn))$ time, where $|s|$ is the length of the string s. Other operations can also be directly performed on the extended suffix tree. For the operation $K.show(s)$ one reads the string s down the suffix tree, in $O(|s|)$ time, and marks the reached node as visible. For the operation $K.ext(s)$, one also reads the string s, and then collects the objects below the reached node: $O(|s| + n)$ time. For the operation $K.incrs(x, O, m)$, one again reads the string s, then follows links from the reached node to find the smallest maximal substrings below, and finally filters those that are frequent in O: $O(|s| + ni)$ time.

The most visible advantage of the functor, compared to the logic cache, is the computation of a rich, yet concise, vocabulary that provides a much better navigation feeling because it dynamically adapts to the context contents. Another advantage is the efficiency in the filling of the context: $O(kn \ln(kn))$ instead of $O(k^4 n^2)$. Even when the vocabulary is restricted to the strings describing objects, the complexity of the logic cache is in $O(kn^2)$, still quadratic in the number of objects. This makes a huge difference in practice when the number of objects gets high, and this efficiency comes with a much larger and useful vocabulary.

4.2 Taxonomy Context

Taxonomies are very helpful in the organization of a collection of documents [Sac00]. They are a simple kind of ontologies in which an axiom states that a term is more specific than another. Examples of such axioms are $inQuebec \prec inCanada$, or $cat \prec mammal$, $mammal \prec animal$. The logical context functor *Taxo* produces contexts in which each object is described by one term, and implicitly has all more general terms. For instance, an object described as a *cat* is also an instance of *mammal*, and *animal*. The logic of those contexts is naturally derived from the taxonomy T that parameterizes it.

Definition 7 (taxonomy logic). *Let $T = (L, A)$ be a taxonomy, where L is the set of terms, and A is a set of taxonomic axioms $f \prec g$. The logic of functor Taxo is defined by $\mathcal{L}_T = (L, \sqsubseteq_T)$, where the subsumption $\sqsubseteq_T$ is defined as the reflexive and transitive closure of the relation $\prec$.*

Because the set of terms of a taxonomy is finite, the vocabulary can be defined as $X = L$, ensuring it to be complete w.r.t. the generation of concepts. The data structure used in the implementation of *Taxo* is the taxonomy seen as a graph: nodes are terms from X, edges are the taxonomic axioms from T, and terms are labelled by their extent. We call *ancestors* of a term f all terms that subsume f (including f), which can be found in the graph by transitively collecting successors.

The addition of a new object, $K.add(o, f)$, consists in adding the term f into X with an empty extent if it is not yet in X, then in adding o to the extent of every ancestor of f. This operation is in $O(a)$ with a the number of ancestors. The operation $K.show(f)$ is unnecessary because every relevant formula is already in the vocabulary. The addition of an axiom $K.axiom(f, g)$ consists in adding an edge from f to g in the graph, and adding the extent of f to the extent of every ancestor of g. This operation is in $O(na)$. It must be noted here that, contrary to a logic cache, it is not necessary to completely recompute the data structure to add an axiom; new axioms can be processed incrementally. This is possible because the functor *Taxo* has knowledge about its logic, and so can update its data structures in parallel to updates in the logic. The operation $K.ext(f)$ consists in a simple access to the label of f in the graph, and is therefore in $O(1)$. The operation $K.incrs(x, O, m)$ consists in filtering among the predecessors of x in the graph those that are frequent in O. This operation is in $O(ni)$.

The shape of the taxonomy has an influence on the complexity of operations. If we assume the taxonomy is a balanced tree with arity k ($a = \ln_k(x)$), the filling of a context is in $n \ln_k(x)$, and the computation of increments is in nk. This suggests to find a compromise between flat taxonomies (small a) and deep taxonomies (large a).

4.3 Product of Contexts

Section 3 presents two examples of contexts, where the formulas can be decomposed as a pair of subformulas. The first example shows valued attributes made of an attribute and a (sub)string; the second example shows pairs (user,tag), where each part is placed into a taxonomy. Instead of developping two new logical context functors, it is more useful to define a logical context functor $Prod(K_1, K_2)$ that produces the product of two simpler contexts. Then, the first example is simply composed as $Prod(Taxo, String)$, and the second example as $Prod(Taxo, Taxo)$. We recall that in logical contexts, each object is described by a single formula, here a pair.

Definition 8 (product of contexts). *Let* $K_1 = (\mathcal{O}, \mathcal{L}_1, X_1, d_1)$ *and* $K_2 = (\mathcal{O}, \mathcal{L}_2, X_2, d_2)$ *two logical contexts sharing a same set of objects. The* product $\mathit{Prod}(K_1, K_2)$ *of these two contexts is defined as a context* $K = (\mathcal{O}, \mathcal{L}, X, d)$*, where* $L = L_1 \times L_2$*,* $(f_1, f_2) \sqsubseteq (g_1, g_2)$ *iff* $f_1 \sqsubseteq_1 g_1$ *and* $f_2 \sqsubseteq_2 g_2$*,* $X = X_1 \times X_2$*, and* $d(o) = (d_1(o), d_2(o))$*.*

The term *product* is justified by the fact that the logical language and the vocabulary are defined as products. There is no equivalent operation on formal contexts [GW99] (i.e., same set of objects, product of attribute sets). The *apposition* uses set union for composing attribute sets, which would entail a loss of the user-tag connection. The *direct product* uses set product to compose object sets, and uses a disjunction that would entail indeterminacy in the operation $K.\mathit{add}$.

The data structure of $K = \mathit{Prod}(K_1, K_2)$ is reduced to the union of the data structures of K_1 and K_2, and there is no proper data structure in the functor *Prod*. Hence, the size of K is the *sum*, and not the product, of the size of the argument contexts. The implementation of *Prod* looks like apposition [GW99], but that is not an apposition because $L \neq L_1 \cup L_2$. This means that X is not explicitely represented, i.e, a pair (f_1, f_2) is only known to belong to X because each part f_i is known to belong to the respective vocabulary X_i. This makes a big difference with a logic cache or conceptual scaling, where the vocabulary $X = X_1 \times X_2$ would be represented explicitly.

Because *Prod* has no proper data structure, operations on K are reduced to a composition of the operations on argument contexts. The operation $K.\mathit{add}(o, (f_1, f_2))$ decomposes itself in the sequence $K_1.\mathit{add}(o, f_1)$; $K_2.\mathit{add}(o, f_2)$. The operations $K.\mathit{show}$ and $K.\mathit{axiom}$ decompose similarly. The time complexity of those three operations are the *sum* of the time complexities in the two argument contexts. Because of the definition of the subsumption $\sqsubseteq$, an object is in the extent of (f_1, f_2) in K if it is in the extent of f_1 in K_1, and in the extent of f_2 in K_2.

$$K.\mathit{ext}((f_1, f_2)) = K_1.\mathit{ext}(f_1) \cap K_2.\mathit{ext}(f_2).$$

The computation of the increments requires the computation of the lower neighbours of a parent increment (x_1, x_2). As the vocabulary X is not explicitely represented, we have to make use of the vocabularies X_1 and X_2, and the respective increments operations. The lower neighbours of (x_1, x_2) are the pairs (y_1, x_2) where y_1 is a lower neighbour of x_1 in X_1, and the pairs (x_1, y_2) where y_2 is a lower neighbour of x_2 in X_2.

$$\begin{aligned} &K.\mathit{incrs}((x_1, x_2), O, m) = \\ &\quad \{((y_1, x_2), O') \mid (y_1, O'_1) \in K_1.\mathit{incrs}(x_1, O, m), O' = O \cap O'_1, |O'| \geq m\} \\ &\quad \cup \{((x_1, y_2), O') \mid (y_2, O'_2) \in K_2.\mathit{incrs}(x_2, O, m), O' = O \cap O'_2, |O'| \geq m\}. \end{aligned}$$

This definition is justified by the definition of the extent in K that makes the extent of a pair (f_1, f_2) in K be a subset of the extent of f_1 and f_2 in their respective context. The time complexity of those operations is equal to the sum of the time complexities in the argument contexts plus additional set intersections: $O(n)$ for $K.\mathit{ext}$, and $O(ni)$ for $K.\mathit{incrs}$.

4.4 Disjoint Union of Contexts

Suppose we want valued attributes whose values belong to one of several value domains, e.g. taxonomic terms and strings. This means we want to define a value domain that is the disjoint union of these value domains. The logical context functor $DisjointUnion(K_1, K_2)$ produces a context whose logic is the disjoint union of the logics of K_1 and K_2: e.g., $DisjointUnion(Taxo, String)$.

Definition 9 (disjoint union of contexts). *Let* $K_1 = (\mathcal{O}_1, \mathcal{L}_1, X_1, d_1)$*, and* $K_2 = (\mathcal{O}_2, \mathcal{L}_2, X_2, d_2)$ *be two logical contexts such that* $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$*, and* $L_1 \cap L_2 = \emptyset$*. The* disjoint union (K_1, K_2) *of these two contexts is defined as a context* $K = (\mathcal{O}, \mathcal{L}, X, d)$*, where* $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$*,* $L = L_1 \cup L_2$*,* $(\sqsubseteq) = (\sqsubseteq_1) \cup (\sqsubseteq_2)$*,* $X = X_1 \cup X_2$*, and* $d(o) = d_1(o)$ *if* $o \in \mathcal{O}_1$*, and* $d(o) = d_2(o)$ *otherwise.*

The condition $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$ is required because an object of K can have only one description, and the condition $L_1 \cap L_2 = \emptyset$ is required to avoid confusion on the meaning of a formula. The term *disjoint union* is justified by the fact that the logical language and the vocabulary are defined as a disjoint union. There is an equivalent operation on formal contexts (i.e., union of objects, union of attributes), also called the disjoint union.

Like the functor *Prod*, the functor *DisjointUnion* has no proper data structure, and only relies on the data structures of its argument contexts. It behaves like a switch, redirecting each operation to one argument context, depending on the parameter formula. For instance, the operation $K.show(f)$ is defined as $K_1.show(f)$ if $f \in L_1$, and as $K_2.show(f)$ if $f \in L_2$. The operation $K.axiom(f, g)$ is defined only when the two formulas f, g belong to the same logic L_1 or L_2. Hence, the worst case time complexity of those operations is the *maximum* of the worst case time complexities in the argument contexts.

4.5 The Root Context Functor

There remains a gap between the previous logical context functors, and the need in most LIS to describe objects with several properties, and to combine features by boolean connectives in queries. This gap is filled by a functor that is called $Root(K_1)$ because it is designed to be used as the outermost functor in a composition of functors. Another function of that logical context functor is the application of the *Closed World Assumption* (CWA) that says that every object that is not instance of a property f is an instance of its negation $\neg f$. This makes querying more intuitive as boolean connectives then match set operations: e.g., the extent of a disjunction $q_1 \vee q_2$ is the union of the extents of q_1 and q_2. The functor $Root(K_1)$ may look artificial, but it is in fact an idiomatic composition of more primitive functors (defined in [FR06]): $Prop(Bottom(Single(Multiset(K_1))))$, where *Prop* applies the Boolean closure, *Single* applies the CWA, and *Multiset* allows for (multi)sets of properties on objects.

Definition 10 (root logic). *Let* $\mathcal{L}_1 = (L_1, \sqsubseteq_1)$ *be a logic. The* root logic *Root* $(\mathcal{L}_1)$ *is defined as a logic* $\mathcal{L} = (L, \sqsubseteq)$*, where:*

– *$L = L_d \cup L_q$, where*
  - *L_d is the set of finite subsets $\{f_1, \dots, f_p\} \subseteq L_1$ (used for descriptions),*
  - *L_q is the smallest set that contains L_1, and such that for every $q_1, q_2 \in L_q$, the formulas $q_1 \wedge q_2, q_1 \vee q_2, \neg q_1$ also belong to L_q (used for queries),*

– *the subsumption $\sqsubseteq$ is characterized by the following inference rules, for every $f_1, \dots, f_p \in L_1$, $d \in L_d$, $q_1, q_2 \in L_q$:*
  1. *if $f_1 \sqsubseteq_1 f_2$ then $f_1 \sqsubseteq f_2$,*
  2. *if $\exists f_i \in d : f_i \sqsubseteq_1 f_2$ then $d \sqsubseteq f_2$,*
  3. *if $d \not\sqsubseteq q_1$ then $d \sqsubseteq \neg q_1$,*
  4. *if $d \sqsubseteq q_1$ and $d \sqsubseteq q_2$ then $d \sqsubseteq q_1 \wedge q_2$,*
  5. *if $d \sqsubseteq q_1$ or $d \sqsubseteq q_2$ then $d \sqsubseteq q_1 \vee q_2$.*

The last three inference rules are justified by the fact that descriptions are understood under the CWA [FR04]. Subsumption can also be defined between queries, but we can save it because it is useful neither for querying, nor for navigation. Indeed, the navigation vocabulary of the argument context of *Root* is sufficient because boolean connectives can be introduced through the navigation process [Fer09].

Definition 11 (root context). *Let $K_1 = (\mathcal{O}_1, \mathcal{L}_1, X_1, d_1)$ be a logical context. The* root context *$Root(K_1)$ is defined as a logical context $K = (\mathcal{O}, \mathcal{L}, X, d)$, where $\mathcal{O}$ is a partition of $\mathcal{O}_1$ ($\bigcup \mathcal{O} = \mathcal{O}_1$, and $\forall o, o' \in \mathcal{O} : o \cap o' = \emptyset$), $\mathcal{L} = Root(\mathcal{L}_1)$, $X = X_1$, and $d(o) = \{d_1(o_1) \mid o_1 \in o\}$.*

An object of the root context is represented by a set of objects in the argument context, each holding a property of the root object. In the following, $\overline{o_1}$ denotes the root object that contains $o_1 \in \mathcal{O}_1$; and by extension, $\overline{O_1}$ denotes the set of all root objects that contain any element of $O_1 \subseteq \mathcal{O}_1$. The proper data structures of *Root* are limited to a table defining each root object as a subset of $\mathcal{O}_1$, and another table from each object $o_1 \in \mathcal{O}_1$ to its root object $\overline{o_1}$.

The operation $K.add(o, \{f_1, ..., f_p\})$ consists in, for each f_i, creating a new object o_i to be added to $\mathcal{O}_1$, calling the operation $K_1.add(o_i, f_i)$, and defining the root object o as the set $\{o_i \mid 1 \le i \le p\}$. Its worst case time complexity is therefore p times the worst case time complexity of the operation $K_1.add$. The operations $K.show$ and $K.axiom$ apply only to formulas of the argument context K_1, and so can be directly transmitted to K_1; their complexities are unchanged.

The computation of the extent of a query follows the definition of subsumption between descriptions and queries. For every $f_1 \in \mathcal{L}_1$, and $q_1, q_2 \in L_q$:

$$\begin{aligned} K.ext(f_1) &= \overline{K_1.ext(f_1)}, & K.ext(q_1 \wedge q_2) &= K.ext(q_1) \cap K.ext(q_2), \\ K.ext(\neg q_1) &= \mathcal{O} \setminus K.ext(q_1), & K.ext(q_1 \vee q_2) &= K.ext(q_1) \cup K.ext(q_2). \end{aligned}$$

The complexity of this operation depends on the number k of atoms in the query: there are k calls to $K_1.ext$, and $(k-1)$ set intersections in $O(n)$ on resulting extents. The computation of increments is based on the fact that, if an increment is frequent in K, it is also frequent in K_1 because for every object $o \in \mathcal{O}$ and $x \in X$: if $o \in K.ext(x)$ then $\exists o_1 \in o : o_1 \in K_1.ext(x)$. Because the reverse is not true, the increments computed in K_1 must be checked to be increments in K.

$$K.incrs(x, O, m) =$$
$$\{(y, O') \mid (y, O'_1) \in K_1.incrs(x, K_1.ext(x) \cap \bigcup O, m), O' = \overline{O'_1}, |O'| \geq m\}$$

The additional cost for computing those increments is limited to computing O' for each increment coming from the argument context: $O(npi)$.

5 Applications

We now present how the context examples presented in Section 3 can be defined with logical context functors. The first context K_1 is a collection of documents described by various properties such as title, authors, publisher. Because each property can be represented by a string-valued attribute, that context is defined as $K_1 = Root(Prod(Attr, String))$, where $Attr$ is an instance of the functor $Taxo$. The use of $Taxo$ for representing attribute names adds the ability to abstract similar attributes into a more general attribute: e.g., "author" and "editor" can be grouped under "person"; "title", "subtitle", and "keywords" can be grouped under "subject". The use of $String$ enables the automatic extraction of keywords from titles. The second context K_2 is a collection of documents described by pairs (user, tag), meaning that some user put some tag on it, where user and tag terms can be organized into two taxonomies. That context is defined as $K_2 = Root(Prod(User, Tag))$, where $User$ and Tag are two instances of $Taxo$. In order to describe and retrieve documents by both valued attributes and (user,tag) pairs, we can defined a context $K_3 = Root(DisjointUnion(Prod(Attr, String), Prod(User, Tag)))$.

Table 1 presents the complexities of the five operations on the contexts K_1 and K_2, depending on the use of a logic cache or logical context functors. Those complexities are expressed in function of the number n of objects, the number p of properties per object in $Root$, the maximum height h of taxonomies, and the maximum length s of strings. In practice, p, h, s are often bounded, so that complexities can be expressed in term of the number n of objects only. We note that the complexity for adding an object, or showing a formula, is in $O(1)$ or $O(\ln(n))$ with functors instead of $O(n)$ with a logic cache. This achieves our objective to make the filling of a context in $O(n)$ or $O(n \ln(n))$ instead of $O(n^2)$. About the browsing operations, the complexity $O(n)$ for computing the extent and each increment is obtained, like with logic caches.

Even if detailed experiments remain to be done, first experiments are very conclusive w.r.t. the efficiency of functors compared to logic caches. We used K_1 for representing BibTEX files, where entries are objects, and fields are valued attributes. For a file with 1500 entries, it took 46s on a standard laptop to build the context with the logic cache, while it took only 26s with functors, comprising the additional computation of all maximal substrings. Without this additional computation, the time is about 10 times less, hence a 20 fold speed up. We

Table 1. Complexities of LIS operations on contexts K_1 and K_2, depending on the use of a logic cache or logical context functors

	K_1		K_2	
operation	logic cache	functors	logic cache	functors
axiom		phn		phn
add	p^2s^5n	$ps\ln(psn)$	p^2h^5n	ph
show	ps^3n	s	ph^3n	1
ext	n	$s + pn$	n	pn
incrs	ni	pni	ni	pni

managed to build the context of a file with 30,000 entries in about 1,000s, while this is not possible with a logic cache, even without generating any navigation feature.

6 Conclusion

Logical context functors are introduced as reusable components for composing logical contexts according to the structure of its formulas. They allow for efficient implementations of LIS because each functor can use specific data structures and algorithms. The genericity of the LIS framework is retained by the ability to freely compose functors. Examples of composed contexts are given for string-valued attributes, taxonomies, and pairs of taxonomic terms. The filling of a context is shown to be in $O(n)$ or $O(n\ln(n))$ instead of $O(n^2)$, comprising the computation of a rich navigation vocabulary on strings. Several taxonomies can be updated incrementally and efficiently. Functors form an open collection so that new functors can be designed and added to the collection, indepently of existing functors. Moreover, if a better data structure or algorithm is found, it can be integrated in an existing functor without impacting other functors or applications using it. We are developping functors for numbers, dates, intervals, and functors can be composed to represent vectors, lists and trees.

References

[BCM$^+$03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

[CES03] Cole, R.J., Eklund, P.W., Stumme, G.: Document retrieval for email search and discovery using formal concept analysis. Journal of Applied Artificial Intelligence 17(3), 257–280 (2003)

[CM00] Chaudron, L., Maille, N.: Generalized formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867. Springer, Heidelberg (2000)

[DVE06] Ducrou, J., Vormbrock, B., Eklund, P.W.: FCA-based browsing and searching of a collection of images. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS, vol. 4068, pp. 203–214. Springer, Heidelberg (2006)

[Fer07] Ferré, S.: The efficient computation of complete and concise substring scales with suffix trees. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS, vol. 4390, pp. 98–113. Springer, Heidelberg (2007)

[Fer09] Ferré, S.: Camelis: a logical information system to organize and browse a collection of documents. Int. J. General Systems 38(4) (2009)

[FR00] Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 371–384. Springer, Heidelberg (2000)

[FR04] Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management 40(3), 383–419 (2004)

[FR06] Ferré, S., Ridoux, O.: Logic functors: A toolbox of components for building customized and embeddable logics. Research Report RR-5871, INRIA, 103 p. (March 2006)

[GK01] Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS, vol. 2120, pp. 129–142. Springer, Heidelberg (2001)

[GMA93] Godin, R., Missaoui, R., April, A.: Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. International Journal of Man-Machine Studies 38(5), 747–767 (1993)

[GW89] Ganter, B., Wille, R.: Conceptual scaling. In: Roberts, F. (ed.) Applications of combinatorics and graph theory to the biological and social sciences, pp. 139–167. Springer, Heidelberg (1989)

[GW99] Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer, Heidelberg (1999)

[Mog89] Moggi, E.: A category-theoretic account of program modules. In: Dybjer, P., Pitts, A.M., Pitt, D.H., Poigné, A., Rydeheard, D.E. (eds.) Category Theory and Computer Science. LNCS, vol. 389. Springer, Heidelberg (1989)

[PS99] Prediger, S., Stumme, G.: Theory-driven logical scaling: Conceptual information systems meet description logics. CEUR Workshop Proceedings, vol. 21, pp. 46–49. CEUR-WS.org (1999)

[Sac00] Sacco, G.M.: Dynamic taxonomies: A model for large information bases. IEEE Transactions Knowledge and Data Engineering 12(3), 468–479 (2000)

[TCH06] Tane, J., Cimiano, P., Hitzler, P.: Query-based multicontexts for knowledge base browsing: An evaluation. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS, vol. 4068, pp. 413–426. Springer, Heidelberg (2006)

In Search of Semantic Compositionality in Vector Spaces[⋆]

Eugenie Giesbrecht

FZI Research Center for Information Technology, Karlsruhe, Germany

Abstract. In spite of the widespread usage of geometric models of meaning in computational linguistics and information retrieval research, they have been until recently mostly utilized for modeling lexical meaning. The ability to deal with concept combination, however, is the essential capacity of human language, and any semantic theory should be able to handle it.

Making use of Word Space Models (Schütze 1998) and Random Indexing (Sahlgren 2005), we explore the hypothesis that compositional meaning can be captured in such models by adopting a number of mathematical operations for vector composition (summation, component product, tensor product and convolution) to model semantic composition in a multiword unit identification task.

1 Introduction

Symbolic or logical approaches to meaning, following the traditions of Montague's semantics and known in the linguistics community as *compositional semantics* (Dowty 1981), have been concerned prevalently with composition of individual units into sentences and not with the meaning of those individual units. The principle of compositionality, commonly attributed to Frege (1914), claims that the meaning of a sentence is a function of the meanings of its parts.

Within this logical tradition of semantics, the meaning of a sentence "Peter washed the car" could be represented as: $washed(Peter, car)$ - which can be obtained by the composition of the constituents $\lambda x \lambda y.washed(x, y)$, $Peter$ and car extracted word-wise from the original sentence.

Such constructions are good at conveying the structural or grammatical properties of language, but unfortunately they do not tell us anything about the meaning of individual units except that some X washed some Y, and this X is in this case *Peter* and Y is *car*. We still may have no idea though, what a *car* is, or how *washed* is different from *polluted*. It is just assumed that there is a referent in the external world or in the speaker's mind. Lexicon is much more empirical than grammar and is therefore harder to formalize (Widdows 2008). According to Jones and Sinclair (1974), "one of the troubles with studying lexis

⋆ This work was supported by German "Federal Ministry of Economics" (BMWi) under the project Theseus (number 01MQ07019).

was making a start somewhere", and a brilliant start was made by geometrical models of meaning, realized in vector spaces.

Vector Space Models (VSMs) embody the so-called distributional hypothesis of meaning, the motto of which is that a word is known "by the company it keeps" (Firth 1957). The meaning of words is thereby defined by contexts in which they occur (in analyzed text collections). The context can be both local, i.e., just the immediate neighbours, as well as global, e.g., a sentence or a paragraph or the whole document. Vector Space Models have been rather intensively and successfully applied for modeling meaning in information retrieval and computational linguistics. Typically the global context is used for modeling the words' meanings within the information retrieval paradigm. To do this, a term-document matrix is constructed and the meaning of the terms is defined by the documents they co-occur in. Table 1 shows an example of such space that consists of three documents and 5 key terms. The numbers in the columns denote how often given terms occur in corresponding documents, e.g., the word *automobile* appears 2 times in *document1* and not at all in *document2*. Usually, more sophisticated weighting schemes are used. According to the vector space in Table 1, the meanings of the words *car* and *automobile* would be not related at all as they never occur in the same documents.

Table 1. Vector Space Model (VSM)

	document1	**document2**	**document3**
automobile	2	0	3
car	0	2	0
garage	1	3	0
factory	0	0	3

In contrast, research in computational linguistics concentrated mostly on modeling the local contexts in that word-by-word matrices are built from text collections. The latter are also called Word Space Models (Schütze 1998). In this framework, the meaning is modeled as an n-dimensional vector, where the dimensions are defined by the co-occurring words within a predefined *context window*. Thus, we can see from Table 2 that the word *car* co-occurs with *garage* and wash in 3 cases in a given collection within, say, a context of 15 words. In this WSM, the words *car* and *automobile* get closer to each other as they co-ocur with the same words at least in a couple of contexts, but this may still be not enough to detect that they are synonyms.

Usually, dimensionality reduction techniques, like Singular Value Decomposition (Deerwester et al. 1990), often referred to as Latent Semantic Analysis (LSA), are applied to the resulting Word Spaces as those are mostly sparse. LSA is a low-rank approximation of the original vector space matrix. Lowering the rank of the matrix is a means of removing extraneous information or noise. By rank reduction, we cut off the dimensions that do not contribute a lot to the meaning of terms. Some information is lost, but the most important one is preserved and emphasized. Therefore, similar words get closer to

Table 2. Word Space Model (WSM)

	street	**garage**	**wash**	**exhibition**
automobile	0	1	1	3
car	4	3	3	1
garage	1	1	1	0
factory	0	1	0	0

each other in vector spaces, although the connections between them may not have been explicitly present in the original data. Thus, if we reduce our matrix (s. Table 2) from 4 to 2 dimensions, the words *car* and *automobile* will get very close or "similar" to each other in the vector space. Thereby second-order representations are achieved. This kind of dimensionality reduction has been shown to improve performance in a number of text-based domains (Berry et al. 1999).

In spite of the widespread usage of Vector and Word Space Models, they have been predominantly used for meaning representation of single words and thereby have been the mainstream of interest in lexical semantics since the 90s. Until recently, little attention has been paid to the way of modeling more complex conceptual structures with such models, which is a crucial barrier for semantic vector models on the way to model language (Widdows 2008). As a consequence, an emerging area of research that receives more and more attention among the advocates of distributional models are the methods, algorithms and evaluation strategies for modeling of compositional meaning within VSM framework.

VSMs are mathematically well-understood, have been shown to work well in modeling lexical meaning, allow integration of uncertainty, and are in line with empirical results from cognitive science (Gärdenfors 2004). The crucial missing piece is how to model compositionality. We address this problem here by examining mathematical models of vector composition in Section 4 and proposing an evaluation framework for compositional models that is inherited from the research on collocations (cf. Sections 3 and 5). In the end, we assess and compare the performance of compositionality operators and provide an outlook for future research.

2 Related Work

Untill recently, the "bag-of-words" approach has been used as a default to get the meanings of phrases and sentence in vector spaces (Landauer and Dumais 1997, Deerwester et al. 1990). The latter consists of just adding up the individual vectors of the words to get the meaning of a phrase or sentence. The vector sum operation can obviously not serve as an adequate means of semantic composition, as word order information is ignored, whereby *Peter washed the car* and *the car washed Peter* would mean the same.

Among the early attempts to provide more compelling combinatory functions to capture word order information and the non-commutativity of linguistic compositional operation in VSMs is the work of Kintsch (2001) who is using though a more sophisticated but still just addition operation to model predicate-argument structures in VSM.

There are approaches under way to work out a combined framework for meaning representation using both the advantages of formal and distributional methods. Thus, Clark and Pulman (2007) aim at a conceptual model which unites symbolic and distributional representations in the similar way as Cognitive Science research does with symbolic and connectionist models, namely by adopting the tensor product representations (Smolensky 2006, Gärdenfors 2004).

They propose to combine the symbolic representation, such as a parse tree on the sentence level, with the distributional co-occurrence signature of the individual words by means of a tensor product. They leave open the practical question of obtaining the vector representation for the dependency relations, such as *subject*, *object* and *predicate* which are essential for the model and without which the suggested framework could not be evaluated.

Recent work which addresses the problem of compositionality in Vector- or Word Space Models include Mitchell and Lapata (2008) and Widdows (2008).

Mitchell and Lapata (2008) introduce a number of addition and multiplication models for vector combinations and evaluate the proposed models on a sentence similarity task (Kintsch 2001). In a nutshell, their conclusion is that multiplicative models are superior to the simple additive models for the similarity task in question. The similarity rates in their evaluation are elicited from the human subjects, with rather small inter-annotator agreement rate of 0.40 (Spearman's rho coefficient). The low agreement rate suggests that the used "gold standard" may not have been reliable enough to serve as a basis for evaluation.

Widdows (2008) proposes a number of more advanced vector operations well-known from quantum mechanics, such as tensor product and convolution, to model composition in vector spaces. Furthermore, he presents two small but intuitive experiments that show the ability of the VSM models to reflect the relational and phrasal meanings on simple similarity tasks. The latter include finding the relations of the kind *Moscow is related to Russia* like *Berlin - to Germany*, or finding out whether *eating apples* is different from *cooking tomatoes* in vector spaces. Both Widdows (2008) and Clark and Pulman (2007) provoke further research in the area by summarizing a number of possible mathematical operators on vectors, thereby inspiring interfaces between language and quantum research, which promises to be a fruitful area.

We take the baton from Widdows (2008) and evaluate empirically the applicability of suggested mathematical vector operations to model semantic composition. As a test-bed for our experiments, we draw on research in multiword units extraction, which has been a long tradition in computational linguistics as identifying non-compositional (or idiomatic) units is an important subtask for any computational system (Sag et al., 2002).

3 Multiword Units (MWUs), Word Spaces and Compositionality

Following Schone and Jurafsky (2001), we define a multiword unit (MWU) as a connected collocation, i.e., a sequence of neighbouring words whose exact and unambiguous meaning cannot be derived from the meaning of its components. *Spill the beans* or *hot dog* are examples of such MWUs. Therefore, a MWU either has a completely opaque meaning, or its constituent words acquire some other nuance of meaning when they are used together, thereby making the expression as a whole non-compositional. Thus, *spill* is no more about *slopping* when it comes together with *the beans*, it is about *revealing (secrets).* In contrast, *buy a ticket* is about *buying* and *ticket* together and is perfectly compositional. Figurative MWUs have always posted a problem for compositional theories of language and have been used as an objection to the principle of compositionality of human language within symbolic approaches.

There have been a number of research efforts looking for methods of automatic identification of such non-compositional MWU with statistical association measures (Evert and Krenn 2000, Evert 2004, Lin 1999) or by means of Latent Semantic Analysis (Schone and Jurafsky 2001, Baldwin et al. 2003, Katz and Giesbrecht 2006). Research in the latter field was a motivation for this work.

Schone and Jurafsky (2001) and Katz and Giesbrecht (2006) explored detection of non-compositional phrases by means of comparing the co-occurrence signatures of a multiword unit as a whole and those of the composed vectors of its constituents. The main assumption in all similar experiments is that compositional MWUs appear systematically in contexts more similar to those in which their component words appear than do non-compositional MWUs, which is a general assumption in the spirit of distributional semantics. Figure 1 illustrates such a vector space in two dimensions. Note that the meaning vector for the MWU *yellow press* is quite similar to that for *gossip* but distant from *yellow*, while the meaning vector for *yellow banana* would be, in contrast, much closer to *yellow*. Indeed *yellow press* is a non-compositional idiom meaning 'newspapers that publish gossip about celebrities'.

Katz and Giesbrecht (2006) showed that the local context of a MWU could reliably distinguish idiomatic uses of MWU from non-idiomatic uses. It was shown that LSA vectors for compositional and non-compositional uses of an idiom (manually annotated) were orthogonal, i.e., unrelated.

However, both of the above mentioned works define the estimated compositional meaning vector by taking it to be the sum of the component vectors, i.e., the compositional meaning of an expression $< word1, word2 >$ consisting of two words is taken to be sum of the vectors for the corresponding words or $< word1 > + < word2 >$. They recognize that the composed vector is clearly nowhere near a perfect model of compositional meaning, but it proved to be just enough to test the hypothesis.

We build here upon the work of Katz and Giesbrecht (2006) and explore more advanced mathematical operations on vectors, suggested by Widdows (2008), as an approximation of "semantic composition" by adopting their evaluation

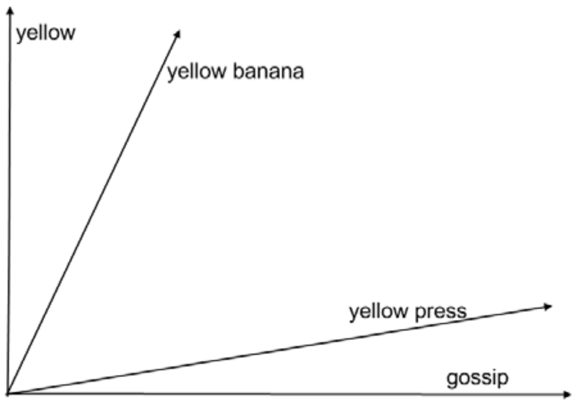


Fig. 1. Two-dimensional word space

paradigm. In particular, we are looking for an answer to the question whether simply applying more advanced mathematical operations on vectors would be enough to achieve better models of the semantic compositionality in vector spaces.

4 Compositional Models

Let *w1w2* denote the composition of two vectors *w1* and *w2*. In the following, we define the operations for vector compositionality models that we test later. The estimated compositional meaning vector *w1w2* is calculated by taking it to be:

1. The sum of the meaning vectors of the parts, i.e., the compositional meaning of an expression *w1w2* consisting of two words is taken to be sum of the meaning vectors for the constituent words *w1* and *w2*: *(w1w2)_i = w1_i+w2_i*;

   Thus, the "compositional" vector for *yellow press* in this case would be the sum of the vectors for *yellow* and *press*.

2. The simplified multiplicative model as it is defined in Mitchell and Lapata (2008): under the assumption that only the ith component of *w1* and *w2* contribute to the ith component of *w1w2*, we can formulate vector multiplication operation as: $(w1w2)_i = w1_i \cdot w2_i$;

3. The tensor product: if the vector of the word *w1* has components $w1_i$ and the vector of the word *w2* has components $w2_j$, then the tensor product $(w1 \otimes w2)$ is a matrix whose ij^{th} entry is $w1_i w2_j$ (cf. Widdows 2008);

4. The convolution product, which is also a kind of vector multiplication that results in the third vector of dimensionality $(m + n - 1)$. Given two vectors

$w1 = [w1_1, w1_2, w1_{\dots}, w1_m]$ and $w2 = [w2_1, w2_2, w2_{\dots}, w2_n]$, their convolution $(w1 * w2)$ is defined as $(w1w2)_i = \sum_j w1_j w2_{i-j+1}$.

For computing meaning similarity for vector addition, component multiplication and convolution, we use the standard measure of cosine of the angle between two vectors (the normalized correlation coefficient) as a metric (Schütze 1998, Baeza-Yates and Ribeiro-Neto 1999), which corresponds for normalized unit vectors to a scalar product of those. In this metric, two expressions are taken to be unrelated if their meaning vectors are orthogonal (the cosine is 0) and synonymous if their vectors are parallel (the cosine is 1). For the tensor product, the natural similarity measure is the inner product on tensors and is defined as the product of the similarities of the constituents: $(w1_1 * w2_1) \times (w1_2 * w2_2)$ (cf. Widdows 2008). The quantitative interpretation of this metric corresponds to that of a scalar product, i.e., the higher the similarity score, the more related the components.

5 Experimental Setup

In our work we make use of the Word Space Model. In this framework, the meaning of a word is modeled as an n-dimensional vector with dimensions being its co-occurrence signature, derived via Random Indexing (Sahlgren 2005). Random Indexing (RI) is a computationally tractable analogue of Singular Value Decomposition (Deerwester et al. 1990) where Word Space Models are built iteratively, unlike in the LSA scenario, where first a complete sparse WSM of a text collection is constructed and then the dimensionality of a WSM is reduced by means of SVD. RI is claimed to achieve comparable results to LSA (cf. Sahlgren 2005). As to our knowledge, this specific technique has not been tested yet for the task of MWU identification.

We build our WSM on the excerpt of a local German newspaper corpus[1]. As our MWU test set we use a database of German (Preposition)-Noun-Verb (PNV) pairs available as an example data collection in the UCS-Toolkit[2]. From this database only word combinations with frequency of occurrence more than 30 in the corpus were considered.

The Semantic Vectors package (Widdows 2008) was used to build the context vectors of reduced dimensionality. We use a window context of 15 words and limit the dimensionality to 100, resulting in 100 dimensional "meaning"-vectors for each word. In our experiments, MWUs were assigned meaning vectors as a whole, using the same procedure. In order not to bias the contribution of the constituent words, the meaning vectors for these words were always computed from contexts in which they appear alone (that is, not in the local context of the other constituent). Table 3 illustrates a possible resulting matrix, which indicates that, e.g., the words *gossip* and *celebrity* occur 20 times with *yellow_press* within a distance of 7 words before or 7 words after the *yellow press*.

[1] Süddeutsche Zeitung (SZ) corpus for 2003 with about 42 million words.

[2] www.collocations.de

Table 3. An example of the MWU word space

	dim1=gossip	dim3=celebrity	dim4=banana	...	dim100=resources
yellow	0	0	20	...	0
press	1	3	0	...	15
yellow_press	20	20	0	...	1

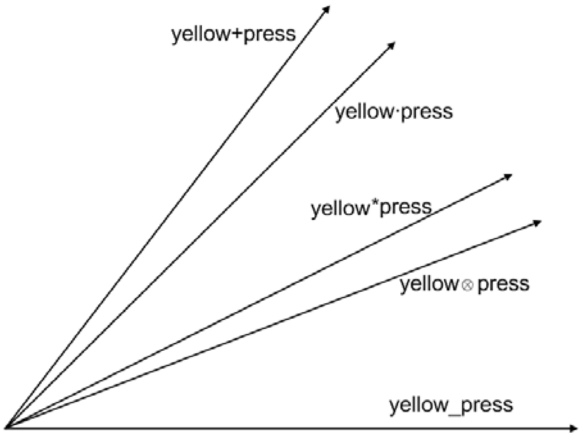


Fig. 2. Vectors for composition operations vs that of the MWU in a 2-dimensional vector space

Our task was to compare the actual vector of a multiword unit with that of the "composed" vector of its constituents, whereas the "composed vector" was defined by the models of compositionality described in Section 4. Figure 2 exemplifies the idea behind the overall procedure.

6 Evaluation

Evaluation is a challenge when it comes to the analysis of compositionality. As to our knowledge, there are just a few evaluation attempts in this field and most of the relevant research has concentrated on the conceptual modeling. The only evaluation in the field was done on sentence similarity judgements. The strategy we suggest in this paper is a bit different.

To evaluate the method, we use the manually annotated collocations database described by Krenn (2000) as our gold standard. This collection includes collocations that have been manually classified into Support Verb Constructions(SVC), figurative expressions, or neither of the two. SVC are (preposition-)noun-verb constructions where a noun provides the main semantic contribution to the meaning of the whole phrase, like in "Peter *took a walk*", or an example in German could be "Peter hat das Problem *in Angriff genommen*" The whole word combination in this case is neither non-compositional nor can it be called

Table 4. Similarity values for tested compositionality models

ADDITION	< 0.2	< 0.3	< 0.4	< 0.5
Precision	0.125	0.28	0.29	0.25
Recall	0.05	0.53	0.84	0.88
F-measure	0.09	0.37	0.43	0.40
MULTIPLICATION	< 0.001	< 0.01	< 0.02	< 0.03
Precision	0.19	0.20	0.19	0.19
Recall	0.47	0.79	0.89	1.00
F-measure	0.27	0.39	0.31	0.31
TENSOR	< 0.03	< 0.05	< 0.1	< 0.15
Precision	0.21	0.29	0.31	0.28
Recall	0.16	0.37	0.84	1.00
F-measure	0.18	0.325	0.45	0.44
CONVOLUTION	< 0.01	< 0.1	< 0.2	< 0.26
Precision	0.22	0.20	0.22	0.25
Recall	0.26	0.47	0.79	1.00
F-measure	0.24	0.28	0.35	0.40

compositional. To be on the safe side, the current evaluation is based solely on the phrases annotated as figurative, as they are per se non-compositional. The latter constitute 19% of our test set (19 out of 100).

The idea behind our evaluation strategy is to use these non-compositional collocations to compare how good different vector composition models are at identifying them. This should give us a clue whether just using a more advanced mathematical operator is enough to reproduce semantic composition in language.

The resulting vector similarity values for tensor product range from -0.009 to 0.55; for vector sum, the cosine values are between 0.04 and 0.79; the products range from -0.009 to 0.03; and finally, convolution ranges from -0.04 to 0.66[3]. Since we cannot directly compare the values between the different composition operations, important are the comparisons within the individual models.

In computational linguistics, one straightforward way of doing this is by means of precision and recall.

Precision is defined in this case as the proportion of the phrases under given cosine similarity thresholds that are multiword units. Recall is the proportion of all given multiword units under the cosine threshold. As there is generally a trade-off between the two measures, the F-measure (Manning and Schütze 1999) is often used instead, which is a weighted harmonic mean of precision and recall. Table 4 gives an overview of precision, recall and F-score values for different cut-offs of a similarity value for the evaluated composition models.

The results in Table 4 and in Fig. 3 show that the tensor product does a consistently better job at recognizing non-compositional multiword units. Though the precision of all the models seems to be rather small at first sight, it is worth

[3] The biggest possible value is 1.0. Remember, the higher the similarity score, the more related the components.

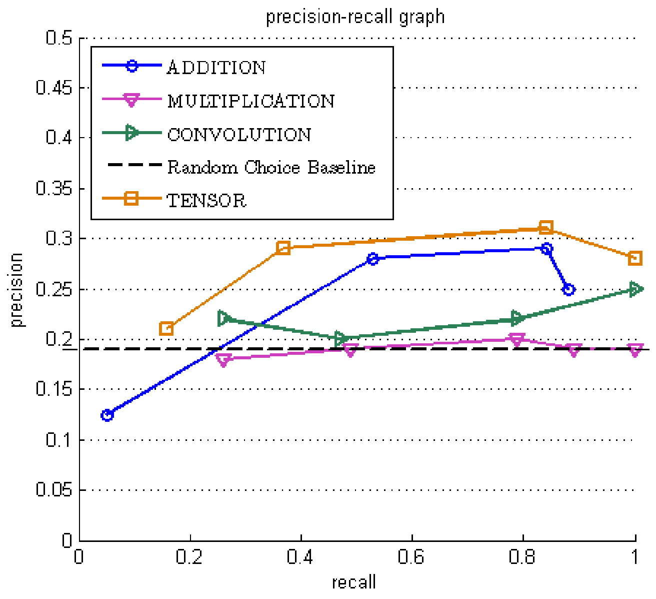


Fig. 3. Precision-recall diagram

mentioning that it is still significantly better than expected by chance alone for almost all models. The outcomes of other methods are rather dispersed in the vector space, especially these of vector addition. Our preliminary results are in line with those of Widdows (2008) who showed on a couple of examples that the tensor product performs well at discovering unlikely word combinations on a phrase similarity task.

7 Conclusions and Outlook

To summarize, we address in this paper the problem of expressing compositionality in vector spaces and propose an alternative evaluation framework for assessing the models of vector composition. The latter is adopted from the recent research on collocations. We compute an approximation of the compositional meaning of MWUs with the help of different mathematical vector combination methods and compare this with the vector of the expression as it is used on the whole. Our preliminary findings prove that the more advanced compositional operators, like tensor products, lead to better results than just using vector addition, which is still the most common operator for vector composition. Though the results are encouraging, there is definitely a need for further explorations on the way to a complete compositionality model in vectors spaces. Like Widdows (2008), who indirectly inspired us to perform this evaluation with his introductory paper on semantic vector products, we are not in the position yet to commit ourselves to any representative statements. It is obvious though, that just using a different mathematical operator will not do the job; and it is not foreseeable at the moment how structural peculiarities of language can be reflected in such models with current approaches.

We believe that the way to go is to merge the strengths of distributional and symbolic approaches. In the future, we plan to investigate advanced compositionality models in vector spaces, especially the tensor product, in an integrated scenario with symbolic structural representations, like conceptual graphs (Sowa 2000), or as proposed by Clark and Pulman (2007).

Language has structure, and the meaning of a sentence in language is dependent on lexis as well as on grammar. Consequently, any successful meaning theory should be able to integrate these.

A further important open issue which could have received more attention both in symbolic and in subsymbolic communities is that compositionality appears to come in degrees (Bannard et al. 2003).

Acknowledgements. Special thanks go to Radomir Zugic and Sebastian Rudolph who have generously spared the time to read and comment on this work. Many thanks also to the anonymous reviewers for their constructive critique.

References

Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press, Addison-Wesley (1999)

Baldwin, T., Bannard, C., Tanaka, T., Widdows, D.: An empirical model of multiword expression decomposability. In: Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment (2003)

Bannard, C., Baldwin, T., Lascarides, A.: A statistical approach to the semantics of verb-particles. In: Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, Sapporo, Japan (2003)

Berry, M., Drmac, Z., Jessup, E.R.: Matrices, Vector Spaces, and Information Retrieval. SIAM Review 41(2), 335–362 (1999)

Clark, S., Pulman, S.: Combining symbolic and distributional models of meaning. In: Proceedings of the AAAI Spring Symposium on Quantum Interaction, Stanford, CA, pp. 52–55 (2007)

Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)

Dowty, D.R., Wall, R., Peters, S.: Introduction to Montague Semantics. Kluwer Academic Publishers, Dordrecht (1981)

Evert, S., Krenn, B.: Methods for the qualitative evaluation of lexical association measures. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (2001)

Evert, S.: The statistics of word cooccurrences: word pairs and collocations. Ph.D. Thesis, University of Stuttgart (2004)

Firth, J.: A synopsis of linguistic theory 1930-1955. Studies in Linguistic Analysis, pp. 1–32. Longman (1957)

Frege, G.: Letter to Jourdain. In: Gabriel, G., et al. (eds.) Philosophical and Mathematical Correspondence, Chicago University Press 1980 (1914)

Gärdenfors, P.: Conceptual Spaces: The Geometry of Thought. The MIT Press, Cambridge (2004)

Jones, S., Sinclair, J.M.: English lexical collocations. Cahiers de Lexicologie 24 (1974)

Katz, G., Giesbrecht, E.: Automatic identification of non-compositional multiword expressions using Latent Semantic Analysis. In: Proceedings of the ACL/Coling Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties (2006)

Kintsch, W.: Predication. Cognitive Science 25(2) (2001)

Krenn, B.: The usual suspects: data-oriented models for the identification and representation of lexical collocations. In: Dissertations in Computational Linguistics and Language Technology, German Research Center for Artificial Intelligence and Saarland University, Saarbrücken, Germany (2000)

Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: the Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review 104, 211–240 (1997)

Lin, D.: Automatic identification of noncompositional phrases. Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (1999)

Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)

Mitchell, J., Lapata, M.: Vector-based models of semantic composition. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 236–244 (2008)

Sag, I.A., Baldwin, T., Bond, F., Copestake, A., Flickinger, D.: Multiword expressions: a pain in the neck for NLP. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, p. 1. Springer, Heidelberg (2002)

Sahlgren, M.: An Introduction to Random Indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE, Copenhagen, Denmark, August 16 (2005)

Schone, P., Jurafsky, D.: Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In: Proceedings of Empirical Methods in Natural Language Processing, Pittsburgh, PA (2001)

Schütze, H.: Automatic word sense discrimination. Computational Linguistics 24(1), 97–124 (1998)

Smolensky, P., Legendre, G.: The Harmonic Mind: from Neural Computation to Optimality-Theoretic Grammar. MIT Press, Cambridge (2006)

Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove (2000)

Widdows D.: Semantic vector products: some initial investigations. In: Proceedings of the Second AAAI Symposium on Quantum Interaction (2008)

Widdows, D., Ferraro, K.: Semantic vectors: a scalable open source package and online technology management application. In: Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008) (2008)

Frequent Itemset Mining for Clustering Near Duplicate Web Documents

Dmitry I. Ignatov and Sergei O. Kuznetsov

Higher School of Economics, Department of Applied Mathematics
Kirpichnaya 33/5, Moscow 105679, Russia
{skuznetsov,dignatov}@hse.ru

Abstract. A vast amount of documents in the Web have duplicates, which is a challenge for developing efficient methods that would compute clusters of similar documents. In this paper we use an approach based on computing (closed) sets of attributes having large support (large extent) as clusters of similar documents. The method is tested in a series of computer experiments on large public collections of web documents and compared to other established methods and software, such as biclustering, on same datasets. Practical efficiency of different algorithms for computing frequent closed sets of attributes is compared.

1 Introduction

Around 30% of documents in Internet have duplicates, which necessitates creation of efficient methods for computing clusters of duplicates [5,6,7,9,10,14,15,17,24,23]. The origin of duplicates can be different: from intended duplicating information on several severs by companies (legal mirrors) to cheating indexing programs of websites, illegal copying and almost identical spammer messages. Usually duplicates are defined in terms of similarity relation on pairs of documents: two documents are similar if a numerical measure of their similarity exceeds a certain threshold (e.g., see [5,6,7]). The situation is represented then by a graph where vertices are documents and edges correspond to pairs of the similarity relation. Clusters of similar documents are computed then as cliques or as connected components of such similarity graphs [7]. The main stages in finding clusters of duplicates are as follows (see, e.g., [7]): representing documents by sets of attributes, making concise document images by choosing subsets of images, defining similarity relation on document images, and computing clusters of similar documents. At the first stage of processing, after removing HTML-markup and punctuation marks, documents are turned into strings of words, which are, in turn, represented by sets of attributes. We have there two main different approaches, called syntactical and lexical. The syntactical approach consists in defining binary attributes that correspond to each fixed length substring of words (or characters). Such substrings are called shingles. Usually a shingle corresponds to a sequence of words and there are two parameters of shingling: the length of a shingle (the number of words in a shingle) and offset, the distance between the beginnings of two shingles. Each shingle is coded by a hash code (equal shingles have equal codes and it is unlikely that different shingles have same codes). Then, by means of a randomization scheme, a subset of shingles is chosen for a concise image of the document [5,6,7]. Such an approach is used in AltaVista search engine and (judging by patent [23]), in Google too. There are several principles for choosing number of shingles in an image: A fixed number, a logarithmic (as in Yandex mail service) number, linear number (each kth shingle), etc. In lexical methods representative words are chosen according to their significance of these words. Usually indices of significance are based on frequencies: those words whose frequencies lie in a certain interval (except for stop-words from a special list of about 30 stop-words with including articles, prepositions and pronouns) are taken: high frequency words can be noninformative and low frequency words can be either haphazard words that could have not appeared in a text or misprints.

In lexical methods such as I-Match [10] a large text corpus is used for generating lexicon, i.e., a set of representative words. A document is represented by the words that occur in the lexicon. In generation of the lexicon the words with lowest and highest frequencies are deleted, I-Match generates signature of a document (set of terms) and hash code of the document, where two documents get the same hash code with the probability equal to their similarity measure (according to the so-called cosine measure). As shown in [17], I-Match is sometimes instable to changes in texts, e.g., to *marginal* randomization of actually identical spammer messages. To avoid this drawback, besides one standard signature, one uses K more signatures, each of which is obtained by random deletion of certain amount of terms from the initial signature (i.e., all new signatures are subsets of the initial one). Two documents are considered to be almost duplicates if their images from K+1 signatures have *large* intersection in at least one of signature. In [17] the authors noted the similarity of this approach to the approach based on supershingles (concatenation of supershingles). In lexical method [15] the focus is towards the construction of a lexicon, a set of descriptive words, which should be concise, but cover well the collection. The occurrence of a word in a document image is robust with respect to small changes in the document. When document images are defined, one defines similarity relation on documents starting from a similarity measure which takes two documents to a number in the [0,1] interval depending on the amount of their common description units (shingles or words, in syntactical or lexical approaches, respectively). Then one chooses a threshold, exceeding which means large similarity of documents (the documents are near-duplicates). This metrics and a threshold define (symmetric and reflexive) similarity relation on document pairs. The similarity relation on document pairs determine clusters of near-duplicates. Definition of a cluster may also vary. A possible definition often used in practice, e.g., in Altavista search engine [7], is as follows: Consider the graph where Internet documents correspond to vertices and similarity relation corresponds to edges. Then a cluster of near duplicates is a connected component of such a graph. An advantage of such definition is efficiency of computation: a connected component may be computed in time linear in the number of edges. An obvious drawback here is that the relation *to be near duplicates* is not transitive, therefore absolutely different documents can occur in a cluster. The strongest definition of a cluster arising from a similarity relation is that based on a graph clique. This definition is more adequate than that based on the connected component, but is much harder computationally, since generation of maximal cliques is a classical intractable problem. These two extreme definitions of a cluster admit for a broad scope of intermediate formulations that realize trade-off between precision and complexity of computing the clusters. Other methods of cluster definition are based on variations of standard methods of cluster analysis, e.g., when clustering a new object uses the distance to the *mass center* of clusters. Methods of this type essentially depend on the sequence in which objects to be clustered arrive. As applied to the problem of clustering duplicates, this means that documents that occurred earlier determine stronger the structure of cluster than documents that occurred later.

In this paper we consider similarity not as a relation on the set of documents, but as an operation taking each two documents to the set of all common elements of their concise descriptions. Here description elements are either syntactical units (*shingles*) or lexical units (*representative words*). A cluster of similar documents is defined as a set of all documents with a certain set of common description units. A cluster of duplicates is defined as a set of documents where the number of common description units exceeds a certain threshold. In this paper we compare results of its application (for various values of thresholds) with the list of duplicates obtained by applying other methods to the same collection of documents. We studied the impact of the following model parameters on the result:

– The use of the syntactical or lexical methods for representing documents,
– the use of method "*n minimal elements in a permutation*" or "*minimal elements in n permutations*" [5,6,7] (the second method, having better probability-theoretical properties, has worse computational complexity.)
– shingling parameter,
– threshold value of similarity of document images.

One of our goals was to relate computation of pairwise similarity with generation of clusters so that, on the one hand, the obtained clusters are independent of the order of document occurrence (in contrast to methods of cluster analysis) and, on the other hand, they would guarantee real pairwise similarity of all document images in the cluster. To this end we employed an approach based on formal concepts: Clusters of documents are given by formal concepts of the context where objects correspond to description units (units of a language describing documents: shingles, lexical units, etc.) and attributes are document names. A cluster of *very similar documents* corresponds then to a formal concept such that the size of extent (the number of common description units of documents) exceeds a threshold given by parameter. Thus, generating very similar documents is reduced to the problem of Data Mining [21] known as generating *frequent closed itemsets.*

The rest of the paper is organized as follows. In the second section we consider briefly a mathematical model underlying methods of composing document images and methods for finding clusters of similar documents. In the third section we give a short description of software tools and experiments with a large collection of web documents. In the fourth section we discuss results of computer experiment and set further problems.

2 Computational Model

2.1 Document Image

For creating document images we used standard syntactical and lexical approaches with different parameters. Within syntactical approach we realized the shingling scheme and computing document image (sketch) with the method "*n minimal elements in a permutation*" and the method "*minimal elements in n permutations*" detailed description of which can be found in [5,6,7]. For each text the program **shingle** with two parameters (*length* and *offset*) generate contiguous subsequences of size *length* such that the distance between the beginnings of two subsequent substrings is *offset*. The set of sequences obtained in this way is hashed so that each sequence receives its own hash code. From the set of hash codes that corresponds to the document a fixed size (given by parameter) subset is chosen by means of random permutations described in [5,6,7]. The probability of the fact that minimal elements in permutations on hash code sets of shingles of documents A and B (these sets are denoted by F_A and F_B, respectively) coincide, equals to the similarity measure of these documents $\mathrm{sim}(A, B)$:

$$sim(A, B) = P[min\{\pi(F_A)\} = min\{\pi(F_B)\}] = \frac{|F_A \cap F_B|}{|F_A \cup F_B|}$$

Permutations (that can be represented by renumbering of shingles) are realized by multiplying of binary vectors that represent document images (each component of such a vector corresponds to the hash code of a particular shingle from the image) on random binary matrices. For each hash code from the set of hash codes of a documents its number in each random permutation is computed as a product of the hash code given in the form of binary vector on the randomly generated binary matrix that corresponds to the permutation. The number of permutations is also a parameter. For each permutation (given by a binary matrix) the minimal element (i.e., hash code of a shingle that became the first after the permutation) is chosen. The image of a document in the method "*n minimal elements in a permutation*" is the set of n minimal (first) hash codes in a permutation. The image of a document in the method "*minimal elements in n permutations*" is the set consisting of minimal (first) hash codes in n independent permutations. In both methods the images of all documents have fixed length n. The second approach has better randomization properties (see [5,6,7] for details), however needs more time for computations (n times more than in the first approach).

2.2 Definition of Similarity and Similarity Clusters by Means of Frequent Concepts

First, we briefly recall the main definitions of Formal Concept Analysis (FCA) [11]. Let G and M be sets, called the set of objects and the set of attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes: for $g \in G$, $m \in M$, gIm holds iff the object g has the attribute m. The triple $K = (G, M, I)$ is called a *(formal) context*. Formal contexts are naturally given by cross tables, where a cross for a pair (g, m) means that this pair belongs to the relation I. If $A \subseteq G$, $B \subseteq M$ are arbitrary subsets, then *derivation operators* are given as follows:

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\},$$
$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}.$$

The pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$ is called a *(formal) concept (of the context K)* with *extent* A and *intent* B.

The operation $(\cdot)''$ is a closure operator, i.e., it is idempotent ($X'''' = X''$), extensive ($X \subseteq X''$), and monotone ($X \subseteq Y \Rightarrow X'' \subseteq Y''$). Sets $A \subseteq G$, $B \subseteq M$ are called *closed* if $A'' = A$ and $B'' = B$. Obviously, extents and intents are closed sets. Formal concepts of context are ordered as follows: $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 \subseteq A_2 (\Leftrightarrow B_1 \supseteq B_2)$. With respect to this order the set of all formal concepts of the context K makes a lattice, called a *concept lattice* $\underline{\mathfrak{B}}(K)$ [11].

Now we recall some definitions related to association rules in Data Mining. For $B \subseteq M$ the value $|B'| = |\{g \in G \mid \forall m \in B(gIm)\}|$ is called *support* of B and denoted by $\sup(B)$. It is easily seen that the set B is closed if and only if for any $D \supset B$ one has $\sup(D) < \sup(B)$. This property is used for the definition of a closed itemset in Data Mining. A set $B \in M$ is called *k-frequent* if $|B'| \leq k$ (i.e., the set of attributes B occurs in more than k objects), where k is parameter. Computing frequent closed sets of attributes (or itemsets) became important in Data Mining since these sets give the set of all association rules [21]. For our implementation where contexts are given by set G of description units (e.g., shingles), set M of documents and incidence (occurrence) relation I on them, we define a cluster of k-similar documents as intent B of a concept (A, B) where $|A| \geq k$. Although the set of all closed sets of attributes (intents) may be exponential with respect to the number of attributes, in practice contexts are *sparse* (i.e., the average number of attributes per object is fairly small). For such cases there are efficient algorithms for constructing all most frequent closed sets of attributes (see also survey [18] on algorithms for constructing all concepts). Recently, a competitions in time efficiency for such algorithms were organized in series of workshops on Frequent Itemset Mining Implementations (FIMI). By now, a leader in time efficiency is the algorithm FPmax* [13]. We used this algorithm for finding similarities of documents and generating clusters of *very similar documents*. As mentioned before, objects are description units (shingles or words) and attributes are documents. For representation of this type *frequent closed itemsets* are closed sets of documents, for which the number of common description units in document images exceeds a given threshold. Actually, FPmax* generates *frequent itemsets* (which are not necessarily closed) and *maximal frequent itemsets*, i.e., frequent itemsets that are maximal by set inclusion. Obviously, maximal frequent sets of attributes are closed.

3 Program Implementation

Software for experiments with syntactical representation comprise the units that perform the following operations:

1. XML Parser (provided by Yandex): it parses XML packed collections of web documents
2. Removing html-markup of the documents
3. Generating shingles with given parameters length-of-shingle, offset
4. Hashing shingles
5. Composition of document image by selecting subsets (of hash codes) of shingles by means of methods *n minimal elements in a permutation* and *minimal elements in n permutations*.
6. Composition of the inverted table the list of identifiers of documents shingle thus preparing data to the format of programs for computing closed itemsets.
7. Computation of clusters of *k-similar documents* with FPmax* algorithm: the output consists of strings, where the first elements are names (ids) of documents and the last element is the number of common shingles for these documents.
8. Comparing results with the existing list of duplicates (in our experiments with the ROMIP collection of web documents, we were supplied by a precomputed list of duplicate pairs).

This unit outputs five values: 1) the number of duplicate pairs in the ROMIP collection, 2) the number of duplicate pairs for our realization, 3) the number of unique duplicate pairs in the ROMIP collection, 4) the number of unique duplicate pairs in our results, 5) the number of common pairs for the ROMIP collection and our results. For the lexical method, the description units are words (not occurring in the stop list) the frequencies of which lie in a certain interval. The amount of words in the dictionary is controlled by making closer the extreme points of the interval.

4 Experiments with ROMIP Data

As experimental data we used ROMIP collection of URLs (see www.romip.ru) consisting of 52 files of general size 4.04 GB. These files contained 530 000 web pages from narod.ru domain. Each document from the collection has size greater or equal to 10 words. For experiments the collection was partitioned into several parts consisting of three to 24 such files (from 5% to 50% percent of the whole collection). Shingling parameters used in experiments were as follows: the number of words in shingles was 10 and 20, the offset was always taken to be 1 (which means that the initial set of shingles contained all possible contiguous word sequences of a given length). Two methods of composing document image described in Section 2.1 were studied: *n minimal elements in a permutation* and *minimal elements in n permutations*. The sizes of resulting document images were taken in the interval 100 to 200 shingles. In case of lexical representation described in Section 2.1, only words from the resulting dictionary were taken in the document image (the set of descriptive words). As frequency thresholds defining *frequent closed sets* (i.e., the numbers of common shingles in document images from one cluster) we experimentally studied different values in intervals, where the maximal value is equal to the number of shingles in the document image, e.g., [85, 100] for document images with 100 shingles, the interval [135, 150] for document images of size 150, etc. Obviously, choosing the maximal value of an interval, we obtain clusters where document images coincide completely.

For parameters taking values in these intervals we studied the relation between resulting clusters of duplicates and ROMIP collection of duplicates, which consists of pairs of web documents that are considered to be near duplicates. Similarity of each pair of documents in this list is based on Edit Distance measure, two documents were taken to be duplicates by authors of this testbed if the value of the Edit Distance measure exceeds threshold 0.85 [27]. As we show below, this definition of a duplicate is prone to errors, however making a testbed by manual marking duplication in a large web document collection is hardly feasible. Unfortunately, standard lists of near-duplicates are missing even for standard corpora such as TREC or Reuters collection [22]. For validating their methods, researchers create ad-hoc lists of duplicates using slightly transformed documents from standard collections.

In our study for each such pair we sought an intent that contains both elements of the pair, and vice versa, for each cluster of *very similar documents* (i.e., for each corresponding closed set of documents with more than k common description units) we take each pair of documents in the cluster and looked for the corresponding pair in the ROMIP collection. As result we obtain the table with the number of common number of near duplicate pairs found by our method (denoted by HSE) and those in the ROMIP collection, and the number of unique pairs of HSE duplicates (document pairs occurring in a cluster of "very similar documents" and not occurring in the ROMIP collection). The results of our experiments showed that the ROMIP collection of duplicates, considered to be a benchmark, is far from being perfect. First, we detected that a large number of false duplicate pairs in this collection due to similar framing of documents. For example the pages with the following information in table 1 about historical personalities 1 and 2 were declared to be near duplicates.

However these pages, as well as many other analogous false duplicate pairs in ROMIP collection do not belong to concept-based (maximal frequent) clusters generated in our approach.

In our study we also looked for *false duplicate clusters* in the ROMIP collection, caused by transitive closure of the binary relation "X is a duplicate of Y"

Table 1. Information about historical personalities

1. Garibald II, Duke of Bavaria	2. **Giovanni, Duke of Milan**
Short information:	Short information:
Full Name: Garibald	Full Name: Giovanni Visconti
Date of birth: unknown	Date of birth: unknown
Place of birth: unknown	Place of birth: unknown
Date of death: 610	Date of death: 1354
Place of death: unknown	Place of death: unknown
Father: Tassilo I Duke of Bavaria	Father: Visconti Matteo I, the Great Lord of Milan
Mother: uknown	Mother: uknown

(as in the typical definition of a document cluster in [7]). Since the similarity relation is generally not transitive, the clusters formed by transitive closure of the relation may contain absolutely nonsimilar documents. Note that if clusters are defined via maximal frequent itemsets (subsets of attributes) there cannot be effects like this, because documents in these clusters share necessarily large itemsets (common subsets of attributes).

We analyzed about 10000 duplicate document pairs and found four *false duplicate clusters*. An example is a cluster of 58 documents containing the webpages

aadobr.narod.ru/Foto/fotofr.html
avut.narod.ru/pages/page02.htm
azer.narod.ru/index.html
b-tour.narod.ru/index.html
barents.narod.ru/foto_nov.html
...

There is no cluster like this generated in our approach. Instead, we have several clusters of similar documents that have a certain amount (depending on the parameter) of common features.

4.1 Performance of Algorithms and Their Comparison

We measured time elapsed on the stages of shingling, composing document images and generating clusters of similar documents (by algorithms for computing frequent maximal itemsets). On the last stage we used and compared various algorithms: several well-known algorithms from Data Mining [12] and AddIntent, an algorithm which proved to be one of the most efficient algorithms for constructing the set of all formal concept and concept lattices [20].

Experiments were carried out on a PC P-IV HT with 3.0 MHz frequency, 1024 MB RAM under Windows XP Professional. Experimental results and time elapsed are partially represented in Tables 2-6.

Results of the method *"n minimal elements in a permutation"***.** For the dataset narod.1-6.xml, which contained 53 539 documents, the following shingling parameters were taken: size of document image 150 shingles, length of shingle 20, offset 1 character.

Table 2. Results of the method "*n minimal elements in a permutation*"

FPmax		All Pairs of Duplicates		Unique pairs of duplicates		Common pairs
Input	Threshold	ROMIP	HSE	ROMIP	HSE	
b_1_20_s_100_n1-12.txt	100	105570	15072	97055	6557	8515
b_1_20_s_100_n1-12.txt	95	105570	20434	93982	8846	11588
b_1_20_s_100_n1-12.txt	90	105570	30858	87863	13151	17707
b_1_20_s_100_n1-12.txt	85	105570	41158	83150	18738	22420
b_1_20_s_100_n1-24.txt	100	191834	41938	175876	25980	15958
b_1_20_s_100_n1-24.txt	95	191834	55643	169024	32833	22810
b_1_20_s_100_n1-24.txt	90	191834	84012	155138	47316	36696
b_1_20_s_100_n1-24.txt	85	191834	113100	136534	57800	55300
b_1_10_s_150_n1-6.txt	150	33267	6905	28813	2451	4454
b_1_10_s_150_n1-6.txt	145	33267	9543	27153	3429	6114
b_1_10_s_150_n1-6.txt	140	33267	13827	24579	5139	8688
b_1_10_s_150_n1-6.txt	135	33267	17958	21744	6435	11523
b_1_10_s_150_n1-6.txt	130	33267	21384	19927	8044	13340
b_1_10_s_150_n1-6.txt	125	33267	24490	19236	10459	14031

Table 3.

Time elapsed, s	Precision	Recall	Threshold	F1
1,2	0,4	0,2	100	0,24
2,0	0,4	0,2	95	0,31
3,1	0,5	0,4	90	0,42
5,3	0,5	0,4	85	0,44
7,2	0,4	0,5	80	0,46

For evaluating we used a popular measure that combines Precision and Recall: the weighted harmonic mean of precision and recall, or F-measure = 2·(precision· recall)/(precision + recall). This is also known as the $F1$-measure, because recall and precision are evenly weighted. The results obtained are given in Table 3, Fig. 1 and Fig. 2.

For the dataset narod.1.xml, which contained 6941 documents, the following shingling parameters were taken: size of document image 100 shingles, length of shingle 10, offset 1 character. The results were obtained are given in Fig. 3 and Table 4.

Comparing results of FPmax with results obtained with Cluto. In our experiments with Cluto we chose the repeated-bisecting algorithm that uses the cosine similarity function with a 10-way partitioning (ClusterRB), which is mostly scalable according to its author [16]. The number of clusters is a parameter, documents are given by sets of attributes, fingerprints in our case. The algorithm outputs a set of disjoint clusters. Algorithms from FIMI repository can process very large datasets, however, to compare with Cluto (which is much more time consuming as we show below) we took collection narod.1.xml that contains 6941 documents.

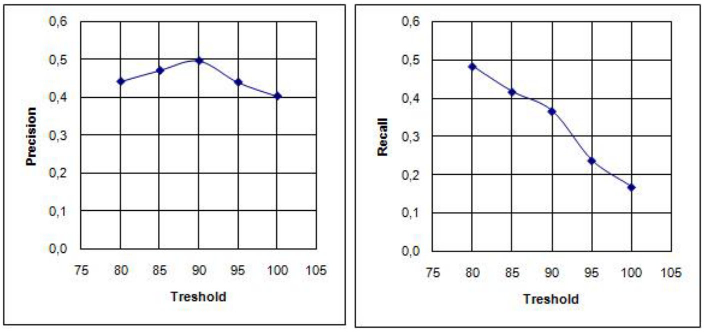


Fig. 1. Results of FPmax* on narod1-6.xml collection

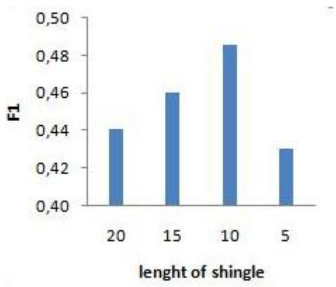


Fig. 2. F-measure vs shingle length

Table 4.

Time elapsed,s	Precision	Recall	Threshold	F1
0,098	0,76	0,25	150	0,38
0,128	0,74	0,29	145	0,42
0,187	0,70	0,39	140	0,50
0,276	0,67	0,50	135	0,57
0,383	0,63	0,57	130	0,60
0,455	0,58	0,64	125	0,61
0,559	0,47	0,64	120	0,54
0,669	0,37	0,67	115	0,48
0,873	0,29	0,70	110	0,41
1,045	0,23	0,73	105	0,35
1,294	0,18	0,69	100	0,29

Graphs and tables show that for 5000 clusters the output of ClusterRB has almost the same value of F-measure (0.63) as FPmax* for threshold 150 (F1=0,61). However, computations took 4 hours for ClusterRB and half a second for FPmax*.

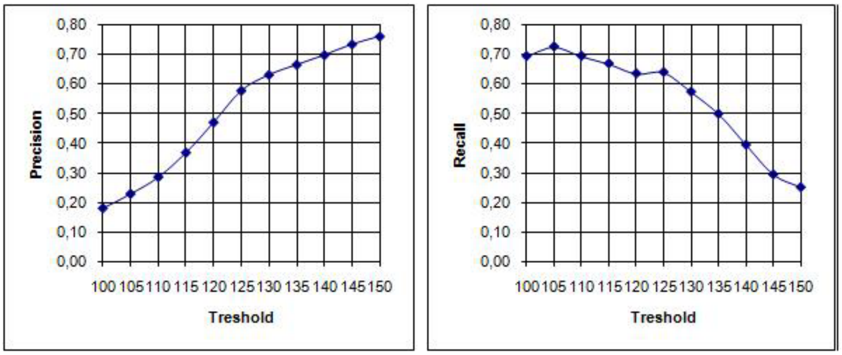


Fig. 3. Results of FPmax* on narod1.xml collection

Table 5.

Time, s	Precision	Recall	Number of clusters	F1
11	0,02	0,90	100	0,04
766	0,09	0,78	1000	0,16
3125	0,19	0,74	2000	0,30
6402	0,28	0,71	3000	0,40
14484	0,64	0,61	5000	0,63
19127	0,90	0,35	6000	0,51

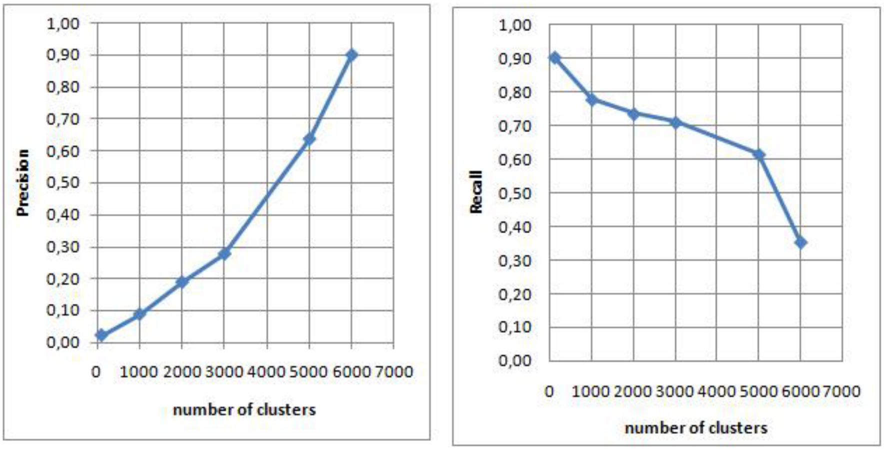


Fig. 4. Results of Cluto on narod1.xml collection

For same data collection narod.1.xml we made comparison to D-miner algorithm [3] and biclustering algorithms from BicAT system [2]. D-miner takes input in FIMI format, but computations were not completed due to lack of memory. Same effect was observed for biclustering algorithms from BicAT, which cannot

Table 6. Results for the method "*minimal elements in n permutations*"

FPmax		All Pairs of Duplicates		Unique pairs of duplicates		Common pairs
Input	Threshold	ROMIP	HSE	ROMIP	HSE	
m_1_20_s_100_n1-3.txt	100	16666	4409	14616	2359	2050
m_1_20_s_100_n1-3.txt	95	16666	5764	13887	2985	2779
m_1_20_s_100_n1-3.txt	90	16666	7601	12790	3725	3876
m_1_20_s_100_n1-3.txt	85	16666	9802	11763	4899	4903
m_1_20_s_100_n1-6.txt	100	33267	13266	28089	8088	5178
m_1_20_s_100_n1-6.txt	95	33267	15439	26802	8974	6465
m_1_20_s_100_n1-6.txt	90	33267	19393	24216	10342	9051
m_1_20_s_100_n1-12.txt	100	105570	21866	95223	11519	10347
m_1_20_s_100_n1-12.txt	95	105570	25457	93000	12887	12570

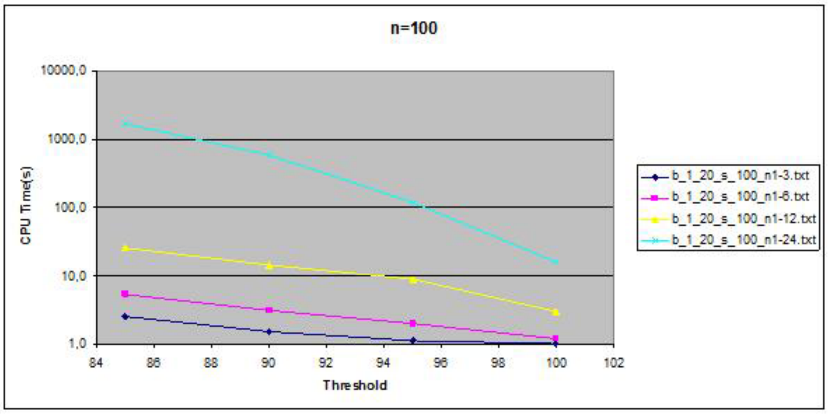


Fig. 5. Time spent by FPmax* for the method "n minimal elements in a permutation"

work with condensed representation and required inputting datatable of size 1.9 Gb in case of our document collection narod.1.xml.

FPmax* algorithm for the method *"minimal elements in n permutations"*. For large collections of documents and lower thresholds FPmax* did not complete computation due to insufficient memory.

In our experiments the best performance is attained by FPmax* algorithm, followed by the AFOPT algorithm [19]. These two algorithms proved to be the fastest in FIMI competitions [12]. AddIntent* (AddIntent modified for maximal frequent itemsets) lags behind these two, however, performs much better than MAFIA [8]. Optimized implementations of APRIORI and ECLAT [4] failed to compute the output even in the case of small subcollections of documents (about 4000 documents or 1% of the whole collection). These relative behaviour of algorithms is similar to that observed in [12] in experiments with low support. In the following Table we present running times in a typical experiment with different algorithms on a subcollection of about 10% of the whole collection.

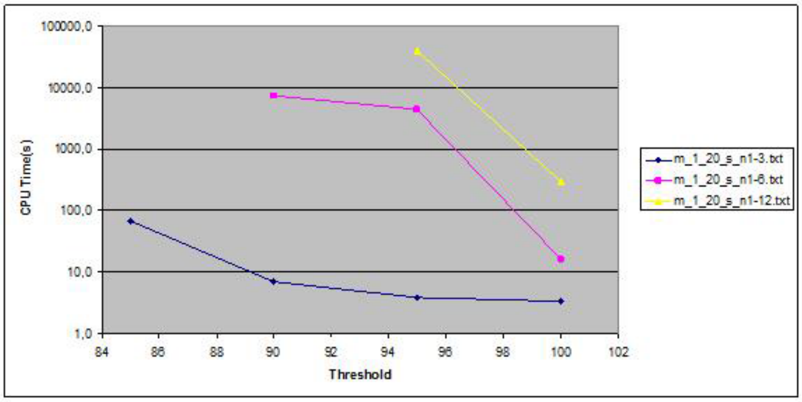


Fig. 6. Time spent by FPmax* for the method "minimal elements in n permutations"

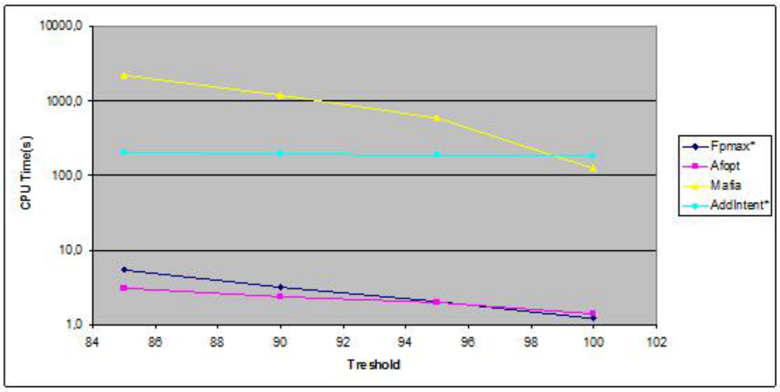


Fig. 7. Comparing efficiency of algorithms that compute maximal closed sets

In the contexts corresponding to these subcollections the number of objects is relatively large as compared to the minsup threshold value defined by parameters in the definition of duplicates. Thus, these are typical problems of generating frequent itemsets in low-support data and relative performance of data mining algorithms in our experiments is similar to that in survey [12].

5 Conclusion and Future Research

Analyzing results of our experiments with concept-based definition of clusters of similar documents with ROMIP data collection we can draw the following conclusions:

– The approach proposed in this paper allows for detecting false duplicates, as shown in experiments with ROMIP near-duplicate collection.

- Approaches based on closed sets of attributes propose adequate and efficient techniques for both determining similarity of document images and generating clusters of very similar documents. They can be efficiently used on the stage of outputting documents relevant to a query, when the number of all found relevant documents does not exceed several thousands (around 10000 documents). However, these algorithms may encounter major difficulties in treating larger collections of documents due to intrinsic exponential worst-case complexity of the problem of computing maximal frequent itemsets.
- For our datasets (which are very "column-sparse") the best Data Mining algorithms for computing frequent closed itemsets, FPmax* and Afopt, outperform AddIntent, one of the best algorithm for constructing concept lattice, adapted for computing maximal frequent itemset.
- Results of syntactical methods essentially depend on the parameter *shingle length*. Thus, in our experiments, for the shingle length 10 the results (pairs of duplicates) were much closer to those in the ROMIP collection as for the lengths of shingles equal to 20, 15, and 5.
- In our experiments the results obtained by different methods of document representation – *n minimal elements in a permutation* and *minimal elements in n permutations* – did not differ much, which testifies in favor of the first, faster method.
- Further work on generating clusters of web duplicates as formal concepts will be related to efficiency issues: more efficient algorithms and new definitions of suitable subsets of all clusters covering a collection of documents.

Acknowledgments

This research was supported by Russian Foundation for Basic Research, project 08-07-92497-NTsNIL_a. The authors would like to thank Sergei A. Obiedkov and Mikhail V. Samokhin for helpful discussions and participating in software realization of the approach.

References

1. Hasnah, A.M.: A New Filtering Algorithm for Duplicate Document Based on Concept Analysis. Journal of Computer Science 2(5), 434–440 (2006)
2. Barkow, S., Bleuler, S., Prelic, A., Zimmermann, P., Zitzler, E.: BicAT: a biclustering analysis toolbox. Bioinformatics 22(10), 1282–1283 (2006)
3. Besson, J., Robardet, C., Boulicaut, J.-F.: Constraint-based mining of formal concepts in transactional data. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS, vol. 3056, pp. 615–624. Springer, Heidelberg (2004)
4. Borgelt, C.: Efficient Implementations of Apriori and Eclat. In: Proc. Workshop on Frequent Itemset Mining Implementations Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)

5. Broder, A.: On the resemblance and containment of documents. In: Proc. Compression and Complexity of Sequences (SEQS: Sequences 1997)
6. Broder, A., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-Wise Independent Permutations. In: Proc. STOC, pp. 327–336 (1998)
7. Broder, A.: Identifying and filtering near-duplicate documents. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 1–10. Springer, Heidelberg (2000)
8. Burdick, D., et al.: MAFIA: A Performance Study of mining Maximal Frequent Itemsets. In: Proc. Workshop on Frequent Itemset Mining Implementations Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)
9. Cho, J., Shivakumar, N., Garcia-Molina, H.: Finding replicated web collections. In: Proc. SIGMOD Conference, pp. 355–366 (2000)
10. Chowdhury, A., Frieder, O., Grossman, D.A., McCabe, M.C.: Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems 20(2), 171–191 (2002)
11. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
12. Goethals, B., Zaki, M.: Advances in Frequent Itemset Mining Implementations: Introduction to FIMI 2003. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, FIMI 2003 (2003)
13. Grahne, G., Zhu, J.: Efficiently Using Prefix-trees in Mining Frequent Itemsets. In: Proc. FIMI 2003 Workshop (2003)
14. Haveliwala, T.H., Gionis, A., Klein, D., Indyk, P.: Evaluating Strategies for Similarity Search on the Web. In: Proc. WWW 2002, Honolulu, pp. 432–442 (2002)
15. Ilyinsky, S., Kuzmin, M., Melkov, A., Segalovich, I.: An efficient method to detect duplicates of Web documents with the use of inverted index. In: Proc. of the 11th International World Wide Web Conference, WWW 2002, Honolulu, Hawaii, USA, May 7-11. ACM, New York (2002)
16. Cluto, G.K.: A Clustering Toolkit. University of Minnesota, Department of Computer Science Minneapolis, MN 55455, Technical Report: 02-017, November 28 (2003)
17. Kolcz, A., Chowdhury, A., Alspector, J.: Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) Proc. KDD 2004, Seattle, pp. 605–610 (2004)
18. Kuznetsov, S.O., Obiedkov, S.A.: Comparing Performance of Algorithms for Generating Concept Lattices. Journal of Experimental and Theoretical Artificial Intelligence 14, 189–216 (2002)
19. Liu, G., Lu, H., Yu, J.X., Wei, W., Xiao, X.: AFOPT: An Efficient Implementation of Pattern Growth Approach. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003) (2003)
20. van der Merwe, D., Obiedkov, S.A., Kourie, D.: AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 372–385. Springer, Heidelberg (2004)
21. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Inform. Syst. 24(1), 25–46 (1999)
22. Potthast, M., Stein, B.: New Issues in Near-duplicate Detection, in Data Analysis. In: Machine Learning and Applications. Springer, Heidelberg (2007)

23. Pugh, W., Henzinger, M.: Detecting duplicate and near-duplicate files, United States Patent 6658423 (December 2, 2003)
24. Shivakumar, N., Garcia-Molina, H.: Finding near-replicas of documents on the web. In: Atzeni, P., Mendelzon, A.O., Mecca, G. (eds.) WebDB 1998. LNCS, vol. 1590, pp. 204–212. Springer, Heidelberg (1999)
25. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW 2008: Proceeding of the 17th international conference on World Wide Web, Beijing, China, pp. 131–140 (2008)
26. Zhao, Y., Karypis, G.: Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. Machine Learning 55, 311–331 (2004)
27. http://company.yandex.ru/academic/grant/datasets_description.xml

System Consequence

Robert E. Kent

Ontologos

Abstract. This paper discusses system consequence, a central idea in the project to lift the theory of information flow to the abstract level of universal logic and the theory of institutions. The theory of information flow is a theory of distributed logic. The theory of institutions is abstract model theory. A system is a collection of interconnected parts, where the whole may have properties that cannot be known from an analysis of the constituent parts in isolation. In an information system, the parts represent information resources and the interconnections represent constraints between the parts. System consequence, which is the extension of the consequence operator from theories to systems, models the available regularities represented by an information system as a whole. System consequence (without part-to-part constraints) is defined for a specific logical system (institution) in the theory of information flow. This paper generalizes the idea of system consequence to arbitrary logical systems.

Keywords: logical system, information flow, information system, channel, system consequence.

1 Introduction

We study the information flow of ontologies and related information resources by using the theory of institutions, which provides an axiomatization of the notion of logical system. The theory of information flow is centered on a particular logical system denoted `IF`. Institutions are based on Tarski's idea, in his semantic definition of truth, that the notion of satisfaction is central (Goguen and Burstall [6]). Ontologies are of two types: populated and unpopulated. Unpopulated ontologies contain theoretical information only, whereas populated ontologies also contain semantic information. Semantic information is related to theoretical information through satisfaction. At the most elemental level, we represent theoretical information as types (universals), semantic information as instances (tokens, particulars), and satisfaction as classification "It is particulars, things in the world, that carry information; the information they carry is in the form of types" (Barwise and Seligman [2]).

Abstraction is used in the theory of institutions: the details of the notions of formal language, sentence, semantic structure and satisfaction are abstracted away from their meaning in specific institutions such as first order logic: languages, sentences and structures are atomic (elemental) notions and satisfaction is a composite notion that relates sentences to structures. Abstraction is used in the theory of information flow: types and tokens (instances) are atomic notions and classification is a composite notion that relates types to tokens. This paper combines these uses of abstraction.

Notions of information flow are used in both theories. The theory of information flow defines the invariance of classification under parameterized atomic flow, the "adjoint connection" between token and type flow. The theory of institutions defines the invariance of satisfaction under parameterized atomic flow, the "adjoint connection" between structure and sentence flow. The theory of information flow defines direct and inverse molecular flow of `IF` theories and `IF` (local) logics, but does not exploit the "adjoint connection" between these. The theory of institutions defines direct and inverse molecular flow of specifications, and exploits the "adjoint connection" between these. This paper extends these uses of information flow.

Until this paper, the theory of institutions has not combined semantics with formalism into something like a local logic. Here, we define and discuss the notion of system consequence, as part of the effort to generalize the theory of information flow to the level of logical systems. At the atomic level the paper generalizes from type sets, classifications and sequents to languages, structures and sentences, respectively. At the molecular level it generalizes from `IF` theories and `IF` (local) logics to specifications and logics (generic, sound or composite), respectively. Logics are unsound and/or incomplete semantic represenations for various information resources such as universal algebras, libraries, knowledge bases (data collections with formal description), or physical (chemical) theories for a portion of the physical (chemical) world.

The paper falls into two parts: section 2 is concerned with the theory of institutions and section 3 is about channel theory. Readers are assumed to be familiar with the basic notions of category theory as presented in Barr and Wells [1] and information flow as presented in Barwise and Seligman [2]. Section 2 describes the two alternate representations for logical systems (institutions): the heterogeneous representation is outlined in subsection 2.1 and the homogeneous representation is discussed in some detail in subsection 2.2. Important concepts of the heterogeneous representation are languages, sentences and structures. Sentences are the atoms of formalism, and structures are the atoms of semantics. Important concepts of the homogeneous representation are specifications and logics. Specifications, the molecules of formalism, are partitioned into complete fiber preorders over their languages. Logics, the molecules of semantics, are partitioned into complete fiber preorders over their structures. Section 2 discusses the information flow between fibers of specifications along language morphisms and between fibers of logics along structure morphisms. In addition, this section also describes several well-known logical systems. Section 3 generalizes channel theory, the theory of information flow, to the theory of institutions. Subsection 3.1 defines distributed and information systems from both the formal and semantic perspectives. Subsection 3.2 defines information channels over distributed systems and describes direct and inverse information flow along channels. System consequence is defined in terms of these notions of information flow.

2 Logical Systems

Any ontology is based on the logical *language* Σ of a domain (of discourse), which often consists of the generic ideas of the connectives and quantifiers from logic and the specific ideas of the signature (the constant, function and relation symbols) for that domain. In the institutional approach, a *sentence* is the atom of formalism and a *structure* is the atom of semantics. Both sentence and structure are described and constrained by the logical language Σ. The collection of sentences and the category[1] of structures are symbolized by $\boldsymbol{sen}(\Sigma)$ and $\boldsymbol{struc}(\Sigma)$, respectively. A structure $M \in \boldsymbol{struc}(\Sigma)$ provides a universe of discourse in which to interpret a sentence $s \in \boldsymbol{sen}(\Sigma)$. In the context supplied or indexed by the language, satisfaction is the composite connecting formalism and semantics. A structure M satisfies (is a model of) a sentence s in the context of Σ, symbolized $M \models_{\Sigma} s$ (a kind of triadic construct), when s (holds in) is true when interpreted in M.

In order to define the flow of information, we make several assumptions. We assume that information resides at a (possibly abstract) location; such a location is represented by, or indexed as, a language Σ. We assume that any two locations can be connected by a link; such a location link is represented by or indexed as a language morphism $\sigma : \Sigma_1 \to \Sigma_2$, which has source language Σ_1 and target language Σ_2. This is also a primitive notion in this paper. The languages Σ_1 and Σ_2 represent two locations and the language morphism σ enables information flow from Σ_1 to Σ_2. We assume that languages form the object collection (and their morphisms form the morphism collection) of a language category **Lang** (Fig. 1). Starting from this base, we describe two equivalent representations for the notion of a *logical system* or *institution*, a heterogeneous representation and a homogeneous representation.

2.1 Heterogeneous Representation

The formal atoms (sentences) and the semantical atoms (structures) can be moved along language morphisms. For any language morphism $\sigma : \Sigma_1 \to \Sigma_2$, there is a sentence function $\boldsymbol{sen}(\sigma) : \boldsymbol{sen}(\Sigma_1) \to \boldsymbol{sen}(\Sigma_2)$ from the collection of source sentences to the collection of target sentences, and there is a structure functor[2] $\boldsymbol{struc}(\sigma) : \boldsymbol{struc}(\Sigma_2) \to \boldsymbol{struc}(\Sigma_1)$ (in the contra direction) from the category

[1] A category $\mathbf{C}$ represents some "species of mathematical structure" (Goguen [4]). It consists of a collection of objects $|\mathbf{C}|$ which have that structure and a collection of morphisms, each directed from a source object to a target object, which preserve that structure (there is an implicit notion of flow here). Morphisms compose associatively, and each object has an identity morphism on itself. As examples, **Set** is the category with sets as objects and functions as morphisms, and **Cat** is the category with categories as objects and functors as morphisms.

[2] A functor $\boldsymbol{F} : \mathbf{A} \to \mathbf{B}$ is a "natural construction on structures of one species, yielding structures of another species" (Goguen [4]). It is a link from a source category $\mathbf{A}$ of one species to a target category $\mathbf{B}$ of another species, which maps the source objects (morphisms) to target objects (morphisms), preserving directionality, composition and identity. As an example, the underlying set functor $|\text{-}| : \mathbf{Cat} \to \mathbf{Set}$ maps a category to its collection of objects and maps a functor to its underlying function on objects. The composition $\boldsymbol{F} \circ \boldsymbol{G} : \mathbf{A} \to \mathbf{C}$ of two functors $\boldsymbol{F} : \mathbf{A} \to \mathbf{B}$ and $\boldsymbol{G} : \mathbf{B} \to \mathbf{C}$ is defined in terms of their object/morphism maps.

of target structures to the category of source structures. Hence, there is a sentence functor $\boldsymbol{sen} : \mathbf{Lang} \to \mathbf{Set}$ and a structure indexed category[3] $\boldsymbol{struc} : \mathbf{Lang}^{\mathrm{op}} \to \mathbf{Cat}$. Passage composition with the underlying set functor yields the structure functor $|\boldsymbol{struc}| = \boldsymbol{struc} \circ |\text{-}| : \mathbf{Lang}^{\mathrm{op}} \to \mathbf{Cat} \to \mathbf{Set}$ (Fig. 1). The satisfaction relation is preserved during this information flow: $\boldsymbol{struc}(\sigma)(M_2) \models_{\Sigma_1} s_1$ iff $M_2 \models_{\Sigma_2} \boldsymbol{sen}(\sigma)(s_1)$. Equivalently, using structure intent (Sec. 2.2), $\boldsymbol{struc}(\sigma)(M_2)^{\Sigma_1} = \boldsymbol{sen}(\sigma)^{-1}(M_2^{\Sigma_2})$. In the institutional approach, this is regarded as the invariance of truth under change of notation. The formal and semantic functors, $\boldsymbol{sen} : \mathbf{Lang} \to \mathbf{Set}$ and $|\boldsymbol{struc}| : \mathbf{Lang}^{\mathrm{op}} \to \mathbf{Set}$, can be combined with satisfaction into a classification functor $\boldsymbol{cls} : \mathbf{Lang} \to \mathbf{Cls}$, where a language Σ maps to the satisfaction classification $\boldsymbol{cls}(\Sigma) = \langle |\boldsymbol{struc}(\Sigma)|, \boldsymbol{sen}(\Sigma), \models_\Sigma \rangle$ and invariance of truth under change of notation corresponds to the infomorphism condition.[4] The heterogeneous representation of logical systems is represented on the left side of Fig. 1.

$$\begin{tikzcd}
 & \mathbf{Set} \\
\mathbf{Lang} \arrow[ur, "\boldsymbol{sen}"] \arrow[r, "\boldsymbol{cls}"] & \mathbf{Cls} \arrow[u, "\boldsymbol{typ}"']
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbf{Lang}^{\mathrm{op}} \arrow[r, "\boldsymbol{cls}^{\mathrm{op}}"] \arrow[d, "\boldsymbol{struc}"'] \arrow[dr, "|\boldsymbol{struc}|"] & \mathbf{Cls}^{\mathrm{op}} \arrow[d, "\boldsymbol{inst}"] \\
\mathbf{Cat} \arrow[r, "|\text{-}|"'] & \mathbf{Set}
\end{tikzcd}$$

indexed categories

$$\begin{tikzcd}
{} \arrow[r, shift left, "homogenization"] & {} \arrow[l, shift left, "heterogenization"]
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbf{Snd} \arrow[r, shift left, "\boldsymbol{inc}"] \arrow[d, shift left, "\boldsymbol{pr}_0"] & \mathbf{Log} \arrow[l, shift left, "\boldsymbol{res}"] \arrow[dl, "\boldsymbol{pr}_0"'] \arrow[d, "\boldsymbol{pr}_1"] \\
\mathbf{Struc} \arrow[u, shift left, "\boldsymbol{nat}"] \arrow[r, "\boldsymbol{int}"] \arrow[dr, "\boldsymbol{lang}"'] & \mathbf{Spec} \arrow[d, "\boldsymbol{lang}"] \\
 & \mathbf{Lang}
\end{tikzcd}$$

fibered categories

Fig. 1. Logical System

2.2 Homogeneous Representation

This description is a way for "homogeneously handling situations of structural heterogeneity" (Goguen [5]). It maps the heterogeneous situations represented by indexed categories to the homogeneous situations represented by fibered

[3] An indexed category $\boldsymbol{C} : \mathbf{B}^{\mathrm{op}} \to \mathbf{Cat}$ is a (contravariant) functor from an indexing category $\mathbf{B}$ to $\mathbf{Cat}$.

[4] $\mathbf{Cls}$, the basic category of the theories of information flow (Barwise Seligman [2]) and formal concept analysis (Ganter and Wille [3]), has classifications as objects and infomorphisms as morphisms. A classification $A = \langle X, Y, \models \rangle$ consists of a set of instances X, a set of types Y and a binary incidence or classification relation $\models$ between instances and types. An infomorphism $f : A_1 = \langle X_1, Y_1, \models_1 \rangle \rightleftharpoons \langle X_2, Y_2, \models_2 \rangle = A_2$ consists of an instance function (in the contra direction) $\check{f} : X_2 \to X_1$ and a type function $\hat{f} : Y_1 \to Y_2$ that satisfy the condition $\check{f}(x_2) \models_1 y_1$ iff $x_2 \models_2 \hat{f}(y_1)$ for any source type x_1 and target instance x_2.

categories[5]. We describe fibered categories for structures, specifications and logics (generic or sound). Specifications are the formal molecules of information that flow along language morphisms. Logics are the semantic molecules of information that flow along structure morphisms. The fibered categories for specifications and logics are defined in terms of this information flow over the base categories of languages and structures, respectively. In all of the fibered categories described here, the base category is ultimately the category of languages **Lang**. The homogeneous representation of logical systems (institutions) is represented on the right side of Fig. 1.

Specifications. An unpopulated ontology expressed in terms of a language Σ is represented as a Σ-specification (Σ-presentation) $T \in \boldsymbol{spec}(\Sigma) = \wp\boldsymbol{sen}(\Sigma)$[6] consisting of a collection of Σ-sentences. In the institutional approach, a specification is a molecule of formalism, which allows for the expression of the laws and facts deemed relevant for a domain. A structure $M \in \boldsymbol{struc}(\Sigma)$ satisfies (is a model of) a specification $T \in \boldsymbol{spec}(\Sigma)$, symbolized $M \models_{\Sigma} T$, when it satisfies every sentence in the specification, $s \in T$ implies $M \models_{\Sigma} s$. A specification T entails a sentence s, symbolized $T \vdash_{\Sigma} s$, when any model of the specification satisfies the sentence. The collection $T^{\bullet} = \{s \in \boldsymbol{sen}(\Sigma) \mid T \vdash_{\Sigma} s\}$ of all sentences entailed by a specification T is called its consequence.

The consequence operator (-)$^{\bullet}$, which is defined on specifications, is a closure operator: (increasing) $T \subseteq T^{\bullet}$, (monotonic) $T_1 \subseteq T_2$ implies $T_1^{\bullet} \subseteq T_2^{\bullet}$, and (idempotent) $T^{\bullet\bullet} = T^{\bullet}$. There is an intentional (concept lattice) entailment order between specifications that is implicit in satisfaction: $T_1 \leq_{\Sigma} T_2$ when $T_1^{\bullet} \supseteq T_2^{\bullet}$; equivalently, $T_1^{\bullet} \supseteq T_2$. This is a specialization-generalization order; T_1 is more specialized than T_2, and T_2 is more generalized than T_1. We symbolize this preorder by $\boldsymbol{fbr}(\Sigma)^{\mathrm{op}} = \langle \boldsymbol{spec}(\Sigma), \leq_{\Sigma} \rangle$. Intersections and unions define joins and meets. Its opposite preorder is symbolize by $\boldsymbol{fbr}(\Sigma) = \langle \boldsymbol{spec}(\Sigma), \geq_{\Sigma} \rangle$. Any specification T is entailment equivalent to its consequence $T \cong T^{\bullet}$. A specification T is closed when it is equal to its consequence $T = T^{\bullet}$. This paper is concerned with extending the notion of consequence from specifications to information systems. Although implicit, we usually include the language in the symbolism, so that a *specification* (presentation) $\mathcal{T} = \langle \Sigma, T \rangle$ is an indexed notion consisting of a language Σ and a Σ-specification $T \in \boldsymbol{spec}(\Sigma)$.[9]

Inverse image preserves closed specifications: for any language morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$, $(-)^{\bullet} \circ \boldsymbol{sen}(\sigma)^{-1} \circ (-)^{\bullet} = (-)^{\bullet} \circ \boldsymbol{sen}(\sigma)^{-1}$; that is, $\boldsymbol{sen}(\sigma)^{-1}(T_2^{\bullet})^{\bullet} = \boldsymbol{sen}(\sigma)^{-1}(T_2^{\bullet})$ for any target specification $T_2 \in \boldsymbol{spec}(\Sigma_2)$. Direct image commutes with consequence: for any language morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$, $\wp\boldsymbol{sen}(\sigma) \circ (-)^{\bullet} = (-)^{\bullet} \circ \wp\boldsymbol{sen}(\sigma) \circ (-)^{\bullet}$; that is, $\wp\boldsymbol{sen}(\sigma)(T_1)^{\bullet} = \wp\boldsymbol{sen}(\sigma)(T_1^{\bullet})^{\bullet}$ for any

[5] A fibered category (fibration) $\boldsymbol{P} : \mathbf{E} \rightarrow \mathbf{B}$ is a category $\mathbf{E}$ whose objects exist above some underlying base category $\mathbf{B}$. Objects X in $\mathbf{B}$ index subcategories (often preorders) $\boldsymbol{fbr}(X) \subseteq \mathbf{E}$ of the fibered category called fibers. Links $f : X \rightarrow Y$ in $\mathbf{B}$ index contravariant inverse image pseudofunctors between fibers $\boldsymbol{fbr}(f) : \boldsymbol{fbr}(Y) \rightarrow \boldsymbol{fbr}(X)$ taking fiber objects indexed by Y to fiber objects indexed by X. Pseudo means preservation of composition and identity up to natural isomorphism.

[6] The symbol '$\wp$' denotes powerset for sets and direct image for functions.

source specification $T_1 \in \boldsymbol{spec}(\Sigma_1)$. The formal molecules (specifications) can be moved along language morphisms. For any language morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$, define the *direct flow* operator $\boldsymbol{dir}(\sigma) = \wp\boldsymbol{sen}(\sigma) : \boldsymbol{spec}(\Sigma_1) \rightarrow \boldsymbol{spec}(\Sigma_2)$ and the *inverse flow* operator $\boldsymbol{inv}(\sigma) = \boldsymbol{sen}(\sigma)^{-1}((\text{-})^{\bullet}) : \boldsymbol{spec}(\Sigma_2) \rightarrow \boldsymbol{spec}(\Sigma_1)$. These are adjoint monotonic functions w.r.t. specification order: $\boldsymbol{inv}(\sigma)(T_2) \leq_{\Sigma_1} T_1$ $\underline{\text{iff}}$ $T_2 \leq_{\Sigma_2} \boldsymbol{dir}(\sigma)(T_1)$.[7] We symbolized this adjunction by $\langle \boldsymbol{inv}(\sigma), \boldsymbol{dir}(\sigma)\rangle : \boldsymbol{fbr}(\Sigma_2) \rightarrow \boldsymbol{fbr}(\Sigma_1)$ between entailment preorders or by $\langle \boldsymbol{dir}(\sigma), \boldsymbol{inv}(\sigma)\rangle : \boldsymbol{fbr}(\Sigma_1)^{\text{op}} \rightarrow \boldsymbol{fbr}(\Sigma_2)^{\text{op}}$ between opposite preorders. Hence, there are indexed categories $\boldsymbol{dir} : \mathbf{Lang} \rightarrow \mathbf{Set}$ and $\boldsymbol{inv} : \mathbf{Lang}^{\text{op}} \rightarrow \mathbf{Set}$ for specifications.

A specification morphism $\sigma : \mathcal{T}_1 \rightarrow \mathcal{T}_2$ is a language morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$ that preserves entailment: $T_1 \vdash_{\Sigma_1} s_1$ implies $T_2 \vdash_{\Sigma_2} \boldsymbol{sen}(\sigma)(s_1)$ for any $s_1 \in \boldsymbol{sen}(\Sigma_1)$. Equivalently, that maps the source specification to a generalization of the target specification $\boldsymbol{dir}(\sigma)(T_1) \geq_{\Sigma_2} T_2$; or that maps the target specification to a specialization of the source specification $T_1 \geq_{\Sigma_1} \boldsymbol{inv}(\sigma)(T_2)$. Thus, the fibered category of specifications **Spec** is defined in terms of formal information flow. The fibered category of specifications **Spec** has specifications as objects and specification morphisms as morphisms (Fig. 1). There is an underlying language functor $\boldsymbol{lang} : \mathbf{Spec} \rightarrow \mathbf{Lang}$ from specifications to languages $\mathcal{T} = \langle \Sigma, T\rangle \mapsto \Sigma$.

Structures. For a language Σ, the conceptual (concept lattice) intent of a Σ-structure $M \in \boldsymbol{struc}(\Sigma)$, implicit in satisfaction, is the (closed) specification $M^{\Sigma} = \{s \in \boldsymbol{sen}(\Sigma) \mid M \models_{\Sigma} s\}$ consisting of all sentences satisfied by the structure. There is an intentional (concept lattice) order between Σ-structures: $M_1 \leq_{\Sigma} M_2$ when $M_1^{\Sigma} \leq_{\Sigma} M_2^{\Sigma}$ (specification order); equivalently, $M_1^{\Sigma} \supseteq M_2^{\Sigma}$. An indexed *structure* $\mathcal{M} = \langle \Sigma, M\rangle$ consists of a language Σ and a Σ-structure M.[9]

An indexed structure morphism $\sigma : \mathcal{M}_1 = \langle \Sigma_1, M_1\rangle \rightarrow \langle \Sigma_2, M_2\rangle = \mathcal{M}_2$ is a language morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$ that preserves satisfaction: $M_1 \models_{\Sigma_1} s_1$ implies $M_2 \models_{\Sigma_2} \boldsymbol{sen}(\sigma)(s_1)$ for any $s_1 \in \boldsymbol{sen}(\Sigma_1)$; that is, $\boldsymbol{sen}(\sigma)^{-1}(M_2^{\Sigma_2}) \leq_{\Sigma_1} M_1^{\Sigma_1}$ meaning $\sigma : \langle \Sigma_1, M_1^{\Sigma_1}\rangle \rightarrow \langle \Sigma_2, M_2^{\Sigma_2}\rangle$ is a specification morphism. Equivalently, (by satisfaction invariance) $\boldsymbol{struc}(\sigma)(M_2)^{\Sigma_1} \leq_{\Sigma_1} M_1^{\Sigma_1}$ or (by definiton of structure order) that maps the target structure to a specialization of the source structure $\boldsymbol{struc}(\sigma)(M_2) \leq_{\Sigma_1} M_1$. The fibered category of structures **Struc** has indexed structures as objects and structure morphisms as morphisms (Fig. 1). There is an underlying language functor $\boldsymbol{lang} : \mathbf{Struc} \rightarrow \mathbf{Lang}$ from structures to languages $\mathcal{M} = \langle \Sigma, M\rangle \mapsto \Sigma$. Also, there is a conceptual intent functor $\boldsymbol{int} : \mathbf{Struc} \rightarrow \mathbf{Spec}$ from structures to specifications, where $\boldsymbol{int} \circ \boldsymbol{lang} = \boldsymbol{lang}$.

Logics. A populated ontology expressed in terms of a language Σ is represented as a (generic) *logic* $\mathcal{L}$ having two components, a structure $M \in \boldsymbol{struc}(\Sigma)$ and a

[7] The paper (Tarlecki, et al [11]) claims that inverse image can be used without first computing the consequence.

specification $T \in \boldsymbol{spec}(\Sigma)$ that share Σ.[8] In the institutional approach, a logic is a molecule of semantics. Although implicit, we include the language in the symbolism, so that a *logic* $\mathcal{L} = \langle \Sigma, M, T \rangle$ is an indexed notion consisting of a language Σ, a Σ-structure M and a Σ-specification T.[9] This notion of logic is a precursor to the local logics defined and used in information flow (Barwise and Seligman [2]), which are represented by the composite logics defined below. For any fixed structure $\mathcal{M} = \langle \Sigma, M \rangle$, the set of all logics $\boldsymbol{log}(\mathcal{M})$ with that structure is a preordered set under the specification order: $\langle \Sigma, M, T_1 \rangle \leq \langle \Sigma, M, T_2 \rangle$ when $T_1 \leq T_2$. We denote this preorder by $\boldsymbol{fbr}(\mathcal{M})^{\mathrm{op}} = \langle \boldsymbol{log}(\mathcal{M}), \leq_{\mathcal{M}} \rangle \cong \langle \boldsymbol{spec}(\Sigma), \leq_{\Sigma} \rangle = \boldsymbol{fbr}(\Sigma)^{\mathrm{op}}$.

A logic morphism $\sigma : \mathcal{L}_1 \to \mathcal{L}_2$ is a language morphism $\sigma : \Sigma_1 \to \Sigma_2$ that is both a structure morphism $\sigma : \langle \Sigma_1, M_1 \rangle \to \langle \Sigma_2, M_2 \rangle$ and a specification morphism $\sigma : \langle \Sigma_1, T_1 \rangle \to \langle \Sigma_2, T_2 \rangle$. The fibered category of logics **Log** has logics as objects and logic morphisms as morphisms (Fig. 1). It is the fibered product of the fibered categories **Struc** and **Spec**. There are projective component functors from logics to structures and specifications, $\boldsymbol{pr}_0 : \mathbf{Log} \to \mathbf{Struc}$ and $\boldsymbol{pr}_1 : \mathbf{Log} \to \mathbf{Spec}$, which satisfy the condition $\boldsymbol{pr}_0 \circ \boldsymbol{lang} = \boldsymbol{pr}_1 \circ \boldsymbol{lang}$.

The semantic molecules (logics) can be moved along structure morphisms. Define direct and inverse flow of logics along structure morphisms in terms of the specification components. For any structure morphism $\sigma : \mathcal{M}_1 = \langle \Sigma_1, M_1 \rangle \to \langle \Sigma_2, M_2 \rangle = \mathcal{M}_2$, define the *direct flow* operator $\boldsymbol{dir}(\sigma) : \boldsymbol{log}(\mathcal{M}_1) \to \boldsymbol{log}(\mathcal{M}_2)$ by $\boldsymbol{dir}(\sigma)(\mathcal{L}_1) = \langle \Sigma_2, M_2, \wp\boldsymbol{sen}(\sigma)(T_1) \rangle$ for source logics $\mathcal{L}_1 = \langle \Sigma_1, M_1, T_1 \rangle$ and the *inverse flow* operator $\boldsymbol{inv}(\sigma) : \boldsymbol{log}(\mathcal{M}_2) \to \boldsymbol{log}(\mathcal{M}_1)$ by $\boldsymbol{inv}(\sigma)(\mathcal{L}_2) = \langle \Sigma_1, M_1, \boldsymbol{sen}(\sigma)^{-1}({T_2}^{\bullet}) \rangle$ for target logics $\mathcal{L}_2 = \langle \Sigma_2, M_2, T_2 \rangle$. These are adjoint monotonic functions w.r.t. logic order: $\boldsymbol{inv}(\sigma)(\mathcal{L}_2) \leq_{\Sigma_1} \mathcal{L}_1$ <u>iff</u> $\mathcal{L}_2 \leq_{\Sigma_1} \boldsymbol{dir}(\sigma)(\mathcal{L}_1)$.

In general, the logics in the institutional approach to information flow are neither sound nor complete. A logic $\mathcal{L} = \langle \Sigma, M, T \rangle$ is sound when the structure models the specification; equivalently, $T \vdash s$ implies $M \models s$; or $M^{\Sigma} \leq T^{\bullet}$. A logic $\mathcal{L}$ is complete when every sentence satisfied by the structure is a sentence entailed by the specification, $M \models s$ implies $T \vdash s$; or $T^{\bullet} \leq M^{\Sigma}$. A logic $\mathcal{L}$ is sound and complete when structure and specification are "conceptually" the

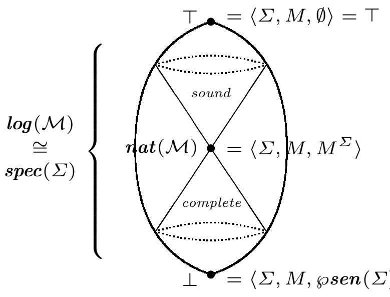
[8] Using only a single structure in logics is not a restriction. For any classification $A = \langle X, Y, \models \rangle$ there is a power instance classification $\wp A = \langle \wp X, Y, \models_{\wp} \rangle$ where $\check{X} \models_{\wp} y$ holds when $x \models y$ for all $x \in \check{X}$. For any infomorphism $f = \langle \check{f}, \hat{f} \rangle : A_1 \rightleftharpoons A_2$ there is a power instance infomorphism $\wp f = \langle \wp\check{f}, \hat{f} \rangle : \wp A_1 \rightleftharpoons \wp A_2$ with direct image instance function. Combining these constructions defines a power instance functor $\wp\boldsymbol{inst} : \mathbf{Cls} \to \mathbf{Cls}$. Hence, for any institution with classification functor $\boldsymbol{cls} : \mathbf{Lang} \to \mathbf{Cls}$, there is an associated institution with classification functor $\boldsymbol{cls} \circ \wp\boldsymbol{inst} : \mathbf{Lang} \to \mathbf{Cls}$. For this power structure institution a logic $\mathcal{L} = \langle \Sigma, \boldsymbol{M}, T \rangle$ consists of a collection of structures $\boldsymbol{M} \subseteq \boldsymbol{struc}(\Sigma)$ and a specification $T \in \boldsymbol{spec}(\Sigma)$, where the individual structures in $\boldsymbol{M}$ may or may not model the specification T. The logic $\mathcal{L}$ is sound when they all model the specification.

[9] Languages index structures, specifications and logics. The homogenization process (Fig. 1) (also called the Grothendieck construction), moving from indexed categories to fibered categories, is the process of combining an indexing language Σ with elements from the indexed category components (fibers) $\boldsymbol{struc}(\Sigma)$, $\boldsymbol{spec}(\Sigma)$, etc.

same, $M^{\Sigma} = T^{\bullet}$, generating the same concept in the satisfaction concept lattice. For any structure morphism $\sigma : \mathcal{M}_1 \rightarrow \mathcal{M}_2$, direct flow preserves soundness: if a source logic $\mathcal{L}_1$ is sound, then the direct flow logic $\boldsymbol{dir}(\sigma)(\mathcal{L}_1)$ is also sound. For any structure morphism $\sigma : \mathcal{M}_1 \rightarrow \mathcal{M}_2$, inverse flow preserves completeness: if a target logic $\mathcal{L}_2$ is complete, then the inverse flow logic $\boldsymbol{inv}(\sigma)(\mathcal{L}_2)$ is also complete.

Sound Logics. Sound logics form a subcategory of logics $\boldsymbol{inc} : \mathbf{Snd} \rightarrow \mathbf{Log}$ with the same projections (Fig. 1). Associated with any indexed structure $\mathcal{M} = \langle \Sigma, M \rangle$ is a natural logic $\boldsymbol{nat}(\mathcal{M}) = \langle \Sigma, M, M^{\Sigma} \rangle$, whose specification is the conceptual intent of $\mathcal{M}$. The natural logic is essentially (up to equivalence) the only sound and complete logic over the given language Σ. Any indexed structure morphism $\sigma : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ induces the natural logic morphism $\sigma : \boldsymbol{nat}(\mathcal{M}_1) \rightarrow \boldsymbol{nat}(\mathcal{M}_2)$. Hence, there is a natural logic functor $\boldsymbol{nat} : \mathbf{Struc} \rightarrow \mathbf{Snd}$. Structures form a reflective subcategory of sound logics, since the pair $\langle \boldsymbol{pr}_0, \boldsymbol{nat} \rangle : \mathbf{Snd} \rightarrow \mathbf{Struc}$ forms an adjunction[10] with $\mathcal{L} \geq_{\Sigma} \boldsymbol{nat}(\boldsymbol{pr}_0(\mathcal{L}))$ and $\boldsymbol{nat} \circ \boldsymbol{pr}_0 = 1_{\mathbf{Struc}}$.

Since the identity language morphism $1_{\Sigma} : \Sigma \rightarrow \Sigma$ is a structure morphism $1_{\Sigma} : \langle \Sigma, M_1 \rangle \rightarrow \langle \Sigma, M_2 \rangle$ <u>iff</u> $M_1 \geq_{\Sigma} M_2$, the structure fiber over Σ w.r.t. $\boldsymbol{lang} : \mathbf{Struc} \rightarrow \mathbf{Lang}$ is the opposite of the structure order. Since the identity language morphism $1_{\Sigma} : \Sigma \rightarrow \Sigma$ is a specification morphism $1_{\Sigma} : \langle \Sigma, T_1 \rangle \rightarrow \langle \Sigma, T_2 \rangle$ <u>iff</u> $T_1 \geq_{\Sigma} T_2$, the specification fiber over Σ w.r.t. $\boldsymbol{lang} : \mathbf{Spec} \rightarrow \mathbf{Lang}$ is the opposite of the specification order $\boldsymbol{spec}(\Sigma) = \boldsymbol{fbr}(\Sigma)^{\mathrm{op}}$. For any fixed structure $\mathcal{M} = \langle \Sigma, M \rangle$, since the identity structure morphism $1_{\mathcal{M}} : \mathcal{M} \rightarrow \mathcal{M}$ is a logic morphism $1_{\mathcal{M}} : \langle \Sigma, M, T_1 \rangle \rightarrow \langle \Sigma, M, T_2 \rangle$ <u>iff</u> $\langle \Sigma, M, T_1 \rangle \geq_{\mathcal{M}} \langle \Sigma, M, T_2 \rangle$ <u>iff</u> $T_1 \geq_{\Sigma} T_2$, the logic fiber over $\mathcal{M}$ w.r.t. $\boldsymbol{pr}_0 : \mathbf{Log} \rightarrow \mathbf{Struc}$ is the opposite of the logic order $\boldsymbol{log}(\mathcal{M}) = \boldsymbol{fbr}(\mathcal{M})^{\mathrm{op}} \cong \boldsymbol{spec}(\Sigma)$. There are larger fibers. For any fixed language Σ, the set of all logics with that language is a preordered set under the structure and specification orders: $\langle \Sigma, M_1, T_1 \rangle \leq \langle \Sigma, M_2, T_2 \rangle$ when $M_1 \leq_{\Sigma} M_2$ and $T_1 \leq_{\Sigma} T_2$. This is the (opposite of the) fiber over Σ w.r.t. the composite functor $\boldsymbol{pr}_0 \circ \boldsymbol{lang} : \mathbf{Log} \rightarrow \mathbf{Lang}$.



$\top = \langle \Sigma, M, \emptyset \rangle = \top$
sound
$\boldsymbol{log}(\mathcal{M}) \cong \boldsymbol{spec}(\Sigma)$
$\boldsymbol{nat}(\mathcal{M}) = \langle \Sigma, M, M^{\Sigma} \rangle$
complete
$\perp = \langle \Sigma, M, \wp\boldsymbol{sen}(\Sigma) \rangle$

Associated with any logic $\mathcal{L} = \langle \Sigma, M, T \rangle$ is its restriction $\boldsymbol{res}(\mathcal{L}) = \mathcal{L} \vee_{\Sigma} \boldsymbol{nat}(\mathcal{M}) = \langle \Sigma, M, M^{\Sigma} \cap T^{\bullet} \rangle$, which is the conceptual join of the logic with the natural logic of its structure component. Clearly, the restriction is a sound logic and $\boldsymbol{res}(\mathcal{L}) \geq_{\Sigma} \mathcal{L}$. There is a restriction functor $\boldsymbol{res} : \mathbf{Log} \rightarrow \mathbf{Snd}$, which maps a logic $\mathcal{L}$ to the sound logic $\boldsymbol{res}(\mathcal{L})$ and maps a logic morphism $\sigma : \mathcal{L}_1 \rightarrow \mathcal{L}_2$

[10] An adjunction (adjoint pair) consists of an adjunction of functors; that is, a pair of oppositely-directed functors that satisfy inverse equations up to morphism. Any "canonical construction from one species of structure to another" is represented by an adjunction between corresponding categories of the two species (Goguen [4]).

to the morphism of sound logics $\boldsymbol{res}(\sigma) = \sigma : \boldsymbol{res}(\mathcal{L}_1) \rightarrow \boldsymbol{res}(\mathcal{L}_2)$. This is well-defined since it just couples the structure morphism condition to the theory morphism condition. The category of sound logics forms a coreflective subcategory of the category of logics, since the pair $\langle \boldsymbol{inc}, \boldsymbol{res} \rangle : \mathbf{Log} \rightarrow \mathbf{Snd}$ forms an adjunction with $\boldsymbol{inc} \circ \boldsymbol{res} = 1_{\mathbf{Snd}}$ and $\boldsymbol{inc}(\boldsymbol{res}(\mathcal{L})) \geq_{\Sigma} \mathcal{L}$ for any logic $\mathcal{L}$. For any structure $\mathcal{M}$, restriction and inclusion on fibers are adjoint monotonic functions $\langle \boldsymbol{res}_{\mathcal{M}}, \boldsymbol{inc}_{\mathcal{M}} \rangle : \boldsymbol{log}(\mathcal{M}) \rightarrow \boldsymbol{snd}(\mathcal{M})$, where $\boldsymbol{log}(\mathcal{M}) = \boldsymbol{fbr}^{\mathrm{op}}(\mathcal{M})$ is the opposite fiber of logics over $\mathcal{M}$ and $\boldsymbol{snd}(\mathcal{M})$ is the opposite fiber of sound logics. For any structure morphism $\sigma : \mathcal{M}_1 \rightarrow \mathcal{M}_2$, the restriction-inclusion adjunction on fibers is compatible with the inverse-direct flow adjunction: $\langle \boldsymbol{res}_{\mathcal{M}_2}, \boldsymbol{inc}_{\mathcal{M}_2} \rangle \cdot \langle \boldsymbol{inv}_{\mathbf{Snd}}(\sigma), \boldsymbol{dir}_{\mathbf{Snd}}(\sigma) \rangle = \langle \boldsymbol{inv}_{\mathbf{Log}}(\sigma), \boldsymbol{dir}_{\mathbf{Log}}(\sigma) \rangle \cdot \langle \boldsymbol{res}_{\mathcal{M}_1}, \boldsymbol{inc}_{\mathcal{M}_1} \rangle$.

The movement of sound logics is a modification of logic flow. Direct flow preserves soundness, hence there is no change. Augment inverse flow by restricting to sound logics (joining with structure-intent). For any structure morphism $\sigma : \mathcal{M}_1 = \langle \Sigma_1, M_1 \rangle \rightarrow \langle \Sigma_2, M_2 \rangle = \mathcal{M}_2$, define the *direct flow* operator $\boldsymbol{dir}(\sigma) : \boldsymbol{snd}(\mathcal{M}_1) \rightarrow \boldsymbol{snd}(\mathcal{M}_2)$ by $\boldsymbol{dir}(\sigma)(\mathcal{L}_1) = \langle \Sigma_2, M_2, \wp\boldsymbol{sen}(\sigma)(T_1) \rangle$ for sound source logics $\mathcal{L}_1 = \langle \Sigma_1, M_1, T_1 \rangle$ and the *inverse flow* operator $\boldsymbol{inv}(\sigma) : \boldsymbol{snd}(\mathcal{M}_2) \rightarrow \boldsymbol{snd}(\mathcal{M}_1)$ by $\boldsymbol{inv}(\sigma)(\mathcal{L}_2) = \langle \Sigma_1, M_1, \boldsymbol{sen}(\sigma)^{-1}(T_2{}^{\bullet}) \vee M_1^{\Sigma_1} \rangle$ for sound target logics $\mathcal{L}_2 = \langle \Sigma_2, M_2, T_2 \rangle$. These are adjoint monotonic functions w.r.t. sound logic order: $\boldsymbol{inv}(\sigma)(\mathcal{L}_2) \leq_{\Sigma_1} \mathcal{L}_1$ iff $\mathcal{L}_2 \leq_{\Sigma_1} \boldsymbol{dir}(\sigma)(\mathcal{L}_1)$ for all sound target logics $\mathcal{L}_2$ and sound source logics $\mathcal{L}_1$.

A *composite logic*, the abstract representation of the (local) logics of the theory of information flow (Barwise and Seligman [2]), consists of a base logic and a sound logic sharing the same language and specification, where any sentence satisfied by the base logic structure is also satisfied by the sound logic structure. Composite logics form a category with two projective component functors to both logics and sound logics. Sound logics are justified as legitimate objects of study, since they are the common abstract form for both universal algebras and knowledge bases. In the approach used in this paper, generic logics are useful as first steps (precursors) toward the definition of sound and composite logics. Composite logics represent the commonsense theories of aritifial intelligence (AI). They are justified by the following argument for unsound or incomplete theories in Barwise and Seligman [2] "Ordinary reasoning is not logically perfect; there are logical sins of commission (unsound inferences) and of omission (inferences that are sound but not drawn). Modeling this, AI has had to cope with logics that are both unsound and incomplete."

2.3 Examples

Examples of logical systems (Goguen and Burstall [6]), (Mossakowski, et al [10]) include: first order, equational, Horn clause, intuitionistic, modal, linear, higher-order, polymorphic, temporal, process, behavioral, coalgebraic and object-oriented logics. In this paper, we describe four important logical systems: unsorted equational logic `EQ`; information flow `IF`, unsorted first-order logic with equality `FOL`, which extends `EQ` and `IF`, and the sketch institution `Sk`.

In equational logic EQ, languages are families of function symbols with arity, sentences are equations between terms of function symbols, structures are abstract algebras (universe, plus operations), and satisfaction is equational satisfaction. First-order logic FOL extends equational logic by adding relation symbols, so that EQ is a subsystem of FOL: sentences are the usual first order sentences (equations, relational expressions, connectives, quantifiers), structures extend those of unsorted equational logic by adding relations with terms, and satisfaction is as usual. In information flow IF, languages are sets of type symbols, language morphisms are maps of type symbols, sentences are sequents of type symbols, structures are classifications, and satisfaction is sequent satisfaction by instances. IF is a subsystem of FOL when types are regarded as unary relation symbols. The sketch institution Sk is the category-theoretic approach to ontological specification (Barr and Wells [1]), whose special cases include multisorted universal algebra, the entity relationship data model (Johnson and Rosebrugh [7]), and topos axiomatizations (foundations).

3 Channel Theory

System Principle: Information flow results from regularities in a distributed system. *(This is the first principle of the theory of information flow, as discussed in Barwise and Seligman [2].)*

This principle motivates the representation of distributed systems by diagrams of objects that can incorporate regularities. We will argue that these objects should be specifications in a formal representation or logics in a semantic representation.

3.1 Information Systems

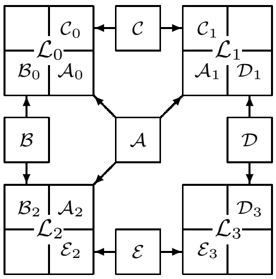
In general systems theory, a system is a collection of interconnected parts, where the whole may have properties that cannot be known from an analysis of the constituent parts in isolation. In an information system, the parts represent information resources and the interconnections represent constraints[11] between the parts.

Example. Consider a semantic information system consisting of the logics of four communities $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ that wish to interact in various ways to share

[11] In general, a constraint is conceptually an interconnection between many parts (a special case is an n-ary relation $R(A_1, A_1, \ldots, A_n)$). It can be represented with the connective form of an n-ary span. An n-ary span $(f_k : R \rightarrow A_k \mid 1 \leq k \leq n)$ in a category consists of n morphisms (directed binary constraints) f_k with a common source or vertex object R (relational concept) connecting n component objects A_k. Thus, we represent a system as a diagram consisting of a collection of objects and a collection of binary constraints. Compare (1) the use of thematic roles (case relations) in conceptual graphs and (2) the representation of entity-relationship modeling with sketches (Johnson and Rosebrugh [7]).

some of their information. We assume that $\mathcal{L}_0$ and $\mathcal{L}_1$ would like to collaborate and share information through a binary span with vertex (reference or bridging) sublogic $\mathcal{C}$. Assume the same is true for $(\mathcal{B}, \mathcal{L}_0, \mathcal{L}_2)$, $(\mathcal{D}, \mathcal{L}_1, \mathcal{L}_3)$, and $(\mathcal{E}, \mathcal{L}_2, \mathcal{L}_3)$. We also assume that $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ would like to collaborate and share information through a ternary span with vertex sublogic $\mathcal{A}$. Hence, the total information system consists of nine logics and eleven logic morphisms: the four community logics mentioned above, plus five mediating logics $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ and eleven linking logic morphisms $\mathcal{L}_0 \leftarrow \mathcal{C} \rightarrow \mathcal{L}_1, \ldots, \mathcal{A} \rightarrow \mathcal{L}_0, \ldots$ making up the spans. The underlying distributed system has the same shape, and consists of nine structures and eleven structure morphisms: four community structures $\mathcal{M}_0 = \boldsymbol{pr}_0(\mathcal{L}_0)$, $\ldots$ underlying the community logics, plus five reference structures $\boldsymbol{pr}_0(\mathcal{A}), \ldots,$ and eleven linking structure morphisms $\mathcal{M}_0 \leftarrow \boldsymbol{pr}_0(\mathcal{C}) \rightarrow \mathcal{M}_1, \ldots$ used to help aid the collaboration.

These communities might use several ways to collaborate. Suppose the three communities in the ternary span $(\mathcal{A}, \mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2)$, want to combine their axioms on a common underlying structure. Here the sublogics $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_2$ share a common underlying structure with logic $\mathcal{A}$, so that $\boldsymbol{pr}_0(\mathcal{A}_0) = \boldsymbol{pr}_0(\mathcal{A}_1) = \boldsymbol{pr}_0(\mathcal{A}_2) = \boldsymbol{pr}_0(\mathcal{A})$. As a concrete example, suppose these communities are businesses that want to collaborate about safety. Logic $\mathcal{A}$ might represent government-mandated rules about safety, whereas sublogics $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_2$ might represent special rules that the three businesses $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ need for some business transactions. Suppose the pairs of communities in the four binary spans above want to bring their vocabularies into alignment. As a concrete example, suppose the two community logics in the binary span $(\mathcal{E}, \mathcal{L}_2, \mathcal{L}_3)$ are government agencies that represent health care for citizens in sublogics $\mathcal{E}_2$ and $\mathcal{E}_3$, where $\mathcal{L}_2$ uses the term "personnel" for a citizen in the language underlying sublogic $\mathcal{E}_2$, but $\mathcal{L}_3$ uses the term "worker" in its sublogic $\mathcal{E}_3$. In order to equivalence this terminology, they use the reference logic $\mathcal{E}$ with "citizen" in its vocabulary, and then map this via morphisms $\mathcal{E}_2 \leftarrow \mathcal{E} \rightarrow \mathcal{E}_3$ as "personnel" $\leftarrowtail$ "citizen" $\rightarrowtail$ "worker".



Structure Principle: Information flow crucially involves structures of the world. *(This is the second principle of the theory of information flow, as discussed in Barwise and Seligman [2] and generalized from classifications to structures.)*

By the world we mean the category of structures, and we papaphrase the quote in the introduction to "It is structures in the world, that carry information; the information they carry is in the form of sentences". This principle motivates the use of structures as the underlying objects for the logics that incorporate the regularities in a distributed system and the use of structure morphisms as the underlying morphisms for the logic morphisms that incorporate the information flow of regularities in a distributed system.

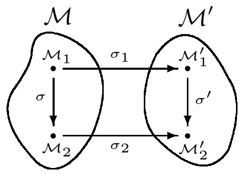
Systems. We have discussed flow links for primitive/composite notions (specification flow over language morphisms, and logic flow over structure morphisms) above, and we will discuss flow links for complex notions (distributed and information systems) below. Here we discuss constraint links. Information systems have either specifications or logics as their information resources, depending on whether they are formal or semantic in nature. A semantic information system can alternatively use sound logics or composite logics instead of generic logics. A formal information system only uses specifications. Just as every specification has an underlying language and every logic has an underlying structure, so also every information system has an underlying distributed system. As such, distributed systems have either languages or structures for their component parts, depending on whether they are under a formal or semantic information system. Without loss of generality, we discuss only semantic systems.

A *distributed system* (Fig. 2) is a diagram[12] $\mathcal{M} : \mathbf{I} \to \mathbf{Struc}$ within the ambient category of structures and structure morphisms. As such, it consists of an indexed family $\{\mathcal{M}_i = \langle \Sigma_i, M_i \rangle \mid i \in |\mathbf{I}|\}$ of structures together with an indexed family $\{\mathcal{M}_e = \sigma_e : \mathcal{M}_i \to \mathcal{M}_j \mid (e : i \to j) \in \mathbf{I}\}$ of structure morphisms. Two distributed systems with the same shape are pointwise ordered $\mathcal{M} \leq \mathcal{M}'$ when the component structures satisfy the same ordering $\mathcal{M}_i \leq \mathcal{M}'_i$ for all $i \in |\mathbf{I}|$. An *information system* (Fig. 2) is a diagram[13] $\mathcal{L} : \mathbf{I} \to \mathbf{Log}$ within the category of logics. This consists of an indexed family of logics $\{\mathcal{L}_i = \langle \Sigma_i, M_i, T_i \rangle : i \in |\mathbf{I}|\}$ and an indexed family of logic morphisms $\{\mathcal{L}_e = \sigma_e : \mathcal{L}_i \to \mathcal{L}_j \mid (e : i \to j) \in \mathbf{I}\}$. Two logical systems with the same shape are pointwise ordered $\mathcal{L} \leq \mathcal{L}'$ when the component logics satisfy the same ordering $\mathcal{L}_i \leq \mathcal{L}'_i$ for all $i \in \mathbf{I}$. This is only a preliminary ordering, since it does not represent the influence of one part of the system upon another. An information system $\mathcal{L}$ with $\mathcal{L}_i = \langle \Sigma_i, M_i, T_i \rangle$ has an underlying distributed system $\mathcal{M} = \mathcal{L} \circ \boldsymbol{pr}_0$ of the same shape with $\mathcal{M}_i = \langle \Sigma_i, M_i \rangle$. This underlying passage preserves order. Distributed and information systems were initially defined in the theory of information flow (Barwise and Seligman [2]) for the special logical system `IF`. In this paper we have defined distributed and information systems in any logical system.

[12] Let $\mathbf{V}$ be any category within which we will work. Of course, one would normally choose a category $\mathbf{V}$ that has some useful properties. We keep that category fixed throughout the discussion and call it the ambient category. We regard the objects and morphisms in the ambient category $\mathbf{V}$ to be values that we want to index, and we focus on a particular part of the ambient category. We use a functor into $\mathbf{V}$ for this purpose. A *diagram* is a functor $\boldsymbol{D} : \mathbf{I} \to \mathbf{V}$ from an indexing or shape category $\mathbf{I}$ into the ambient category $\mathbf{V}$. The objects in the indexing category are called indexing objects and the morphisms are called indexing morphisms.

[13] The representation of systems as diagrams allows for systems of systems, and systems of systems of systems, etc. Just use diagrams of diagrams. However, the product-exponential adjointness for functors then allows for the conflation of a system of systems into just a system with product indexing shape.

3.2 Information Flow



Link Types. In this paper, there are two kinds of links: constraint links and flow links. We think of constraints as being orthogonal to flow and being of a static nature. Constraints are used in the alignment of systems of information resources. For example, let $\mathcal{L}$ be an information system with underlying distributed system $\mathcal{M} = \mathcal{L} \circ \boldsymbol{pr}_0$ and let $\mathcal{M}'$ be another distributed system. A constraint link $\sigma : \mathcal{L}_1 \to \mathcal{L}_2$ in information system $\mathcal{L}$ from an information resource $\mathcal{L}_1$ located at $\mathcal{M}_1 = \boldsymbol{pr}_0(\mathcal{L}_1)$ to an information resource $\mathcal{L}_2$ located at $\mathcal{M}_2 = \boldsymbol{pr}_0(\mathcal{L}_2)$ represents the alignment of various elements in $\mathcal{L}_2$ with certain elements in $\mathcal{L}_1$. Although we think of constraints as being static in nature, there is actually a local flow, either direct or inverse, along a constraint link in order to compare the information resources at source and target to check satisfaction of alignment requirements. Flow links are used to specify and compute the fusion (Kent [8]) and consequence of systems; for example, a flow link $\sigma_1 : \mathcal{M}_1 \to \mathcal{M}'_1$ connecting a structure $\mathcal{M}_1$ in system $\mathcal{M}$ to a structure $\mathcal{M}'_1$ in system $\mathcal{M}'$, can denote the flow of information between systems $\mathcal{M}$ and $\mathcal{M}'$, either directly from M_1 to $\mathcal{M}'_1$ or inversely from M'_1 to $\mathcal{M}_1$. Flow interacts with constraints; for example, the flow links $\sigma_1 : \mathcal{M}_1 \to \mathcal{M}'_1$ and $\sigma_2 : \mathcal{M}_2 \to \mathcal{M}'_2$ connecting the $\mathcal{M}$-constraint $\sigma : \mathcal{M}_1 \to \mathcal{M}_2$ to the $\mathcal{M}'$-constraint $\sigma' : \mathcal{M}'_1 \to \mathcal{M}'_2$ should satisfy "preservation of constraints" in the sense that composition of the direct (or inverse) flow along constraint/flow paths is equal, $\boldsymbol{dir}(\sigma) \circ \boldsymbol{dir}(\sigma_2) = \boldsymbol{dir}(\sigma_1) \circ \boldsymbol{dir}(\sigma')$.

Connection Principle: It is by virtue of regularities among connections that information about some components of a distributed system carries information about other components. *(This is the third principle of the theory of information flow, as discussed in Barwise and Seligman [2].)*

This principle motivates the use of logics over structures, which lift specifications over languages, to represent information flow over covering channels of a distributed system.

Channels. For any distributed system $\mathcal{M} : \mathbf{I} \to \mathbf{Struc}$, we think of the component structures $\mathcal{M}_i$ as being parts of the system. We would like to represent the whole system as a structure, where we might use different structures for different purposes. The theory of part-whole relations is called mereology. It studies how parts are related to wholes, and how parts are related to other parts within a whole. In a distributed system, the part to part relationships are modeled by the structure morphisms $\mathcal{M}_e = \sigma_e : \mathcal{M}_i \to \mathcal{M}_j$ indexed by $e : i \to j$. We can model the whole as a structure $\mathcal{C}$ and model the part-whole relationship between some part $\mathcal{M}_i$ indexed by $i \in |\mathbf{I}|$ and the whole with a structure morphism $\gamma_i : \mathcal{M}_i \to \mathcal{C}$. An *information channel* $\langle \gamma : \mathcal{M} \Rightarrow \Delta(\mathcal{C}), \mathcal{C} \rangle$ (Fig. 2) (called a corelation by Goguen [5]) consists of an indexed family $\{\gamma_i : \mathcal{M}_i \to \mathcal{C} \mid i \in |\mathbf{I}|\}$ of structure morphisms with a common target structure $\mathcal{C}$ called the core of the channel[14]. A channel $\langle\gamma, \mathcal{C}\rangle$ *covers* a distributed system $\mathcal{M} : \mathbf{I} \to \mathbf{Struc}$ when the part-whole relationships respect the system constraints (are consistent with the part-part relationships): $\gamma_i = \sigma_e \cdot \gamma_j$ for each indexing morphism $e : i \to j$ in $\mathbf{I}$. Covering channels respect the intraconnectivity of the system. For any two covering channels $\langle\gamma', \mathcal{C}'\rangle$ and $\langle\gamma, \mathcal{C}\rangle$ over the same distributed system $\mathcal{M}$, a *refinement* $\rho : \langle\gamma', \mathcal{C}'\rangle \to \langle\gamma, \mathcal{C}\rangle$ is a constraint (structure morphism) between cores $\rho : \mathcal{C}' \to \mathcal{C}$ that respects the part-whole relationships of the two channels: $\gamma'_i \cdot \rho = \gamma_i$ for $i \in |\mathbf{I}|$. In such a situation, we say the channel $\langle\gamma', \mathcal{C}'\rangle$ is a refinement of the channel $\langle\gamma, \mathcal{C}\rangle$. A channel $\langle\iota, \coprod \mathcal{M}\rangle$ is a *minimal cover* or *optimal(ly refined) channel* of a distributed system $\mathcal{M}$ (Fig. 2) when it covers $\mathcal{M}$ and for any other covering channel $\langle\gamma, \mathcal{C}\rangle$ there is a unique refinement $[\gamma, \mathcal{C}] : \coprod \mathcal{M} \to \mathcal{C}$ from $\langle\iota, \coprod \mathcal{M}\rangle$ to $\langle\gamma, \mathcal{C}\rangle$. Any two minimal covers are isomorphic[15].

Channel Principle: The regularities of a given distributed system are relative to its analysis in terms of information channels. *(This is the fourth principle of the theory of information flow, as discussed in Barwise and Seligman [2].)*

The core of a channel connects the parts of a distributed system, reflecting the constraints when it covers the system. More refined means closer connections.

System Consequence. Without loss of generality, we discuss only the semantic version of system consequence. The fibered category **Log** is cocomplete and its projection functor $\boldsymbol{pr}_0 : \mathbf{Log} \to \mathbf{Struc}$ is cocontinuous, since the fibers $\boldsymbol{fbr}(\mathcal{M})$ are complete preorders for all indexing structures $\mathcal{M}$, direct and inverse flow are adjoint monotonic functions $\langle \boldsymbol{dir}(\sigma), \boldsymbol{inv}(\sigma)\rangle : \boldsymbol{fbr}(\mathcal{M}_1) \to \boldsymbol{fbr}(\mathcal{M}_2)$ for all indexing structure morphisms $\sigma : \mathcal{M}_1 \to \mathcal{M}_2$, and **Struc** is cocomplete (minimal covers exist for any distributed system). Then, information flow can be used to compute colmits in **Log** and to define system consequence. This holds also for sound and composite logics. It holds in the formal version and we can define system consequence for formal information systems, since comparable properties hold for the fibers $\boldsymbol{fbr}(\Sigma)$ and the category **Lang**. This holds for the logical systems `IF`, `EQ` and `FOL`, and the special cases of `Sk` mentioned above. It is based upon the colimit theorem (Tarlecki, et al [11]), a general criterion for when such colimits of specifications and logics actually exist. Let $\mathcal{L} : \mathbf{I} \to \mathbf{Log}$ be an information system with underlying distributed system $\mathcal{M} = \mathcal{L} \circ \boldsymbol{pr}_0 : \mathbf{I} \to \mathbf{Struc}$ and optimal (minimal covering) channel $\langle\iota, \coprod \mathcal{M}\rangle$. The optimal core $\coprod \mathcal{M}$ is called the sum of the distributed system $\mathcal{M}$, and the optimal channel components (structure morphisms) $\iota_i : \mathcal{M}_i \to \coprod \mathcal{M}$ for $i \in |\mathbf{I}|$ are flow links. Fusion and consequence represent the component "logics of the system and the way they fit together" [2].

[14] The notation Δ(-) denotes the constant operator, which maps objects to diagrams. For any structure $\mathcal{A}$, the distributed system $\Delta(\mathcal{A}) : \mathbf{I} \to \mathbf{Struc}$ is (constantly) the structure $\Delta(\mathcal{A})_i = \mathcal{A}$ for each index $i \in |\mathbf{I}|$ and the identity $\Delta(\mathcal{A})_e = 1_{\mathcal{A}} : \mathcal{A} \to \mathcal{A}$ for $(e : i \to j) \in \mathbf{I}$.

[15] In category theory, a covering channel $\langle\gamma : \mathcal{M} \Rightarrow \Delta(\mathcal{C}), \mathcal{C}\rangle$ is called a cocone over $\mathcal{M}$, and a minimal cover is called a colimiting cocone over $\mathcal{M}$.

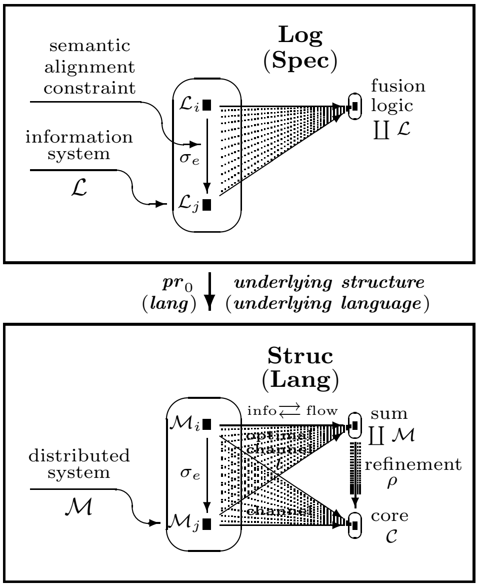


Fig. 2. Channel Theory

The *fusion* (unification) $\coprod \mathcal{L}$ of the information system $\mathcal{L}$ represents the whole system in a centralized fashion. This is called fusion in Kent [8] and theory blending in Goguen [5]. The fusion logic is defined as *direct system flow* (unification). Direct system flow has two steps: (i) direct logic flow of the component parts $\{\mathcal{L}_i \mid i \in |\mathbf{I}|\}$ of the information system along the optimal channel over the underlying distributed system to a centralized location (the lattice $\boldsymbol{fbr}(\coprod \mathcal{M})$ at the optimal channel core $\coprod \mathcal{M}$), and (ii) lattice meet combining the contributions of the parts into a whole: $\coprod \mathcal{L} \doteq \bigwedge\{\boldsymbol{dir}(\iota_i)(\mathcal{L}_i) \mid i \in |\mathbf{I}|\}$.

The *consequence* $\mathcal{L}^\blacklozenge$ of the information system $\mathcal{L}$ represents the whole system in a distributed fashion. This is an information system, defined as *inverse system flow* (projective distribution). Inverse system flow has two steps: (i) consequence of the fusion logic, and (ii) inverse logic flow of this consequence back along the same optimal channel, transfering the implications (theorems) of the whole system (the fusion logic) to the distributed locations $\mathcal{M}_i$ of the component parts: $\mathcal{L}^\blacklozenge \doteq \{\boldsymbol{inv}(\iota_i)(\coprod \mathcal{L}) \mid i \in |\mathbf{I}|\} : \mathbf{I} \to \mathbf{Log}$. The consequence operator $(\text{-})^\blacklozenge$, which is defined on information systems, is a closure operator: (increasing) $\mathcal{L} \geq \mathcal{L}^\blacklozenge$, (monotonic) $\mathcal{L}_1 \geq \mathcal{L}_2$ implies $\mathcal{L}_1^\blacklozenge \geq \mathcal{L}_2^\blacklozenge$ and (idempotent) $\mathcal{L}^{\blacklozenge\blacklozenge} = \mathcal{L}^\blacklozenge$. [16]

This is a true abstract system consequence operator, and an improvement over the "distributed logic" operator in Lecture (Chapter) 15 of Barwise and Seligman [2]

[16] By allowing system shape to vary, channels can be generalized to (co)morphisms of distributed systems. Then a notion of relative fusion (direct system flow) can be defined in terms of left Kan extension, and a notion of relative system consequence can be defined as the composition of direct followed by inverse system flow.

for three reasons: it maps an information system to another information system, recognizing the existence of constraint links between the indexed components of the system consequence; it recognizes the fact that system consequence is a closure operator, satisfying the monotonicity, increasing and idempotency laws; and it is a true generalization from the specific logical system IF to an arbitrary logical system. Fig. 2 provides a graphic representation for the calculation of system consequence: in the **Log** category information systems are illustrated as ovals, ontologies represented by logics are illustrated as nodes within ovals, and alignment constraints between ontologies are illustrated as edges between nodes; and in the **Struc** category distributed systems are illustrated as ovals, structures are illustrated as nodes within ovals, and channels are illustrated as triangular shapes.

Any information system $\mathcal{L} : \mathbf{I} \rightarrow \mathbf{Log}$ restricts as the sound information system $\boldsymbol{res}\,\mathcal{L} = \mathcal{L} \circ \boldsymbol{res} : \mathbf{I} \rightarrow \mathbf{Snd}$, where each component logic $\mathcal{L}_i$ restricts as the sound component logic $\boldsymbol{res}_{\Sigma_i}(\mathcal{L}_i)$ for each $i \in \mathbf{I}$. Any sound information system $\mathcal{L} : \mathbf{I} \rightarrow \mathbf{Snd}$ is included as the (generic) information system $\boldsymbol{inc}\,\mathcal{L} = \mathcal{L} \circ \boldsymbol{inc} : \mathbf{I} \rightarrow \mathbf{Log}$, where each sound component logic $\mathcal{L}_i$ is included as the (generic) logic $\boldsymbol{inc}_{\Sigma_i}(\mathcal{L}_i)$ for each $i \in \mathbf{I}$. Two questions arise. (1) How is the system consequence $\mathcal{L}^{\blacklozenge}$ of a sound information system $\mathcal{L}$ related to the system consequence $(\boldsymbol{inc}\,\mathcal{L})^{\blacklozenge}$ of its inclusion $\boldsymbol{inc}\,\mathcal{L}$? The direct system flow along a channel of a sound information system is sound, and hence the system consequence of a sound information system $\mathcal{L}$ is the restriction of the system consequence of its inclusion: $\mathcal{L}^{\blacklozenge} = \boldsymbol{res}\,(\boldsymbol{inc}\,\mathcal{L})^{\blacklozenge}$. (2) How is the system consequence $\mathcal{L}^{\blacklozenge}$ of an information system $\mathcal{L}$ related to the system consequence $(\boldsymbol{res}\,\mathcal{L})^{\blacklozenge}$ of its sound restriction $\boldsymbol{res}\,\mathcal{L}$? In general, since the sound restriction of the fusion logic of an information system is more specialized than the fusion logic of its sound restriction $\boldsymbol{res}(\coprod \mathcal{L}) \leq \coprod(\boldsymbol{res}\,\mathcal{L})$, the sound restriction of the system consequence of an information system $\mathcal{L}$ is more specialized than the system consequence of its sound restriction $\boldsymbol{res}(\mathcal{L}^{\blacklozenge}) \leq (\boldsymbol{res}\,\mathcal{L})^{\blacklozenge}$. An authentic example showing strict inequality, would demonstrate that restriction before fusion loses information; thus providing strong justification for the use of unsound/incomplete logics.

The pointwise entailment order $\leq$ is only a preliminary order, since it does not incorporate interactions between system component parts. Just as system consequence $(-)^{\blacklozenge}$ is analogous to specification consequence $(-)^{\bullet}$, we think of $\leq$ as analogous to $\supseteq$, reverse subset order for specifications. Extending this analogy, system entailment order $\mathcal{L}_1 \preceq \mathcal{L}_2$ for any two I-shaped information systems $\mathcal{L}_1, \mathcal{L}_2 : \mathbf{I} \rightarrow \mathbf{Log}$ is defined by $\mathcal{L}_1^{\blacklozenge} \leq \mathcal{L}_2^{\blacklozenge}$; equivalently, $\mathcal{L}_1^{\blacklozenge} \leq \mathcal{L}_2$. Pointwise order is stronger than system entailment order: $\mathcal{L}_1 \leq \mathcal{L}_2$ implies $\mathcal{L}_1 \preceq \mathcal{L}_2$. System entailment is a preorder: (reflexive) $\mathcal{L} \preceq \mathcal{L}$ and (transitive) if $\mathcal{L}_1 \preceq \mathcal{L}_2$ and $\mathcal{L}_2 \preceq \mathcal{L}_3$, then $\mathcal{L}_1 \preceq \mathcal{L}_3$. This is a specialization-generalization order; $\mathcal{L}_1$ is more specialized than $\mathcal{L}_2$, and $\mathcal{L}_2$ is more generalized than $\mathcal{L}_1$. Any information system $\mathcal{L}$ is entailment equivalent to its consequence $\mathcal{L} \cong \mathcal{L}^{\blacklozenge}$. An information system $\mathcal{L}$ is closed when it is equal to its consequence $\mathcal{L} = \mathcal{L}^{\blacklozenge}$.

A specific example of system consequence in IF, the logical system of information flow, is the file copying example in Lecture (Chapter) 15 of Barwise and Seligman [2], which involves file properties such as content, time stamp, protection, etc. Two general examples of system consequence occur in any logical system $\mathcal{L} : \mathbf{I} \rightarrow \mathbf{Log}$. The first is a system with a discrete shape $\mathbf{I} = I$, which is essentially an indexing set. Then the system consequence is the pointwise consequence $\mathcal{L}^{\blacklozenge} = \{\mathcal{L}_i^{\bullet} \mid i \in I\}$. The second is a system with any shape $\mathbf{I}$, but constant underlying distributed system $\Delta(\mathcal{M}) : \mathbf{I} \rightarrow \mathbf{Struc}$ for some single structure $\mathcal{M}$. Then the minimal cover is identity, direct system flow is the meet operation (specification union), inverse system flow is specification consequence, and system consequence is the constant information system $\Delta((\bigwedge \mathcal{L}_i)^{\bullet}) : \mathbf{I} \rightarrow \mathbf{Log}$. A (formal) concrete example of this system consequence occurs in `FOL`, the logical system of first order logic, where the information system $\mathcal{T} = (T_{\circ} \xleftarrow{1} T_{1} \xrightarrow{1} T_{\propto})$ with span shape $\mathbf{I} = \cdot \leftarrow \cdot \rightarrow \cdot$ consists of the three specifications for reflexive relations T_1, preorders (reflexive-transitive relations) $T_{\circ}$ and reflexive-symmetric relations $T_{\propto}$, with T_1 a subspecification of the other two. The underlying language for all three specifications is a single binary relation symbol. The fusion specification is $\mathcal{T}^{\blacklozenge} = T_{\equiv}$ the (closed) specification for equivalence relations (reflexive-symmetric-transitive relations), and the system consequence is the constant information system $\Delta(T_{\equiv}) : \mathbf{I} \rightarrow \mathbf{Spec}$.

4 Conclusion

This paper has discussed system consequence, one step in the program to combine and extend the theories of institutions and information flow. The institutional approach was first formulated by Goguen and Burstall [6]. Revealing its importance, many people have either independently discovered or implicitly used the institutional approach. The theory of information flow is one example of this. The Information Flow Framework [12] has followed many of the ideas of information flow, and hence has implicitly followed the institutional approach. The paper Goguen [5] is an excellent survey of the institutional approach to information integration. However, the current paper, in contrast to Goguen [5], believes that information flow follows, and is in great accord with, the institutional approach. An indication of this accord is revealed by the "Interpretations in First-Order Logic" example in Barwise and Seligman [2]. We do not believe that institutions are more abstract than information flow, but that the theory of institutions has not been fully applied in order to generalize the theory of information flow, and that the theory of information flow has not been fully applied in order to extend the theory of institutions. That is the goal of this paper.

This paper has combined two approaches to universal logic, the theories of information flow and institutions, and has applied them to information systems. We have given dual descriptions (heterogeneous and homogeneous) of logical systems and have demonstrated how important concepts in the theory of information flow can be defined within any logical system: distributed/information systems, channels, information flow, all leading up to system consequence. Hence, these concepts are independent of the particular logical system in which one works. Thus, the theory of institutions generalizes the theory of information flow, and the theory of information flow extends the theory of institutions.

A central problem of distributed logic is to understand how one part of a distributed system affects another part. This paper has solved this problem in the general case of any logical system. The solution is expressed in terms of system consequence.

References

1. Barr, M., Wells, C.: Category Theory for Computing Science. Prentice-Hall, Englewood Cliffs (1999)
2. Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge University Press, Cambridge (1997)
3. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1999)
4. Goguen, J.: A Categorical Manifesto. Math. Struc. Comp. Sci. 1, 49–67 (1991)
5. Goguen, J.: Information Integration in Institutions. Draft paper for the Jon Barwise memorial volume edited by Moss, L. (2006)
6. Goguen, J., Burstall, R.: Institutions: Abstract Model Theory for Specification and Programming. J. Assoc. Comp. Mach. 39, 95–146 (1992)
7. Johnson, M., Rosebrugh, R.: Fibrations and Universal View Updatability. Th. Comp. Sci. 388, 109–129 (2007)
8. Kent, R.E.: Semantic Integration in the Information Flow Framework. In: Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S., Uschold, M. (eds.) Semantic Interoperability and Integration, Dagstuhl Research Online Publication Serve. Dagstuhl Seminar Proceedings, vol. 04391 (2005)
9. Krotzsch, M., Hitzler, P., Zhang, G.: Morphisms in Context. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS, vol. 3596, pp. 223–237. Springer, Heidelberg (2005)
10. Mossakowski, T., Goguen, J., Diaconescu, R., Tarlecki, A.: What is a Logic? In: Beziau, J.Y. (ed.) Logica Universalis, pp. 113–133. Birkhäuser, Basel (2005)
11. Tarlecki, A., Burstall, R., Goguen, J.: Some Fundamental Algebraic Tools for the Semantics of Computation. Part 3: Indexed Categories. Th. Comp. Sci. 91, 239–264 (1991)
12. The Information Flow Framework (IFF), http://suo.ieee.org/IFF/

Fusion of Claude Bernard's Experiments for Scientific Discovery Reasoning

Claire Laudy[1,2], Bassel Habib[2], and Jean-Gabriel Ganascia[2]

[1] THALES Research & Technology, Palaiseau, France
[2] Computer Science Laboratory of Paris 6 (LIP6), University of Paris 6, Paris, France

Abstract. We are interested in using a fusion process to complete information prior to the reasoning process about scientific discoveries. In particular, using fusion to complete the set of experiments used as the source of information of the process that led Claude Bernard to his discovery about the effects of curare. Our reconstruction of the discovery process is based on his experiments as they are illustrated in his notebooks. Our main problem is the lack of some important information in his notebooks containing descriptions of his set of experiments. In order to fill in the gaps in his set of experiments, we propose to use fusion between experiments. Prior to fusion, we must ensure that the experiments are compatible according to some similarity measures and depending on the objectives of the fusion. The paper presents our domain-independent approach for similarity checking and fusion, including similarity and fusion strategies.

1 Introduction

In previous papers, we studied the process of scientific discovery [5] and [6]. Our study aimed at constructing computer programs that simulate, at a grosser or finer level of approximation, the paths that have been followed by Claude Bernard on his road to important discoveries including his discovery on the effects of curare. Many works from Cognitive Science and AI focus on modeling scientific reasoning. For instance, the work on DENDRAL and Meta-DENDRAL [1], on AM [3], on MOLGEN [19], on BACON and related programs [10] and on KEKADA [9] among others.

The focus of our research is to study discoveries that occur in experimental sciences. Since the research leading to such discoveries sometimes spans months or years, it is not practical to gather continuous protocols of the process. Thus, we must seek other sources for insights into the processes: for example, scientists' recollections, published papers on the discovery, and accounts from diaries and laboratory notes.

Our reconstruction of the process that led Claude Bernard to the discovery of the effects of curare is based on his notebooks. In other words, we reconstruct the course of his discovery using descriptions of experiments as illustrated in his laboratory notebooks. In most experimental sciences it is customary for scientists to record the details of their experimental activity on a daily basis in a laboratory notebook or log. That is why logs may contain reasons for carrying out an experiment, observations, hypotheses and conclusions drawn from the data.

We aim, in a future direction, at the reasoning about Bernard's scientific approach by constructing a causal network that links observations obtained after an experiment with deduced hypotheses. Therefore, descriptions about experiments must be complete. But the main problem, with using notebooks as the source of insight into the discovery process, is gaps in these notebooks. In the case of Bernard's notebooks, gaps are, especially, due to the lack of information about hypotheses deduced from observations. Generally, these gaps may be filled in by other sources such as : retrospective recollections of the discoverer during his lifetime or even by his published papers. But in the case of Claude Bernard and as we are interested in detailed experiments as they are illustrated in his notebooks, those other sources are not of great use.

That is the reason we propose another way to fill in these gaps by fusing descriptions about experiments. To fuse descriptions about experiments, we use a generic domain-independent approach that we presented in [13]. The aim is to take two partial experiments and build from them a more precise and more complete experiment. But before being able to fuse two experiments, we have to make sure that they are compatible according to some similarity measures and regarding some precise objectives. If so, the result of their fusion should complete one of the experiment's description with information provided by the other one. Therefore, we will be able to complete the set of Bernard's experiments prior to our reasoning.

The paper is organized as follows: In Section 2, we provide an overview of formal representation of Claude Bernard's experiments. In Section 3, we explain how we process the similarity of two experiments and in Section 4, we emphasize on the fusion aspects. Section 5 describes our results showing the fusion of two selected experiments. Finally, the conclusion summarizes our approach and describes our future directions.

2 Knowledge Representation

2.1 Epistemological Study on Claude Bernard's Manuscripts

As previously introduced, the focus of our work is on Claude Bernard's discovery about the effects of curare. This discovery is based on data gathered from his notebooks and manuscripts between 1845 and 1875. Since Claude Bernard's manuscripts contain descriptions of experiments in natural language, it was necessary to abstract from these descriptions a number of attributes (experimental criteria), which are rich enough to reflect the complexity of the original descriptions, and sufficiently representative of their variability. An attribute is created if this potential attribute intervenes in a significant proposition of available experiments.

Claude Bernard's manuscripts have been the subject of an epistemological study, which consists of several steps:

– The transcription of these manuscripts using a text editor. These manuscripts contain experiments using curare or strychnine as a toxic substance.
– The surrender of this work in a chronological order.
– The formalization of an Excel table in which Claude Bernard's experiments are annexed according to several experimental criteria (attributes) such as: weight, age, dose, animal, preparation/manipulation, point of insertion, date, ideas of experiments, observations, hypotheses and references.

2.2 Lack of Important Information in Descriptions of Experiments

The identification of the main attributes allowed us to formalize Bernard's experiments. This is a preliminary step to the simulation of these experiments in a virtual laboratory previously built [8]. Prior to the simulation, Claude Bernard's experiments are classified into several sets of experiments. The classification of experiments may be done according to one precise criterion; for instance, the set of experiments using dogs as experimental animals, or even the set of experiments including some nerve manipulations, etc. This classification is a methodological problem, because it constitutes an important step in the process of empirical discovery that concerns us, but it is not systematic, and even less, automatic.

Since Claude Bernard does not write down all the details about preparation, observations or even less about the deduced hypotheses, some experiments are not complete comparing to others in the same set of experiments. Hence comes the idea to complete descriptions about some experiments using descriptions about other experiments from the same set, which are compatible according to some similarity measures, using fusion between experiments.

Fusion allows us, on the one hand, to reduce the number of experiments within a set of experiments and thus, to reduce the number of possible simulations in a particular set of experiments since each experiment may be the object of a simulation. On the other hand, fusion allows to complete descriptions about some experiments with information of a great interest in our reasoning process. After the fusion step, information includes not only the complete set of observations resulting from an experiment takes place but also the hypotheses deduced by Claude Bernard.

2.3 Using Conceptual Graphs to Represent Experiments

As said above, our goal is to fuse Claude Bernard's experiments. Before fusion takes place, the experiments have to be represented by conceptual graphs where their concepts are the main attributes selected beforehand.

The Conceptual Graphs model was developed by JF Sowa in [18]. The model encompasses a formalism for knowledge representation and integrates linguistical, psychological and philosophical aspects. It was conceived in order to develop a logical system, able to represent natural language in a simple manner and allowing deductions and inferences.

The conceptual graphs model is essentially composed of an ontology (called support) and the graphs themselves. The ontology defines the different types

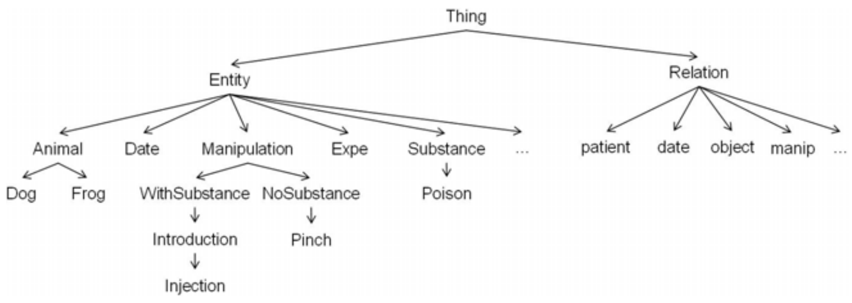


Fig. 1. Type hierarchy for Bernard's experiments

of concepts and relations which are used in the conceptual graphs. To describe Claude Bernard's experiments using conceptual graphs, we first had to define the support on which the description will be based. Therefore, we used the ontology that Claude Bernard himself defined during his work (see [8] for more details). The support defines a set of type labels as well as a partial order over the type labels and the support defines the lattice of the conceptual types. The conceptual types are, for instance, Experiment, Poison, Muscle, etc.

Figure 1 depicts a subset of the support that we formalized in order to represent Claude Bernard's experiments.

Conceptual graphs are bipartite graphs, composed of concept and relation nodes. Figure 2 shows an example of a conceptual graphs. The relation nodes are represented in ovals while the boxes represent the concept nodes. The conceptual graph of figure 2 stands for *"The patient of the second experiment is a dog named dog1"*.

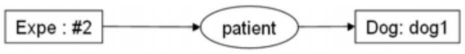


Fig. 2. Example of a conceptual graph

In the two following Sections, we detail the approach that we use, both for discriminating among the experiments the ones that could complete each others, and for fusing these similar experimentations.

Regarding the whole process of grouping similar experiments and fusing them, the discrimination phase can be viewed as a classification. The aim is to group together the experiments that can be fused in order to reduce the total number of experiments that should be simulated. It also reminds of the issue of conceptual clustering. Conceptual clustering was defined by Michalki and Stepp in [15]. Given a set of objects associated with their descriptions, conceptual clustering allows to define a set of classes that group these objects together. We are particularly interested in the work related in [2] that aims at representing and storing event descriptions as conceptual graphs. These descriptions are extracted from texts. Some subsets of event descriptions are then generalized in order to build patterns or prototypes of event.

As a second step of the conceptual clustering, the naming allows to define a description of each class. The fusion phase can be viewed as a particular type of naming where the fused experiment description describes the cluster of experiments. In [4] the authors' aim is to store descriptions of events, classify them according their similarity and generalize them in the naming phase. However, the authors also explain how all the information contained in the initial descriptions must be available in order to simulate the process of memory. Our aim with the fusion of Claude Bernard's experiments is also to keep all the information that is given by the initial experiment descriptions.

3 Discrimination between the Experiments

Before to fuse two experiments, one has to determine whether they are compatible or not. The compatibility of two experiments depends on the objectives that we have when we want to fuse them. For instance, sometimes Claude Bernard does not write down all the observations of an experiment because some of them were already observed during an earlier experiment. In such cases, we aim at fusing experiments that have almost similar preparation phases in order to complete the observations. Then the similarity of experiments will be processed regarding the preparation phase. For instance, we will emphasize on similar animals and similar poison. On the contrary, if our aim was to aggregate all the different effects of curare, we would process similarity only regarding the poison with no restriction on the animal, point of insertion, observations, etc.

To determine whether two experiments are sufficiently similar to be fused, we use a similarity measure. Several similarity measures between conceptual graphs have been proposed. Some, as [17] rely on the similarity of the structure and values of the two graphs taken as a whole. Other works concern the semantic distance between conceptual types, like [20] and [7]. Studies were also performed regarding the issues of classifying a set of conceptual graphs ([2] and [16] for instance). This is also a way to find the conceptual graphs closed one to another.

Our approach relies on combining these different approaches. Relying on [2] and [7], we define a similarity measure of two concept nodes that depends on the distance between their conceptual types, the distance between their individual values and the similarity of their immediate neighborhood. The similarity between two graphs is then computed, regarding the best matching of their nodes, as [17] does it.

One of our goals is to compare two experiments using only local comparison that are cheap in terms of processing time. Therefore, we propose to compare the different pairs of concepts of the two graphs. The global graph structure of the experiments' descriptions will be handle during the fusion process.

In the next Section, we use the following notations:

- C denotes the set of concepts with conceptual types defined on a support S;
- $c_1 \in C$ and $c_2 \in C$ are two concept nodes;
- $c_1 = [T_1 : v_1]$ and $c_2 = [T_2 : v_2]$ with $T_1 \in S$, $T_2 \in S$, the two conceptual types of c_1 and c_2.

3.1 Similarity between Two Concepts

To measure the similarity between two concepts, we propose to compare their conceptual types, their individual markers as well as their neighborhood. The study of the neighborhood gives clue about the context in which a concept is used. In our case, the neighborhood is composed of the relation nodes that are immediately linked to the concept nodes in the initial graphs.

The similarity measure is expressed as follows:

$sim : E \times E \rightarrow \ [0,1]$ where

$$sim(c_1, c_2) = p_1(T_1,T_2) sim_{Type}(T_1,T_2) * \left(\frac{p_2(T_1,T_2) sim_{Ref}(v_1,v_2) + p_3 sim_{rel}(c_1,c_2) - p_4 diss_{Rel}(c_1,c_2)}{p_2+p_3}\right)$$

– p_1, p_2, p_3 and p_4 are weights that allow to give more importance to some elements with regards to the others;
– sim_{Type}, sim_{Ref}, $sim_{RelComm}$ and $diss_{Rel}$ are local similarity/dissimilarity measures that we detail hereafter.

Similarity between Conceptual Types: $\mathbf{sim_{Type}}$. The similarity measure, between two conceptual types, depends on their distance in the lattice of concepts. Among the different studies that exist, concerning this problem, we are particularly interested in the distance between types proposed by [7]. The approach related by Gandon and colleagues proposes an extension of the distance between types proposed in [20].

Our objective is to fuse the different experiments in order to make them more precise. Therefore, unlike most of the existing measures that use the nearest common parent of the two types to be compared as key feature, we will use the nearest common subtype as key type in our measure.

The distance between two types is defined as follows:

$$\forall (t_1, t_2) \in S \times S$$
$$dist(t_1,t_2) = min_{\{t \leq t_1, t \leq t_2\}} \Big(l_S(t_1,t) + l_S(t_2,t) \Big)$$

with $\forall (t,t') \in S \times S, t \leq t'$

$$l_S(t,t') = \sum_{t_i \in \langle t,t' \rangle, t_i \neq t} \left[\frac{1}{2^{prof(t_i)}}\right]$$

where $\langle t,t' \rangle$ is the shortest path between t and t' and $prof(t)$ is the depth of t in the support.

Given this distance, the similarity between t_1 and t_2 is given by $1 - dist(t_1,t_2)$.

Similarity between Two Referents: $\mathbf{sim_{Ref}}$. The similarity between the values of two concepts depends on the conceptual types of the concepts and the application domain. For instance, the similarity between two strings can be based on the editing distance ([14]) between these two strings. If the concepts to be compared are numeric values or dates, the difference between them can be used to compute similarity.

A lot of distances exist on different types of data and are specific to different application domains. The similarity measure between two referents can be based on any of them.

Similarity Regarding Neighborhoods: $\mathbf{sim_{Rel}}$. In order to compare the context in which the two concepts are expressed, we propose to compare their immediate neighborhood. Intuitively, the similarity measure of two concepts given the common neighboring relations is processed by measuring the proportion of relations linked to the concepts and that have the same conceptual type.

The similarity of c_1 and c_2 is given by:

$$sim_{RelComm}(c_1, c_2) = \frac{2 * nbRelComm(c_1, c_2)}{nbRel(c_1) + nbRel(c_2)}$$

with $nbRelComm(c_1, c_2)$: the number of relations shared by c_1 and c_2, regarding their conceptual types;
and $nbRel(c)$: the total number of relations linked to the concept c.

Dissimilarity Regarding Neighborhoods: $\mathbf{diss_{Rel}}$. As for common neighboring relations, we compare the neighboring relations that are different. The similarity of $c_1$and c_2, given the relations that are different in their neighborhood, is given by:

$$diss_{Rel}(c_1, c_2) = \frac{nbRelDiff(c_1, c_2)}{nbRel(c_1) + nbRel(c_2)}$$

with $nbRelDiff(c_1, c_2)$: the number of relations that are not shared between c_1 and c_2;
and $nbRel(c)$: the total number of relations linked to the concept c.

The Weights. Our similarity measure encompasses several weight (p_1, p_2, p_3 et p_4 and p_{noeud}). This allows us to give more importance to some parts of the measure, according to the specificities and objectives of the information that is encoded in the graphs. A detailed example concerning the comparison of several experiments of Claude Bernard is given in [5].

3.2 Similarity between Two Experiments

The similarity of two experiments depends on the different matchings that exist between the concepts of the two graphs. It is processes given the similarity of the matching concepts.

Given a matching of the concepts of the two graphs G_1 and G_2, the similarity between G_1 and G_2 is computed as follows:

$$sim_{match}(G_1, G_2) = \frac{\sum_{(c_1,c_2) \in app} sim_{concept}(c_1, c_2)}{min(|C_1|, |C_2|)}$$

- C_1 (resp. C_2) is the set of concepts of G_1 (resp. G_2) and $|C_1|$ (resp. $|C_2|$) is the number of concepts in the graph G_1 (resp. G_2);
- $c_1 \in C_1$ (resp. $c_2 \in C_2$) is a concept of G_1 (resp. G_2).

The global similarity measure between two graphs G_1 and G_2 is then computed by maximizing the similarity of the different possible matchings:

$$sim(G_1, G_2) = \max_{\forall match \subseteq V_1, V_2} sim_{match}(G_1, G_2)$$

4 Fusion of Claude Bernard's Experiments

4.1 Maximal Join as Fusion Operator

In [13], [11] and [12], we presented a framework for high-level information fusion based on the use of the conceptual graphs formalism. In the ontology, we describe all the entities of the external world and the relations that may be observed among them. Our approach is generic. In [11], we used conceptual graphs to represent TV program descriptions. The descriptions were coming from different sources of information and were related to the same TV program. In this work, we represent Claude Bernard's experiment descriptions. The descriptions that we want to fuse relate to quite similar but different experiments. Furthermore, they are all coming from the same source of information: Claude Bernard's notebooks.

The fusion process relies on the conceptual graphs model. We use the maximal join operation defined by Sowa in order to fuse information. As shown in Figure 3, the maximal join operation allows to fuse two compatible sub graphs of two conceptual graphs. Graph G3 is the result of the fusion of G1 and G2 using the maximal join operation.

The maximal join operation copies all the information that is present in the initial graphs in the new one. Intuitively, when one wants to join two graphs maximally, the first step is to look for two compatible sub graphs in the two initial ones. The initial graphs are then joined, according to the compatible sub graphs. Furthermore, two concepts are compatible if:

- their conceptual types share a common sub-type different from $\perp$;
- their referents conform their most general subtype; and
- either one of their referent is undefined, or their referents are identical.

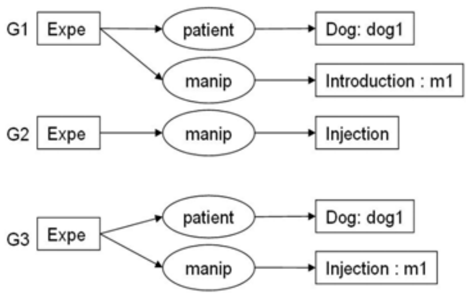


Fig. 3. Example of Maximal Join

The maximal join keeps the most specific elements of two compatible sub graphs and complete one graph according to the information contained in the other one. Furthermore, it gives several results, which depict the different ways of combining the information, that is to say, the different fusion hypotheses.

The maximal join is a fusion operator. However, as stated in [11], using the maximal join only is not sufficient in order to fuse information coming from real systems. Real data is noisy and knowledge about the domain is often needed in order to fuse two different but compatible values into a single one. Observations such as a person named "J. Smith" and a one named "Mr. John Smith" are not equals, but our background knowledge let us believe that the two observations rely to the same person.

It is necessary to extend the notion of compatibility between different concepts in the maximal join operation by introducing domain knowledge. The notion of compatibility between concepts is extended from compatibility of conceptual types only to compatibility of individual values and referents. We introduced the notion of fusion strategies. They are rules encoding domain knowledge and fusion heuristics. We use them to compute the fused value of two different observations of the same object. On the one hand, the fusion strategies extend the notion of compatibility that is used in the maximal join operation. According to some fusion strategy, two entities with two different values may be compatible and thus mergeable. On the other hand, the strategies encompass functions that give the result of the fusion of two compatible values.

4.2 Using Fusion Strategies to Handle Noisy Data

The fusion strategies are used to extend the maximal join operation that was initially defined by Sowa. Therefore, the building of the set of the fusion hypotheses of two graphs is still directed by the search of compatible projections. The notion of compatibility between two concept nodes is extended, as details hereafter. Furthermore, due to the extension of the compatibility between concepts, the construction of the joint (i.e. fused) concepts is also modified, allowing to use heuristics in order to choose the concept values. This is the concept resolution step.

The definition of the fusion strategies are divided into two parts:

- The definition of the compatibility conditions between two concepts; and
- the process of the fused value of two concepts.

The fusion strategies are expressed as the composition of two functions: Let E be the set of concept nodes defined on a support (ontology) S. Let $G1$ and $G2$ be two conceptual graphs defined on S. A fusion strategy $strategy_{fusion}$ is defined as follows:

$$strategy_{fusion} = f_{fusion} \circ f_{comp} : E \times E \rightarrow E \cup \{E \times E\}$$

where

- $f_{fusion} : \{true, false\} \times E \times E \rightarrow E \cup \{E \times E\}$ is a fusion function upon the concepts nodes of the graphs;

– $f_{comp} : E \times E \rightarrow \{true, false\} \times E \times E$ is a function testing the compatibility of two concept nodes.

The fusion strategies applied on two concept nodes result either in a fused concept node if the initial nodes are compatible, or in the initial nodes themselves if they are incompatible.

5 Case Study

5.1 Context

Within the context of understanding the process of scientific discovery followed by Claude Bernard, the goal here, as said before, is to reduce the number of experiments and, therefore, to reduce the number of simulations of his experiments, which their descriptions became more complete after fusion.

Therefore, our first step consisted of the classification of the experiments into several classes (sets of experiments). We will take here the example of the class of experiments that Claude Bernard achieved on dogs, using curare as the toxic substance. This set of experiments includes ten experiments. Once the selection of the set of experiments is done, we use our similarity measures and process the similarity between the experiments. In our example, the similarity measures include the given animal (a dog), manipulations (introduction of a toxic substance), the poison used (the curare) and the observations (whether the curare affects the animal or not). Including the observations in the similarity measure allows to ensure that we will aggregate the hypotheses that relate to experiments that have the same conclusions or observations. As a result of the similarity step, we could devide our set of experiments into two subsets, the first one contains the experiments for which the curare affects the experimental animal (six experiments including experiments 2 and 4 for which we will show the result of the fusion). The second subset is the one for which the curare has no effect (four experiments).

The last step is to apply fusion strategies on the experiment descriptions within the same subset of experiments. The fusion will allow, as said before, to aggregate all the hypotheses given by Claude Bernard, regarding a same subset of experiments. Therefore, after the fusion of the experiments, only one simulation will be sufficient to validate or invalidate several hypotheses.

5.2 From Natural Language Descriptions to Conceptual Graphs

As said before, the preliminary step of the process when we want to simulate Bernard's reasoning is to formalize his experiments. This is part of the work realized by the epistemological study and that we completed in order to transform the Excel table into a set of conceptual graphs. In the following sections, we illustrate our approach on a concrete example that uses two of Bernard's experiments on curare. The experiments are described in his notebooks as follows:

"Experiment 2: Small dog, 12 days-old. Arrow of curare lodged in the tissue of the thigh. 3 minutes later, death without screams or convulsions. Immediately after death, reflex movements are abolished. The heart beats a few more moments. At the autopsy, nothing can explain the death."

"Experiment 4: Small dog, 12 days-old. Dissolution of five centigrams of curare in water and injected by the anus in the stomach. 5 minutes later, death with the same symptoms as in the experiment 2. Immediately after death, we can not produce any reflex movement. The nerves in the legs being naked, cut or pinched do not give any contraction in the muscles. At the autopsy, nothing can explain the death."

The corresponding conceptual graphs are shown in figures 4 and 5. For a matter of readability the figures only depict a subset of the experiment descriptions.

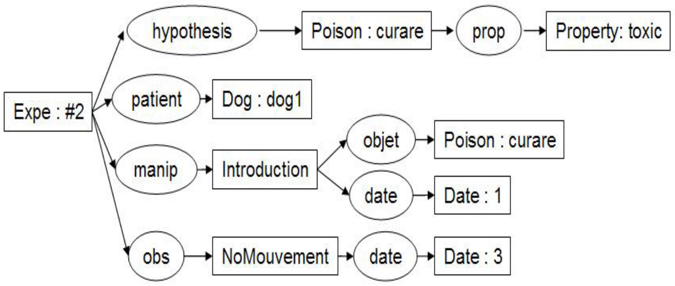


Fig. 4. First experiment

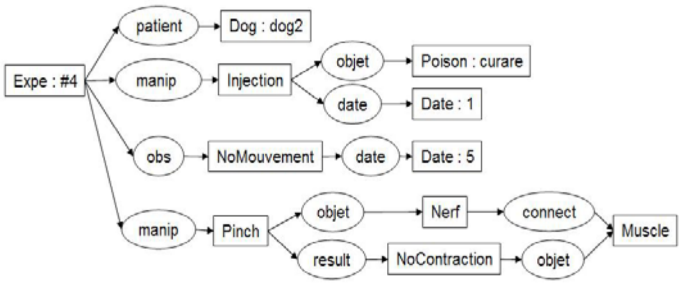


Fig. 5. Second experiment

5.3 Similarity between Experiments

The aim is to complete the observations and/or the details of the preparation of the animal for the experiments. Therefore, we will emphasize on the similarity of the preparation phase, and particularly on the type of animal, the poison used and the manipulation that is done. We propose to use the following weights:

- $p_1(T_1, T_2) = 1$ if T_1 and T_2 are sub-types of Expe, Animal, Manipulation, Date and Poison;
- $p_1(T_1, T_2) = 2$ if T_1 and T_2 are sub-types of Observation;
- $p_1(T_1, T_2) = 0$ in the other cases;
- $p_2(T_1, T_2) = 1$ if T_1 and T_2 are sub-types of Poison and Date;
- $p_2(T_1, T_2) = 0$ in the other cases;
- $p_3 = p_4 = 0$.

The similarity between 2 concepts referents is precessed as follows:

- $sim_Ref(v_1, v_2) = 1 - edit\_distance(v_1, v_2)$ where $edit_{distance}$ gives the result of the normalized version of the Leventshtein edit distance if v_1 and v_2 are strings;
- $sim_Ref(v_1, v_2) = 1 - \frac{v_1 - v_2}{max(v_1, v_2}$ if v_1 and v_2 are numerical values.

Using these weights and similarity measures, we compared two experiments that were done on dogs, using curare poison. In the first experiment, the poison was introduced in the dog's anus, while the in the second one, it was injected in a second dog's leg. The similarity rate of these experiments is 0,82.

5.4 Completing Descriptions of Experiments

As said before, figures 4 and 5 show the description of two experiments that are very similar regarding our similarity measure. We used our fusion platform to fuse them. Figure 6 shows the result of the fusion process. The experiment descriptions have been completed regarding the observations that were made during the second experiment.

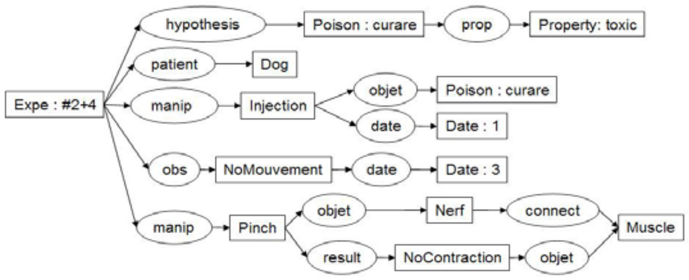


Fig. 6. Fusion of two experiment's descriptions

On the one hand, our process allows to fuse experiments regarding the observations. Thanks to that, the implicit observations that Claude Bernard didn't rewrite from one experiment to an other one are used in the simulation. On the other hand, we can also fuse the experiment descriptions regarding the hypothesis. Then, one simulation will be sufficient to validate or invalidate several hypothesis. As the simulation phase is very time consuming, it saves a considerable amount of time within the global study of scientific discovery.

6 Conclusion and Perspectives

In this paper, we showed how we used similarity and fusion strategies to fill in the gaps in Claude Bernard's notebooks. Since he doesn't always write down every detail of preparation, observations or even hypotheses deduced by these observations, the fusing of his experiments helps us to complete them, which is of great interest for the reasoning on his discovery process.

However, to simulate the reasoning about Bernard's scientific approach, especially, in his reasoning about the process of the discovery of the effects of curare, we would like to add a generalization feature among the different experiments. From several almost similar experiments, Claude Bernard was able to generalize some aspects such as the properties of curare. For instance, from two experiments where the mode of introduction of curare is different, but where the observations are similar, one would like to automatically deduce that the introduction mode doesn't affect the curare properties.

The studies performed by [2] and [16] for instance, aim at classifying conceptual graphs. After constructing classes of conceptual graphs, one of the goals is to find a single conceptual graph to represent each set of classified graphs. This graph is more general than each graph in the class it describes. We will rely on these works in order to propose a method for generalizing knowledge from a set of experiments. We aim at mixing generalization and fusion processes. In do so, we will not only deduce general knowledge from a set of different related experiments, but also take advantage of fusion to complete the generalization of several partial experiment descriptions.

References

1. Buchanan, B.G., Sutherland, E.A., Feigenbaum, E.A.: Heuristic DENDRAL: A Program for Generating Explanatory Processes in Organic Chemistry. Machine Intelligence 4 (1969)
2. de Chalendar, G., Grau, B., Ferret, O.: Conceptual Graphs Generalization. In: Proceedings of RFIA 2000. 12ème Congrès Francophone AFRIT-AFIA. Reconnaissance des Formes et Intelligence Artificielle, Paris (2000)
3. Davis, R., Lenat, D.: AM: Discovery in Mathematics as Heuristic Search in Knowledge System in Artificial Intelligence, Part one. McGraw-Hill, New York (1982)
4. Ferret, O., Grau, B.: Construire une mémoire épisodique à partir de textes: pourquoi et comment? Revue d'Intelligence Artificielle 12(3), 377–409 (1998)
5. Ganascia, J.-G., Habib, B.: An attempt to rebuild C. Bernard's scientific steps. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS, vol. 4755, pp. 248–252. Springer, Heidelberg (2007)
6. Ganascia, J.-G., Debru, C.: CYBERNARD: A Computational Reconstruction of Claude Bernard's Scientific Discoveries. Studies in Computational Intelligence (SCI) 64, 497–510 (2007)
7. Gandon, F., Corby, O., Diop, I., Lo, M.: Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique. In: Proceedings of the Semantic similarity workshop, EGC 2008, Sophia Antipolis, France (2008)

8. Habib, B., Ganascia, J.-G.: Using AI to Reconstruct Claude Bernard's Empirical Investigations. In: The 2008 International Conference on Artificial Intelligence (ICAI 2008), Las Vegas, USA, July 2008, pp. 496–501 (2008)
9. Kulkarni, D., Simon, H.A.: The Processes of Scientific Discovery: The Strategy of Experimentation. Cognitive Science 12, 139–175 (1988)
10. Langley, P., Simon, H.A., Bradshaw, G.L., Zytkow, J.M.: Scientific Discovery: Computational Explorations of the Creative Processes. The MIT Press, Cambridge (1987)
11. Laudy, C., Ganascia, J.-G.: Information Fusion in a TV program Recommendation System. In: Proceeding of the 11th International Conference on Information Fusion (FUSION 2008), Cologne, Germany, July 2008, pp. 1455–1462 (2008)
12. Laudy, C., Ganascia, J.-G.: Information Fusion using Conceptual Graphs: a TV Programs Case Study. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS, vol. 5113, pp. 158–165. Springer, Heidelberg (2008)
13. Laudy, C., Ganascia, J.-G., Sedogbo, C.: High-level Fusion based on Conceptual Graphs. In: Proceeding of the 10th International Conference on Information Fusion (FUSION 2007), Québec, Canada (July 2007)
14. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Docklady 10, 707–710 (1966)
15. Michalski, R.S., Stepp, R.E.: Learning from observation: Conceptual clustering. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning: An Artificial Intelligence Approach, Palo Alto, CA, pp. 331–363 (1983)
16. Mineau, G., Godin, R.: Automatic Structuring of Knowledge Bases by Conceptual Clustering. In: IEEE Transactions on Knowledge and Data Engineering, pp. 824–829 (1995)
17. Sorlin, S., Champin, P.-A., Solnon, C.: Mesurer la similarité de graphes étiquetés. In: Proceedings of 9èmes Journées Nationales sur la résolution pratique de problèmes NP-Complets, Amiens, France, pp. 325–339 (2003)
18. Sowa, J.F.: Conceptual Structures. In: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
19. Stefik, M.: Planning with Constraints (MOLGEN Part 1). Artificial Intelligence 16(2), 111–140 (1981)
20. Zhong, J., Zhu, H., Li, J., Yu, Y.: Conceptual Graph Matching for Semantic Search. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS, vol. 2393, pp. 92–106. Springer, Heidelberg (2002)

Distinguishing Answers in Conceptual Graph Knowledge Bases

Nicolas Moreau, Michel Leclère, and Madalina Croitoru

LIRMM, Univ. Montpellier 2, CNRS
161, rue Ada
34392 Montpellier, France
{moreau,leclere,croitoru}@lirmm.fr

Abstract. In knowledge bases (KB), the open world assumption and the ability to express variables may lead to an answer redundancy problem. This problem occurs when the returned answers are comparable. In this paper, we define a framework to distinguish amongst answers. Our method is based on adding contextual knowledge extracted from the KB. The construction of such descriptions allows clarification of the notion of redundancy between answers, based not only on the images of the requested pattern but also on the whole KB. We propose a definition for the set of answers to be computed from a query, which ensures both properties of non-redundancy and completeness. While all answers of this set can be distinguished from others with a description, an open question remains concerning what is a good description to return to an end-user. We introduce the notion of smart answer and give an algorithm that computes a set of smart answers based on a vertex neighborhood distance.

1 Motivation

In the semantic web age, a large number of applications strongly rely on the building and processing of knowledge bases (KB) for different domains (multimedia, information management, semantic portals, e-learning, etc.). The formal languages used for representation will encounter obvious scaling problems and therefore rely on implicit or explicit graph based representations (see for example Topic Maps, RDF, Conceptual Graphs, etc.) As a direct consequence, querying such systems will have to be done through graph based mechanisms and, accordingly, optimization techniques implemented [1].

In ICCS'08 [2], we identified the *semantic database context* in which a set of answers has to be computed. For this case, there are two kinds of answer graphs: answers as subgraphs of the knowledge base graph that are used for browsing the KB or for applying rules; and answers as graphs, independent of the KB, corresponding to the classical use of a querying system[1]. With this latter kind

[1] For example, in SPARQL, as blank node labels can be renamed in results, see [1] 2.4.

of answer, an important problem concerns detecting redundant answers. Unlike classical databases, redundancies are not limited to duplicate tuples. Indeed, the presence of unspecified entities (generic markers in CG or blank nodes in RDF) and also a type hierarchy leads us to consider that an answer which is more general than another as redundant. In [2], we studied this problem and proposed to return irredundant forms of the most specific answers to a query.

An important problem arising in this context is ultimately related to the nature of a KB vs. a database. With a classical database, one can assume that a query designer knows the database schema and is thus able to build a query corresponding to its needs. With a KB, the schema is extended to an ontology and the different assertions are not supposed to instantiate a specific frame (the knowledge is semi-structured). Consequently, it is difficult for a query designer to specify the content of the searched knowledge: he/she wants to obtain some information about a specific pattern of knowledge. The suppression of redundant answers only by comparing answer graphs independently of the KB results in the problem not being considered. A better way to address the redundancy problem consists of completing the answer graphs with their neighborhood to obtain more detailed answers in order to return some relevant knowledge to the end-user in order to get insight (restitution, reflet) into the diversity of the knowledge in the KB. Moreover, users seem to prefer answers in their context (e.g. paragraph) rather than the exact answer[3].

In this paper we focus on answers given in a graph based form. Our motivation stems from the homogeneity of preserving the same format between the KB, query and answer. Moreover, this will allow the reuse of answers as a KB for different future answers (see for example nested queries). Note that while this paper focuses solely on the problem of distinguishing graph based answers, the same research problem will arise and results will be obtained when of answers are represented as a tuple.

2 Contribution and Related Work

Figure 1 shows a query, a conceptual graph formalized KB (see section 3.1) and all five answers to the query in the KB (from A_1 to A_5, in gray in the picture).

The problem that arises in this scenario is how to define relevant answers when they are independent of the KB (*i.e.* a set of answer graphs and not a set of answer subgraphs of the KB). For instance, answers A_1 and A_5 are equivalent (*"there is a human who owns an animal"*), and knowledge expressed by these answers is expressed by all of the others. The open world assumption makes it impossible to state that all humans or animals represent distinct elements of the described world. Therefore it seems preferable to only return answers that bring more knowledge, i.e. in our example that *"Mary owns a cat"* and *"a human owns a dog"*. But another relevant answer could be that *"there is a human knowing Mary who owns a cat"*.

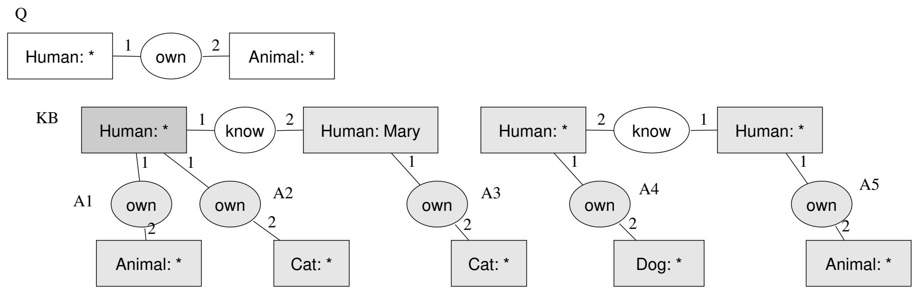


Fig. 1. Query, KB and highlighted answers

The contribution of the paper is to refine our preceding notion of redundancy between answers, to take the knowledge of the KB into account, through the notion of a *fair extension* of an answer, a graph that specifically describes an answer. Thus, an answer is irredundant if there is such a "fair" extension. A special case of this problem (when the answer is a concept node) corresponds to the problem of Generation of Referring Expressions (GRE) studied in the conceptual graph context in [4], that we extend for our purposes. Based on the extended notion of redundancy, we define two answer subset properties of *non-redundancy* (there is no redundant answer in the subset) and *completeness* (each answer is redundant to an answer of the subset). We then explain how to compute all the non-redundant and complete subsets of answers, based on a *redundancy graph*. We then discuss what is a good set of referring graphs to be returned to the user, and give an algorithm that fulfills these good properties.

As previously mentioned, a special case of answer identification was studied in [4]. The RDF query language SPARQL offers a way to describe answers (by the DESCRIBE primitive) but it does not address the specific problem of distinguishing one answer from another according to semantically sound syntactic criteria. The problem of answer redundancy in the Semantic Web context has been studied in [5], but the redundancy stated in this paper concerns the union of all answers, and corresponds, in the CG field, to the classical notion of irredundancy (see section 3.1), as opposed to our redundancy between answers. In an article about OWL-QL [6] the notion of server terseness is defined, which is the ability of a server to always produce a response collection that contains no redundant answers (i.e. there is no answer that subsumes another one). This corresponds to our previous notion of redundancy defined in [2].

In the next section, preliminary notions about conceptual graphs and our query framework are given. Section 4 deals with the extended notion of redundancy between answers and its application to a subset of answers. In section 5 we discuss the relevancy of the set of referring graphs returned to the user. We conclude our work in the last section.

3 Preliminary Notions

3.1 Simple Graphs

The conceptual graph formalism we use in this paper has been developed at LIRMM over the last 15 years [7]. The main difference with respect to the initial general model of Sowa [8] is that only representation primitives allowing graph-based reasoning are accepted.

Simple graphs (SGs) are built upon a *support*, which is a structure $S = (T_C,\ T_R,\ I,\ \sigma)$, where T_C is the set of concept types, T_R is the set of relations with any arity (arity is the number of arguments of the relation). T_C and T_R are partially ordered sets. The partial order represents a specialization relation ($t' \leq t$ is read as "t' is a specialization of t"). I is a set of individual markers. The mapping σ assigns a signature to each relation specifying its arity and the maximal type for each of its arguments.

SGs are labeled bipartite graphs denoted $G = (C_G, R_G, E_G, l_G)$ where C_G and R_G are the concept and relation node sets respectively, E_G is the set of edges and l_G is the mapping that labels nodes and edges. Concept nodes are labeled by a couple $(t : m)$, where t is a concept type and m is a marker. If the node represents an unspecified entity, its marker is the generic marker, denoted $*$, and the node is called a *generic* node, otherwise its marker is an element of I, and the node is called an *individual* node. Relation nodes are labeled by a relation r and, if n is the arity of r, it is incidental to n totally ordered edges.

A specialization/generalization relation corresponding to a deduction notion is defined over SGs and can be easily characterized by a graph homomorphism called *projection*. When there is a projection π from G to H, H is considered to be more specialized than G, denoted $H \leq G$. More specifically, a projection π from G to H is a mapping from C_G to C_H and from R_G to R_H, which preserves edges (if there is an edge numbered i between r and c in G then there is an edge numbered i between $\pi(r)$ and $\pi(c)$ in H) and may specialize labels (by observing type orders and allowing substitution of a generic marker by an individual one).

In the following, we use the notion of *bicolored SG* that was first introduced in [9]. A bicolored SG is an SG $H = \langle H_0, H_1 \rangle$ in which a color on $\{0, 1\}$ is assigned to each node of H, in such a way that the subgraph generated by 0-colored nodes, denoted H_0 and called the core of H, is a subSG. H_1, which is the sub-SG defined by 1-colored vertices and 0-colored concepts that are in relation with at least one 1-colored concept, is called the description.

3.2 Query Framework

The chosen context is a base composed of assertions of entity existences and relations over these entities, called *facts*, and stored in a single graph (not necessarily connected) named the *knowledge base*. This graph is assumed to be *normalized* if it does not contain two individual nodes with the same marker i. A normal form is easily computed by merging duplicate individual nodes of the graph. On the other hand, we do not require the KB graph to be in *irredundant form*: a graph G is in irredundant form iff there is no a projection from G in one of these strict subgraphs; otherwise this graph is said to be in *redundant form*. Indeed, computation of the irredundant form of a graph is expensive as the base can be large [10] and, moreover, there is not any local criterion for computation of the irredundant form and thus no incremental method (started at each updating of the base) can be expected. Such a KB graph B is simply queried by specifying a SG Q called the *query*. There is no constraint on the query (normalization or irredundancy). The answers to the query are found by computing the set $\Pi(Q, B)$ of projections from Q to B. The primary notion of answer consists of returning the set of subgraphs of B image of Q by a projection in $\Pi(Q, B)$.

Definition 1 (Answer set). *The set of answers of a query Q in a base B, denoted $Q(B)$, is $\{\pi(Q) \mid \pi \in \Pi(Q, B)\}$*[2]*.*

The first research question we address in this paper is whether this set of answers contains redundant answers? In fact, three kinds of redundancies can arise:

1. **Duplication:** two answers are identical. There is a duplication when two projections define the same subgraph. This problem can be solved easily by only keeping one of the duplicate subgraphs (this is done in the answer set $Q(B)$). An example of duplication is given in fig. 3(b) taken as the KB and fig. 3(c) taken as the query: there are two projections from the query whose images are the whole KB.
2. **Inclusion:** An answer is contained in another one. There is an inclusion when an answer is in a redundant form. Then its subgraph, in irredundant form, is also an answer. In the previous example (fig. 3(b) and 3(c)), there are two projections that define included answers.
3. **Redundancy:** An answer is more general or equivalent than another one. There is a redundancy when two answers are comparable (thus the knowledge expressed by one is also expressed by another); as an example answers A_1 and A_2 of figure 1.

Inclusion will be studied in section 4.4. The true redundancy problem becomes crucial since the answers are no longer KB subgraphs. Indeed, two answers can appear redundant when they are not really redundant. In [2], we define the notion of redundancy based only on the comparability of the answer graphs. This approach was motivated by the following argument: as the returned set of answers is independent of the KB, the subset of the *more specific irredundant answers*, denoted R_{min}, is sufficient to bring the entire range of answers. Moreover, R_{min} is minimal in terms of vertex number. In the following section, we characterize the true redundancy.

4 Dealing with True Redundancy

In [2], the *completeness* criterion (the knowledge expressed by each initial answer is expressed by one of the answers contained in the returned subset of answers)

[2] "Answer set", denoted $Q(B)$, and "answer" notions correspond respectively in our previous work [2] to notions of "answers by image subgraphs", denoted $R_{IP}(Q, B)$, and "images of proof". Names have been changed for simplification.

ensures that no knowledge is lost when a redundant answer is deleted. But this redundancy is based only on the comparability of answer subgraphs (that are semantically close as they are specializations of the query).

4.1 The True Redundancy

The true redundancy has to take the knowledge brought by the neighbor vertices of the answer into account. With this aim, we introduced the notion of extended answer, which is an answer supplemented with some knowledge extracted from its neighborhood.

Definition 2 (Extension of an answer). *Let B be a KB graph and Q a query graph. An extension of an answer A from $Q(B)$ is a bicolored graph $E = \langle E_0, E_1 \rangle$, where E_0 is isomorphic to A and such that there is a projection π from E to B with $\pi(E_0) = A$.*

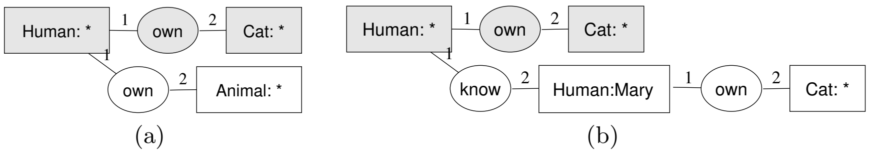


Fig. 2. Several extensions of answer A_2 of fig. 1

Fig. 2 represents several extensions of answer A_2 of fig. 1. Note that the extensions are not necessarily isomorphic copies of subgraphs of the KB.

With the previous definition of redundancy (section 3.2), if two answers have two incomparable extensions, they are two distinct answers and thus have to be considered as non-redundant answers, the one w.r.t. the other one. However, these two answers must not be distinguished in an "artificial" way: i.e. if the answers are distinguished by selectively adding knowledge to the extension of only certain answers but not to all of those which possess this knowledge in their neighborhood.

Definition 3 (Fairness property). *An extension E of an answer A from a set $Q(B)$ is fair iff there is no extension E' of another answer A' from $Q(B)$ such that there is a projection π from E to E' with $\pi(E_0) = E'_0$ and $\pi(E_1) = E'_1$.*

The extension of figure 2(b) is a fair extension of the answer A_2 of figure 1, contrary to the extension of figure 2(a). One can now give a definition of the true redundancy:

Definition 4 (True redundancy). *An answer A from a set of answers $Q(B)$ is truly redundant if there is no fair extension of this answer.*

In the example of fig. 1, answer A_1 is redundant. The search for a fair extension may seem to be a difficult task, considering that the number of extensions that can be generated for an answer is infinite. However, these extensions are semantically bounded by a more specific one that corresponds to an isomorphic copy of the base, $\langle A, B \setminus A\rangle$[3]. Naturally, the more one adds knowledge to the extension (the more it is specific), the more one potentially distinguishes this answer. So A is irredundant only if $\langle A, B \setminus A\rangle$ is a fair extension.

Theorem 1. *There is a fair extension E of an answer A iff $\langle A, B \setminus A\rangle$ is a fair extension.*

Proof. By contraposition. Suppose that there is a fair extension $E = \langle E_0, E_1\rangle$ of A and that $\langle A, B \setminus A\rangle$ is not a fair extension of A. $\langle A, B \setminus A\rangle$ is an extension of A and is isomorphic (without considering the colors) to B. As E is an extension of A, there is a projection π from E to B such that $\pi(E_0) = A$. As $\langle A, B \setminus A\rangle$ is not a fair extension of A, there is a projection π' from $\langle A, B \setminus A\rangle$ to B such that $\pi(R_0) \neq A$. So there is a projection $\pi'' = \pi \circ \pi'$ from E to B such that $\pi''(E_0) \neq A$. Thus E is not a fair extension of A. □

Corollary 1. *An answer A of $Q(B)$ is truly irredundant iff $\langle A, B \setminus A\rangle$ is a fair extension.*

4.2 The GRE Problem

The problem of finding a fair extension of an answer is strongly related to the problem of generation of referring expressions (GRE) known in the natural language processing field, which aims to describe an object in a scene such that the description only refers to this object. The GRE problem was formalized in the CG framework in [4], where the scene is an SG and the object to identify is a concept of the SG. A *referring graph* of a concept v is a subgraph of the KB containing this concept, and a *distinguishing graph* is a referring graph whose v is a fixed point for all projections of the graph in the KB. Our problem of answer redundancy can be seen as an extension of this formalization.

Definition 5 (Referring graph). *A referring graph of a subgraph G' of a graph G is a subgraph R of G that contains G' as a subgraph. To distinguish G' from the rest of the referring graph, we denote it as a bicolored graph $R = \langle R_0, R_1\rangle$ where $R_0 = G'$ and $R_1 = R \setminus G'$.*

Definition 6 (Distinguishing graph). *A referring graph R of a subgraph G' of a graph G is distinguishing if, for each projection π from R to G, $\pi(R_0) = G'$.*

The referring graph of figure 2(b) is a distinguishing graph of the answer A_2 of figure 1, contrary of the referring graph of figure 2(a). We can now link the true redundancy notion with the existence of a distinguishing graph for a given answer. The next properties strengthen the notion of true redundancy by showing its independence in the query and thus in the other answers.

Property 1. Let A be an answer of $Q(B)$. There is a fair extension of A iff there is a distinguishing graph of A w.r.t. B.

[3] For clarity, we sometimes denote subgraphs of KB as cores or descriptions of bicolored graph instead of their isomorphic copies.

Proof. Let $E = \langle E_0, E_1 \rangle$ be a fair extension of A. Then $\pi(E)$ is a distinguishing graph of A w.r.t. B. On the other hand, a distinguishing graph of A is also a fair extension of A. □

Corollary 2. *An answer A of $Q(B)$ is irredundant iff $R = \langle A, B \setminus A \rangle$ is a distinguishing graph of A w.r.t. to B.*

Thus, determining whether an answer is irredundant can be done by testing the distinguishness of the referring graph built from KB. However, the cost of such a test is exponential in the size of the KB.

4.3 Redundancy and Subset of Answers

Since the redundancy of an answer has been defined, it could be considered that for building a set of answers without redundancies one could simply remove redundant answers. However, the redundancy defined in the preceding section hides the fact that they are two types of redundancy. This is shown in fig. 3 (a query and three different KBs) :

The first case will arise when an answer is completely redundant with respect to another answer (fig. 3(b)). This means that A_2 is redundant w.r.t. A_1 but the reverse does not hold. In this case, we say that A_2 is strongly redundant w.r.t. A_1. Thus, we only return the answer A_1.

The second case arises when there are a set of answers which are redundant amongst themselves (fig. 3(c) and 3(d)). In these two examples, an answer is redundant with respect to the others and vice-versa. In this case, we say that the answers are mutually redundant and we have to choose an answer in this set.

Note that the answer redundancy problem still holds in KBs in irredundant forms, as in the example of fig. 3(d).

Strong and mutual redundancies are based on the redundancy of an answer *w.r.t. another one*:

Definition 7 (Redundancy relation). *An answer A is redundant to an answer A' iff for each referring graph R^A of A, there is a projection π from R^A to a referring graph $R^{A'}$ of A' with $\pi(R_0^A) = \pi(R_0^{A'}) = A'$.*

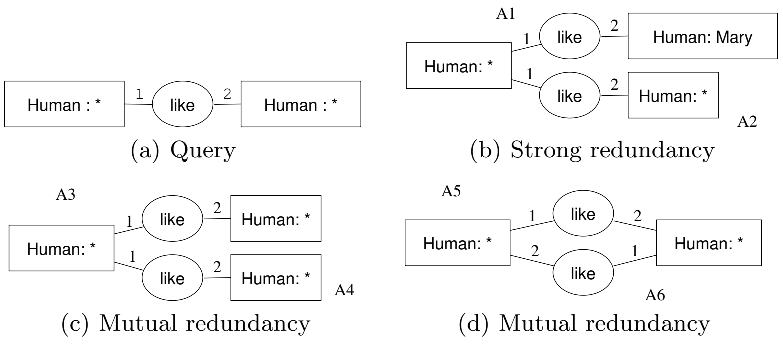


Fig. 3. Different cases of redundant answers

One can test the redundancy relation with the KB referring graph:

Property 2. An answer A is redundant to an answer A' iff there is a projection π from $\langle A, B \setminus A\rangle$ to a referring graph $R^{A'}$ of A' with $\pi(A) = A'$.

Therefore an answer A is redundant to an answer A', the redundancy relation between A' and A defines the two previously seen redundancy cases:

Definition 8 (Redundancies between answers). *Given A and $A' \in Q(B)$, such that A is redundant to A':*

- *A is strongly redundant to A' if A' is not redundant to A ;*
- *A and A' are mutually redundant if A' is redundant to A ;*

Based on the notion of true redundancy, we can refine our notions of *non-redundancy* and *completeness* of subsets of answers that we defined in [2] :

Definition 9 (Non-redundant subset of answers). *A subset of answers $\mathcal{A}$ of $Q(B)$ is non-redundant if there are no two answers A and $A' \in \mathcal{A}$ such that $A \neq A'$ and A is redundant to A'.*

Definition 10 (Completeness). *A subset of answers $\mathcal{A}$ of $Q(B)$ is complete if for each answer A of $Q(B)$ there is an answer A' of $\mathcal{A}$ such that A is redundant to A'.*

We define a *graph of redundancies* based on the notions of strong and mutual redundancies. All types of redundancies of definition 8 can be viewed as relations on $Q(B)^2$, for example (A, A') belongs to the strong redundancy relation if A is strongly redundant to A'. Thus we can characterize properties of these relations. Particularly, mutual redundancy defines equivalent classes over the set of answers, and strong redundancy links all answers of an equivalent class to all answers of another equivalent class. We construct the redundancy graph such that each vertex is an equivalent class, and is linked by the strong redundancy:

Definition 11 (Graph of redundancy). *The graph of redundancy $G = (V, E)$ of $Q(B)$ is a directed graph, where vertices represent equivalent classes of the mutual redundancy relation and where there is an edge between v_1 and v_2 if all answers of the class represented by v_1 are strongly redundant to all answers of the class represented by v_2.*

Fig. 4(a) represents the redundancy graph of query and KB of fig. 1, whereas fig. 4(b) represents the redundancy graph of query of fig. 3(a) and KB defined by the union of KBs of fig. 3(b), 3(c) and 3(d).

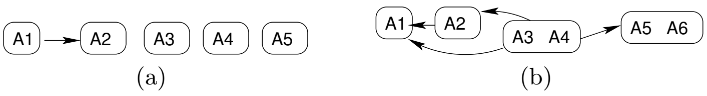


Fig. 4. Redundancy graph of the previous examples

To construct a complete and non-redundant set of answers, one has to only choose a single answer per equivalent class (to avoid mutual redundancy), only from equivalent classes that are sinks (to avoid strong redundancy), and for all of them (to be complete). This is why there is more than one non-redundant complete subset in the second example (fig. 4(b)): there is an equivalent class that is a sink and contains more than one element ($\{A_5, A_6\}$).

Theorem 2. *A subset of answers $\mathcal{A}$ of $Q(B)$ is complete and non-redundant iff for each sink in the redundancy graph there is a single answer of $\mathcal{A}$ that is represented by this sink.*

Proof. Given a non-redundant and complete subset of answers $\mathcal{A}$. Non-redundancy means that there is a single answer for each equivalent class represented in $\mathcal{A}$ and that there are no two answers A_i and A_j such that there is a path from A_i to A_j. Completeness ensures that for all answers A_i of $Q(B)$ (particularly answers belonging to a sink) there is an answer A_j such that A_i is redundant to A_j. So $\mathcal{A}$ has to contain an answer of each sink. When combining completeness and non-redundancy, only one answer of each sink is taken.
• Given a subset of answers $\mathcal{A}$ that is composed of an element of each sink of the redundancy graph. Given two answers of $\mathcal{A}$. These answers are not mutually redundant because there are no two answers of the same equivalent class. These answers are not strongly redundant because each answer comes from a sink. Thus $\mathcal{A}$ is non-redundant. For each answer A_i of $Q(B)$, either there is an answer A_j of the same equivalent class in $\mathcal{A}$ (thus A_j is redundant to A_i), or there is an answer A_k in $\mathcal{A}$ that comes from a sink such that there is a path from the equivalent class of A_i to the sink, and thus A_i is redundant to A_k. $\mathcal{A}$ is complete.

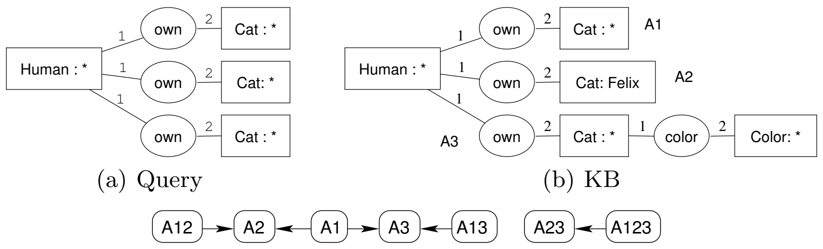
The construction of a redundancy graph combines the problem of finding all projections of a graph into another graph (to compute the set of answers) and the problem of computation of the irredundant form of a graph. Indeed, as stated by property 2, computation of all redundancy links of all answers used to construct the graph is based on all projections of the most specific referring graph of each answer (i.e. a bicolored isomorphic copy of the whole KB). Therefore, a way to compute all redundancy links is to compute all projections from the KB into itself, and check images of a each answer by all projections.

4.4 Inclusion of Answers

As mentioned in section 3.2, the inclusion of answers is one of the problems that can occur. If treated as a kind of duplication, included answers are just deleted from the answer set before computation of a non-redundant complete subset. But this approach is not the best one. We think that the inclusion problem should be treated after the redundancy problem.

In the example of fig. 5, there are seven answers: three that contain only one relation (A_1, A_2, A_3), three that contain two relations (A_{12}[4], A_{13}, A_{23}), and one

[4] Answer A_{ij} represents the answer that is the union of answers A_i and A_j.



(a) Query (b) KB

(c) Redundancy graph

Fig. 5. A query and a KB that produce included answers, and their redundancy graph

with three relations (A_{123}). Considering all answers, only answers A_2, A_3 and A_{23} are irredundant (see fig. 5(c)), but it is not suitable to keep A_2 and A_3. Otherwise, if only non-included answers are kept, this leads to keeping answers that were redundant to deleted answers (here A_{123}, redundant to A_{23}), which is not good.

So the best way is to deal with redundancy first (by computing a subset that is non-redundant and complete) and to delete included answers after that. In the previous example, this strategy led to the subset $\{A_{23}\}$.

4.5 Redundancy at a Considered Distance

In the query framework section, it was stated that computation of the irredundant form of the KB is a difficult problem. But we also see that computation of the redundancy graph also requires finding all projections of the KB into itself. Thus, we propose to restrict referring graphs of an answer to a portion of the base that is "near" this answer. This is also due to the fact that a referring graph that contains too much knowledge, even if it distinguishes an answer, does not help the user much. So we propose to bound referring graphs by a distance k, and to only consider vertices that are in the distance field of one of the vertex of an answer, which forms the *k-neighborhood graph* of the answer:

Definition 12 (k-neighborhood graph). *Given a KB B, a subgraph S of B, and a step k ($k \geq 0$), the k-neighborhood graph, denoted $N^k(S)$, is defined recursively by:*
- *$N^0(S) = S$*
- *$N^{n+1}(S)$ is composed of $N^n(S)$ expanded by every relation r not in $N^n(S)$ and which is linked to a concept of $N^n(S)$, and by all concepts linked to r.*

Recursion stops when $n = k$ or $N^n(S) = N^{n+1}(S)$ and returns $N^n(S)$.

It seems obvious that all definitions put forward previously should now take this constraint into account. Bounding referring graphs can be seen as a restriction of the KB, which is now considered as the union of the k-neighborhood graph of each answer, for a given distance.

Definition 13 (Truncated KB). *Given a KB B, a query Q, and a distance $k \geq 0$, the truncated KB at distance k is $B_k = \bigcup_{A \in Q(B)} N^k(A)$*

Now we can apply all previous definitions to the truncated KB. For example, an answer A of $Q(B)$ is irredundant considering the distance k iff $R = \langle A, B_k \setminus A \rangle$ is a distinguishing graph of A w.r.t. B_k (see corollary 2).

5 Building Smart Answers

In the previous section, redundancy and completeness were only studied from a theoretical standpoint. A more user-aspect oriented standpoint was introduced in the section 4.5. Even if the user gets a subset of answers that is non-redundant and complete, he/she has no way to distinguish answers, i.e. the only assumption that can be made is that there is, for each answer, a way to distinguish them from the others. Otherwise, property 1 states that the most specific referring graph is one of them, but it usually contains too much knowledge.

So what are the properties of a really good set of referring graphs returned to the user? To keep the notions of non-redundancy and completeness, only answers of such a subset of answers should have a referring graph. As all of these answers can be distinguished, all referring graphs should be a distinguishing graph. Finally, to not introduce unnecessary redundancies in the set of referring graphs, there should be only one distinguishing graph by referred answer. A set of referring graphs that fulfills these properties is called a *set of smart answers*:

Definition 14 (Set of smart answers). *A set of smart answers $\mathcal{S}$ of a query Q on a KB B is a set of distinguishing graphs of all answers of a non-redundant complete set $\mathcal{A}$ of $Q(B)$ such as $|\mathcal{S}| = |\mathcal{A}|$.*

We propose the Bounded Smart Answers (BSA) algorithm (see algo. 1) that takes the answers and a distance as parameters, and returns a set of smart answers of truncated KB at distance k such that each extension is the minimal k-neighborhood graph that distinguishes the answer[5].

Theorem 3. *Algorithm $BSA(Q(B), k)$ produces a set of smart answers of Q on the truncated KB B_k.*

Proof. In DAK, all referring graphs are constructed with the same distance. Therefore, thanks to the distance conservation property of the projection, that if there is a bicolored projection from $N^k(A_i)$ to $N^k(A_j)$, there is a projection from $N^k(A_i)$ to B_k such that image of the core of $N^k(A_i)$ is equal to A_j (i.e. $N^k(A_i)$ is a referring graph of A_j in B_k). For each answer A that belongs to an equivalent class that is not a sink, A will not be returned by DAK because of the second "foreach" of DAK. The third foreach of DAK ensures that for each equivalent class that is a sink, the algorithm will only add one (the first taken) answer that belongs to this class once the good distance is reached. Thus the union of all answers returned by DAK is a non-redundant complete subset of answers of B_k. As BSA returns only one graph per answer of the computed non-redundant complete subset, BSA returns a set of smart answers of B_k. $\square$

[5] Note that comparisons in DAK (algo. 2) are between bicolored graphs, that is $B \leq B'$ iff there is a π from B' to B such that $\pi(B'_0) = B_0$.

```
Data: Answers $Q(B)$, a distance $k$
Result: A set of smart answers of truncated KB $B_k$
begin
    $i \leftarrow 0$
    $D \leftarrow \emptyset$ // distinguished answers
    $S \leftarrow \emptyset$ // smart answers
    while $i \leq k$ or $D \neq Q(B)$ do
        $D' \leftarrow$ DISTING_AT_K$(Q(B), D, i)$
        $D \leftarrow D \cup D'$
        foreach $A \in D'$ do
            $S \leftarrow S \cup \{\langle A, N^i(A) \setminus A \rangle\}$
    return $S$
end
```

Algorithm 1. BOUNDED SMART ANSWERS (BSA)

```
Data: Answers $Q(B)$, distinguished answers $D$, a distance $k$
Result: A set $D'$ of distinguished answers at distance $k$
begin
    $D' \leftarrow \emptyset$
    foreach $A \in Q(B) \setminus D$ do
        disting $\leftarrow$ true
        foreach $A_2 \in Q(B) \setminus \{A\}$ do
            if $N^k(A) \geq N^k(A_2)$ and $N^k(A_2) \ngeq N^k(A)$ then
                disting $\leftarrow$ false
        foreach $A_3 \in D'$ do
            if $N^k(A) \geq N^k(A_3)$ then
                disting $\leftarrow$ false
        if disting = true then
            $D' \leftarrow D' \cup \{A\}$
    return $D'$
end
```

Algorithm 2. DISTING AT K (DAK)

6 Conclusion

We extended our previously defined notion of answer redundancy. Our new framework now considers answers but also descriptions (called referring graphs) that can distinguish an answer amongst others. These descriptions could be adapted to query languages of the semantic web, e.g. by giving a formal definition of the SPARQL `DESCRIBE` query form. Therefore this new redundancy is now linked to the whole KB. Based on this new redundancy, we refined two other previous definitions concerning subsets of answers: *non-redundancy* property (there is no redundant answer in the subset) and *completeness* (each answer is redundant to an answer of the subset). We proposed a way to construct all non-redundant and complete subsets of answers using a *redundancy graph*. We introduced the notion of smart answer and gave an algorithm that computes a set of smart answers based on a vertex neighborhood distance.

Answer redundancy arises because of open world assumption and undefined objects (generic concepts). A deeper redundancy still exists, that is not related to the formalized world (the KB), but rather to the "real world" (described by the KB). For example, it is possible that non-redundant answers "a big cat" and "a white cat" refer to a single cat of the "real world", which is big and white. The study of this new kind of redundancy can provide a foundation for using aggregation operators (e.g. number of results to a query) in graph based KB query languages.

References

1. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. Technical report, W3C (2008)
2. Leclère, M., Moreau, N.: Query-answering cg knowledge bases. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS, vol. 5113, pp. 147–160. Springer, Heidelberg (2008)
3. Lin, J.J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., Karger, D.R.: What makes a good answer? the role of context in question answering. In: INTERACT (2003)
4. Croitoru, M., van Deemter, K.: A Conceptual Graph approach to the Generation of Referring Expressions. In: Proceedings of the 20th International Joint Conference on Artificial Intelligenc, IJCAI 2007 (2007)
5. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of semantic web databases. In: PODS 2004: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 95–106. ACM Press, New York (2004)
6. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL, a language for deductive query answering on the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web 2(1), 19–29 (2004)
7. Chein, M., Mugnier, M.-L.: Conceptual Graphs: Fundamental Notions. Revue d'Intelligence Artificielle 6(4), 365–406 (1992)
8. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
9. Baget, J.-F., Mugnier, M.-L.: The SG family: Extensions of simple conceptual graphs. In: IJCAI, pp. 205–212 (2001)
10. Mugnier, M.L.: On generalization/specialization for conceptual graphs. Journal of Experimental & Theoretical Artificial Intelligence 7(3), 325–344 (1995)

A Practical Exploration of Ontology Interoperability

Simon Polovina, James Cooke, and Jeremy Loke

Cultural, Communication & Computing Research Centre (CCRC)
Sheffield Hallam University, Sheffield, United Kingdom
{s.polovina,j.loke}@shu.ac.uk, jcooke3@hera.shu.ac.uk

Abstract. ISO Common Logic (CL, ISO/IEC 24707:2007) offers the Semantic Web (SW) a new and powerful dimension in achieving the effective discovery, automation, integration, and reuse across applications, data and knowledge. The paper shows how it is possible to explore such interoperability through small scale exemplar projects. As Conceptual Graphs (CG) is a key technology in CL, we focused on the Amine CG software and for the SW we focused on the Protégé OWL software, exploring the possible mappings between ontologies captured in OWL and in Amine. Through this practical exercise the dimensions and extent of the desired interoperability could be demonstrated. This small but significant experiment provided a practical insight into how CG Tools can actually interoperate towards achieving the wider goal of Ontology interoperability between CL and the SW.

1 Introduction and Motivation

At the Panel Session during the Conceptual Structures Tools Interoperability Workshop (CS-TIW) at ICCS 2008 (www.inra.fr/iccs08/workshops) one of us (Polovina) expressed concern over the ongoing poor progress of Conceptual Graphs (CG) software tools' ability to interoperate with one another. It was shown how interoperability could be progressed in a practical way through small-scale exemplar projects such as an Ontology Importer for Amine [1]. One of the key ensuing comments that if we wanted interoperability was for individuals simply to do it for themselves. Also arising from the session was how CG tools could interoperate with the Semantic Web (SW), reflecting the desires and developments towards this goal [3,8]. Thus we had a spectrum from exploring interoperability that ranged from a being a 'cottage industry' to globally interoperating CG tools with the Web itself.

In this larger view, CG is a key technology in the ISO Common Logic (CL) standard (ISO/IEC 24707:2007)[4]. ISO CL propels 'non-SW' technologies like CG (and their tools) from being disparate cottage industries into the global arena of the SW. This arises from the fact that standards play a key part in the adoption of any technology however promising that technology is. Standards diminish the risk of being locked into a non-interoperable technology. Interoperability across standards offers any technology the most appropriate standard for it to belong to whilst allowing it to be used with technologies in other standards according to their individual strengths. ISO CL accordingly offers the SW a new and powerful dimension in achieving its aims of the effective discovery, automation, integration, and reuse across applications, data and knowledge. Interoperating W3C's SW recommendations with ISO CL offers the most lucrative route in best realising these aims.

The 'challenge' of 'doing it ourselves' may nonetheless continue to offer a worthwhile route in this larger endeavour. Given the previous experience of the Amine Ontology Importer project referred to earlier, and in taking up this remark, we present a further small-scale exemplar practical project. Namely we investigated how an ontology produced in an emergent ISO CL software tool can interoperate with a SW one through interoperating CG's Amine software (amine-platform.sourceforge.net) with the SW's Protégé OWL (Web Ontology Language) software (protege.stanford.edu).

2 CG and OWL

CG and OWL provide various layers of functionality for supporting ontologies. CG are a way of structuring knowledge in a form readable by both humans and computers. CG can be read in a graphical or linear manner [9]. CG are constructed from Concepts, Relations and Arcs. Concepts consist of a type name within a rectangle, they may also have a referent which refers to an individual or instance of that type. Relations are shown as the relation type name within a circle or ellipsis and refer to the relationship between the two concept types. Arcs are shown as arrows between the concepts and the relations, the direction of the arrow dictates which way the formal logic should be read. If the arrow is directed towards the relation then the relationship will generally be 'has a', if the arrow is directed away from the relation then the relationship will generally be 'which is/who is/is a'.

Thus `[Concept-1] -> (Relation) -> [Concept-2]` states that "Concept-1 has a Relation which is a Concept-2". `[Cut] -> (Inst) -> [Knife]` denotes that "Cut has an Inst(rument) which is Knife". (In CG is generally easier to start the description with the relation, so for this example we would state that "The Instrument of Cut is a Knife".) An introduction to CG provides a description in more detail including the use of referents [6].

In CG, an ontology is a hierarchy of Concept Types and Relations with Universal at the topmost super-type of all types and Absurd at the bottommost subtype of all types; as such an ontology denotes a 'catalogue of modes of existence' [9], and the purpose of an ontology is to model knowledge in a formally logical structure so that facts can be derived from it.

OWL is a SW technology (www.w3.org/TR/owl-features/) that uses Web technologies, such as XML and Uniform Resource Identifiers (URI, www.ietf.org/rfc/rfc2396.txt) which uniquely identify ontologies and elements within ontologies across the Web. The OWL language is based on the SW's XML/RDF Schema (www.w3.org/RDF). OWL is a high level mark-up language that can easily read by humans as well as computers in its raw form.

2.1 The Amine Suite

The Amine CG platform provides an entire suite of applications aimed at creating complex intelligent systems. Amine ontologies allow the use of CG in their structure both as 'Definitions' and 'Canons'. The difference between a Canon and a Definition is essentially that the former describes a good use of a CG whilst the latter is a definition of a given Concept or a Relation, and examples distinguishing the two can be found (e.g. www.huminf.aau.dk/cg/Module_III/1152.html). Canons and Definitions provide an invaluable and powerful basis to structure a model of knowledge based on the CG ontology format described above. Each of these nodes can contain a Canon or a Definition in the form of a CG which relates to other nodes in the structure to form the inherent conceptual links which the ontology carries. The Amine ontologies are stored as XML files, and it is through parsing between Amine's Ontology XML format and Protégé OWL's XML format that interoperability across the two ontology formats is explored.

2.2 Protégé OWL

Protégé OWL is a platform which enables creation and manipulation of ontologies in the W3C's OWL format. This platform provides an intuitive interface enabling a graphical representation of an OWL ontology. It may include descriptions of classes, properties and their instances that are used to build the model of knowledge used to reason facts. There are three subsets of the OWL language: OWL lite, OWL DL and OWL full. OWL DL was judged to be the most appropriate level by which to investigate interoperability across the two ontology formats as it supports Description Logic (The 'DL' in OWL DL), given the previous correspondence between CG and DL [2]. In passing, OWL uses URI to identify objects it allows inter ontology transfer of data over the Web, a simple functionality that does not exist in the Amine platform at present but could easily be implemented.

3 Mapping between Amine Ontology and Protégé OWL

We based our approach on JXML2OWL, a project using a Java XML DOM parsing approach to migrate data from a standard XML data structure to the OWL Web Ontology Language. However we used "LINQ To XML" parsing in VB.NET to conduct the mappings [5,7]. This choice capitalised on the particular experience that one of us had (Cooke). We accordingly explored the pathways between a Protégé OWL and Amine ontology, creating it in one of them and testing the parsed ontology in the other. Our findings were as follows.

3.1 Amine Concepts to OWL Classes

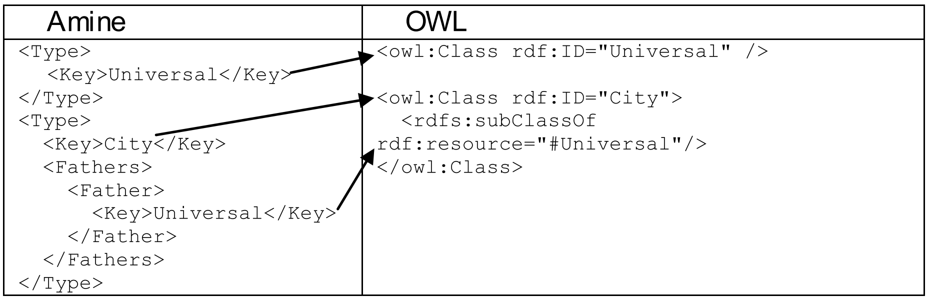
Amine Concept Types. In Amine, concept types represent a type of a concept. The concept types build up a hierarchy of concepts in Amine through their children and fathers. For example, the following XML describes the concept type 'City', and its father (Universal):

```
<Child>
  <Key>City</Key>
</Child>
<Fathers>
<Father>
  <Key>Universal</Key>
</Father>
```

OWL Classes. To construct the same hierarchy, OWL uses Classes, and the subClassOf construct to define the Super-Class (or in Amine terms, the 'father'). For example, the following XML describes the classes Universal and City and the hierarchy between them via the subClassOf construct:

```
<owl:Class rdf:ID="Universal" />
<owl:Class rdf:ID="City">
 <rdfs:subClassOf rdf:resource="#Universal"/>
</owl:Class>
```

Analysis. Concept types in Amine can be mapped across to OWL easily as Classes, in addition the structure of the Concept Types can be translated using OWL's (or to be specific, RDFs) subClassOf feature. Using this simple mapping, the Amine ontology type hierarchy can be translated into an OWL file and still maintain its Sub-Type/Super-Type structure, this file can then be modified or expanded using the Protégé OWL Suite. This is achieved by parsing the Amine XML and finding all of the `<Type>.<Key>` nodes. Then within each node find all of the `<Fathers>.<Father>.<Key>` nodes. If there are no `<Fathers>.<Father>.<Key>` nodes present in a `<Type>` node then there is no subClassOf construct in the OWL XML, illustrated by Figure 1.



Amine

```
<Type>
  <Key>Universal</Key>
</Type>
<Type>
  <Key>City</Key>
  <Fathers>
    <Father>
      <Key>Universal</Key>
    </Father>
  </Fathers>
</Type>
```

OWL

```
<owl:Class rdf:ID="Universal" />

<owl:Class rdf:ID="City">
  <rdfs:subClassOf
rdf:resource="#Universal"/>
</owl:Class>
```

Fig. 1. Amine Concept to OWL Classes

3.2 Amine Relation Types to OWL Properties

Amine Relation Types. In Conceptual Graph theory a relation type is a type of relation which can contain a CG. Amine defines a Relation Type as a conceptual structure which is always a subtype of the special 'Relational Root'. In line with CG theory, Amine allows a relational type to have a CG. For example, the following XML is direct output from Amine and shows the definition of the Relation Type ‘Agnt’ and that it’s Super-Type is ‘Relation’:

```
<RelationType>
  <Key>Agnt</Key>
  <Fathers>
    <Father>
      <Key>Relation</Key>
    </Father>
  </Fathers>
</RelationType>
```

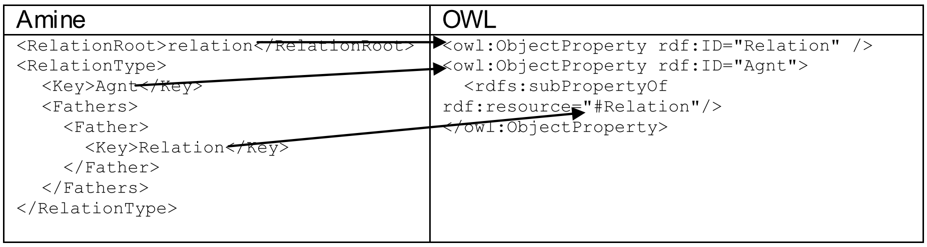
OWL Properties. To construct the same hierarchy, OWL uses Properties, and the subPropertyOf construct to define the Super-Class (or in Amine terms, the “father”). For example, the following XML describes the Properties ‘Relation’ and ‘Agnt’ and the hierarchy between them via the subPropertyOf construct:

```
<owl:ObjectProperty rdf:ID="Relation" />
<owl:ObjectProperty rdf:ID="Agnt">
  <rdfs:subPropertyOf rdf:resource="#Relation"/>
</owl:ObjectProperty>
```

Analysis. An Amine ontology can include conceptual Relation Types, integrating a relational type hierarchy with the Concept Type hierarchy to capture the relations between the concept types [9]. These can be mapped to OWL properties and using RDF’s subPropertyOf feature, enabling the Sub-Type/Super-type structure of the type hierarchy to be maintained. These similarities lead us to believe that there is a pathway to convert Amine Relation Types to OWL Properties through 2 stages:

1. Parse the XML to find the `<RelationRoot>` node and its value, then create a OWL property with no domain or range as the root property.
2. Parse the XML and find all of the `<RelationType>.<Key>` nodes (except for the Relation Root) then search within each of the `<RelationType>` nodes (except for the Relation Root) for `<Fathers>.<Father>.<Key>` nodes then writing the OWL ObjectProperty and subPropertyOf.

Figure 2 provides an example.


Amine

```
<RelationRoot>relation</RelationRoot>
<RelationType>
  <Key>Agnt</Key>
  <Fathers>
    <Father>
      <Key>Relation</Key>
    </Father>
  </Fathers>
</RelationType>
```

OWL

```
<owl:ObjectProperty rdf:ID="Relation" />
<owl:ObjectProperty rdf:ID="Agnt">
  <rdfs:subPropertyOf
rdf:resource="#Relation"/>
</owl:ObjectProperty>
```

Fig. 2. Amine Relation Types to OWL Properties

3.3 Amine Individuals to OWL Individuals

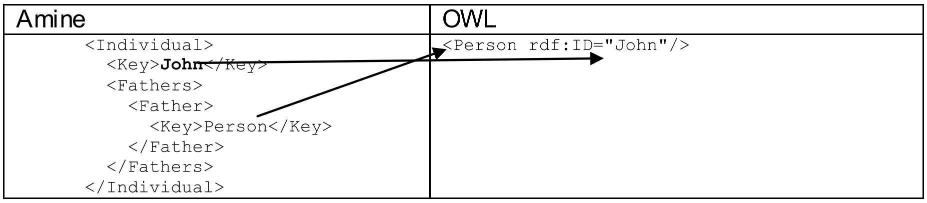
Amine Individuals. In Amine, an individual is a specific member of a group. For example, the following XML describes the Individual 'John' and that he is a member of the type 'Person':

```
<Individual>
  <Key>John</Key>
  <Fathers>
    <Father>
      <Key>Person</Key>
    </Father>
  </Fathers>
</Individual>
```

OWL Individuals. To construct the same hierarchy OWL uses the coincidently named Individuals, the definition of OWL Individuals is simpler than Classes and Properties. For example:

```
<Person rdf:ID="John"/>
```

Analysis. In an Amine ontology an individual is an instance of a concept type, OWL also has individuals which are defined as instances of classes. Following our mappings thus far this is a natural conversion. First we must parse the XML and find all of the `<Individual>` nodes, for each node write the `<Individual>.<key>` value as the Individual's name then Write the `<Individual>.<Fathers>.<Father>.<key>` value as the name of the class this individual belongs to. Figure 3 provides an example.



Amine	OWL
`<Individual>` `<Key>John</Key>` `<Fathers>` `<Father>` `<Key>Person</Key>` `</Father>` `</Fathers>` `</Individual>`	`<Person rdf:ID="John"/>`

Fig. 3. Amine Individuals to OWL Individuals

3.4 Amine CG to OWL Mapping

So far we have proposed pathways/mappings to convert between an Amine ontology and Protégé OWL. However we noted that these mappings do not take advantage of the expressiveness of CG and their use within Amine. Therefore we lay out a method for translating CG in an Amine ontology to an appropriate structure within an OWL ontology.

In an Amine ontology Concept Types and Relation Types have the capability to contain both definitions and canons in the form of CG; these CG connect the nodes together and define the relationships between them. Here a parallel can be drawn with properties of OWL.

Figure 4 tabulates the mappings.

Amine ->	-> OWL	Comments
Concept Type	Class	Mapping is simple although Universal name will be lost.
Relation Type	Properties	In OWL, properties are dyadic; in CG, relations are n-adic. We thus used Class conditions to assert relations.
Individuals	Individuals	Maps
Definition	Class Conditions	Definition CG is parsed and the relations are created as a class condition.
Canon	No support	A canon cannot be translated as only one model of knowledge may be defined, but this can be saved as an RDF comment
Situation	No support	No support in OWL for situations as they model a specific moment in time; but this can be saved as an RDF comment.
Synonyms	No support	Could be added as an RDF comment.

Fig. 4. Amine-OWL mappings

4 Binary Relations vs. n-adic Relations

Considering we have a larger more complex CG with many arcs we would break them up into smaller, binary CG and model them as OWL properties one by one. With the nature of the n-adic relations of CG and the Binary relations of OWL it may not seem as simple for these two technologies to interoperate as this paper implies. However using the pathways identified in this paper the complex many arced CG are broken down into binary OWL Properties.

4.1 The n-adic Dimension

OWL properties along with their domain and range can be used to express a dyadic relationship between individuals. However as CG may be n-adic we need to employ Class Conditions in OWL in order lay out the logic as it appears in CG. Using necessary asserted conditions we can map individuals, properties and classes together and translate some of the model of knowledge contained within the Amine ontology.

In order to achieve this we must first programmatically set the Domain and Range of all of the Properties. This can only be done if the property (or in Amine, the Relation) is used somewhere in the ontology. For example if we had in a Amine ontology with the relation 'Agnt' but this relation was never used in a CG, the computer would have no way to know what concepts it exists to relate. For this reason we will assume that all of the Relation types in the Amine ontology we are converting to OWL are referenced by a CG somewhere.

Let us take a CG from the previous example: `[Go] -> (Inst) -> [Bus]` This CG is the Definition of the 'Go' concept in an Amine ontology, from this CG we must get the Domain and Range for the "Inst" Property.

Domain. It is logical that the domain will be a Super-Type of the 'Go' Concept Type, however there could be n-amount of Super-Types for the Concept Type in question, its direct Super-Type could be 'Action', should this be the domain?, the Super-Type of 'Action' could be 'Movement', should this be the domain? The computer has no way of knowing what the correct Domain for a given Concept Type should be with reference to creating an OWL Property, it is for this reason that we have decided to use the direct Super-Type as it cannot be incorrect and will be valid for other uses of the Concept Type. For example now that we have established that the Domain of the property 'Inst' is 'Action' this domain will still be valid when used in a CG like this: `[Return] -> (Inst) -> [Bus]` as the Super-Type of 'Return' is also 'Action'.

Range. The same can be said of the Range of the Property 'Inst' as if we use the Super-Type of 'Bus' which is 'Vehicle', the property would still be valid for a CG like this `[Return] -> (Inst) -> [Car]`. However this will cause an issue if we encounter a CG like this: `[Cut] -> (Inst) -> [Knife]` as 'Knife' is not a Sub-Type of 'Vehicle'. It is for this reason that when the parser encounters a scenario like this the Domain or Range (whichever caused the error) will be 'pushed up' from its existing hierarchical level to a level which will satisfy both of the CG. For example if the above CG is encountered and causes a conflict, the Range will be changed from 'Vehicle' to a concept type (or in OWL Class) which is a Super-Type of 'Car', 'Bus' and 'Knife', in this case it would be 'Entity'.

4.2 Class Restrictions

Now that we have established a method of getting the Domain and Range of the for the OWL properties we can begin asserting conditions on the Classes based on CG from the Amine Ontology.

To continue with our simple example: `[Go] -> (Inst) -> [Bus]` In order to translate this knowledge 'The instrument of Go is Bus' we need to use a restriction on the 'Go' Class in the OWL ontology, this is because this statement is changing the structure of the 'Go' Class. Programmatically this is decided by which ever concept in the CG is logically read first.

When the parser reaches the point of asserting Class Conditions and it encounters a CG it will take the first logical concept, in this case 'Go' and add an OWL Restriction. The onProperty attribute will set the property which restricts the Class, in this case the 'Inst' Property. As the second relation in this CG ('Bus') is a Concept and not an Individual, the construct someValuesFrom is used to allow any Sub-Type of 'Bus' to be accepted as valid.

Figure 5 illustrates.

CG	OWL
`[Go] > (Inst) > [Bus]`	`<owl:Class rdf:about="#Go">`
	`  <owl:Restriction>`
	`    <owl:onProperty rdf:resource="#Inst"/>`
	`    <owl:someValuesFrom rdf:resource="#Bus"/>`
	`  </owl:Restriction>`
	`</owl:Class>`

Fig. 5. Class restrictions

5 Levels of Conceptual Interoperability

We informally assessed the mappings according to the Levels of Conceptual Interoperability Model (LCIM) [10,11]. We were pleased to detect that even within this small-scale project LCIM level 3 (the semantic level) could be achieved, and elements of level 4 (the pragmatic/dynamical level) may be present. In our view Level 5 (the conceptual level, a common view of the world is established i.e. a way to formalize the knowledge about a given domain) would not presently be attained. For this we would turn to the larger-scale CL-SW interoperability projects alluded to towards the beginning of this paper. But considering the ambitious nature of such a small project, "$3\frac{1}{2}$ out of 5 isn't bad".

6 Concluding Remarks

Through this practical exercise we have managed to demonstrate the actual dimensions and extent of useful interoperability. Whilst originally taken in good humour, 'doing it yourself' can nonetheless provide useful results and a context and direction for larger scale projects. The success of our small 'cottage industry' project demonstrates that work in the interoperability arena is not as impossible a task as it may seem. Ours indeed was a small but significant experiment that provided a practical insight into how CG Tools can actually interoperate towards achieving the wider goal of Ontology interoperability between CL and the SW.

References

1. Abdulrub, S., Polovina, S., Sandberg-Petersen, U., et al.: Implementing interoperability through an ontology importer for Amine. In: Croitoru, M., Jäschke, R., Rudolph, S. (eds.) CS-TIW 2008 Proceedings, CEUR-WS, Germany, vol. 352, pp. 1–6 (2008)
2. Coupey, P., Faron, C.: Towards correspondences between Conceptual Graphs and Description Logics. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS (LNAI), vol. 1453, pp. 165–178. Springer, Heidelberg (1998)
3. Corby, O.: Web, Graphs and Semantics. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 43–61. Springer, Heidelberg (2008)

4. Delugach, H.S.: Towards Conceptual Structures Interoperability Using Common Logic. In: Croitoru, M., Jäschke, R., Rudolph, S. (eds.) CS-TIW 2008 Proceedings, Germany. CEUR-WS, vol. 352, pp. 13–21 (2008)
5. Meijer, E., Beckman, B., Bierman, G.: LINQ: Reconciling Object, Relations and XML in the.NET Framework. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, p. 706. ACM, New York (2006)
6. Polovina, S.: An introduction to conceptual graphs. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 1–15. Springer, Heidelberg (2007)
7. Rodrigues, T., Rosa, P., Cardoso, J.: Moving from Syntactic to Semantic Organizations using JXML2OWL. Computers in Industry 59(8), 808–819 (2008)
8. Rudolph, S., Krötzsch, M., Hitzler, P.: Quo Vadis, CS? - On the (non)-Impact of Conceptual Structures on the Semantic Web. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 464–467. Springer, Heidelberg (2007)
9. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
10. Tolk, A., Muguira, J.: The Levels of Conceptual Interoperability Model (LCIM). In: Fall Simulation Interoperability Workshop, Washington DC (April 2004)
11. Tolk, A., Turnitsa, C., Diallo, S.: Implied Ontological Representation within the Levels of Conceptual Interoperability. Intelligent Decision Technologies 2, 3–19 (2008)

Relation Algebra Operations on Formal Contexts

Uta Priss

Edinburgh Napier University, School of Computing
www.upriss.org.uk

Abstract. This paper discusses the use of relation algebra operations on formal contexts. These operations are a generalisation of some of the context operations that are described in the standard FCA textbook (Ganter & Wille, 1999). This paper extends previous research in this area with respect to applications and implementations. It also describes a software tool (FcaFlint) which in combination with FcaStone facilitates the application of relation algebra operations to contexts stored in many formats.

1 Introduction

Relation algebra (RA) operations provide a generalisation of some of the context operations that are described in the standard Formal Concept Analysis (FCA) textbook (Ganter & Wille, 1999). Using relation algebra is of interest because it provides an alternative to SQL-based conceptual modelling. Currently, some FCA software tools allow users to perform a limited number of context operations, for example, generating the dual context which switches objects and attributes. But the complete set of RA operations did not use to be available. Furthermore, software exists[1] which allows to extract formal contexts from relational databases via SQL queries. But FCA users who are not relational database experts may find it difficult to translate their natural language queries into SQL. In many applications, users therefore have to manually edit their formal contexts until they have the desired format and FCA visualisations can be applied. There is software for RA operations[2], but this is not designed for FCA and cannot easily be used for formal contexts. As far as we know up to now there has not been an extensive analysis of RA operations on FCA contexts and there has not been a software tool that implements RA operations systematically. This situation is changed by the new FcaFlint software[3] which provides an interface for performing RA operations directly on formal contexts stored in a variety of formats.

This paper starts with an introduction to relation algebra (Section 2). It then continues with an exploration of using RA for formal contexts (Section 3) and context schemata (Section 4). Section 5 considers RA for the modelling of lexical databases. Previously, the use of relation algebra for the modelling of lexical databases has been described (Priss & Old, 2006). Because, lexical databases are usually quite large, it

[1] Tupleware available at http://tockit.sourceforge.net/

[2] http://www.informatik.uni-kiel.de/~progsys/relview.shtml

[3] FcaFlint will be bundled with the next edition of FcaStone available at http://fcastone.sourceforge.net/

is not practical to generate the lattice of a formal context that contains all of the data of the lexical database. Thus mechanisms must be provided that allow for the extraction of meaningful smaller contexts. Furthermore over time, users of lexical databases may want to extract different types of formal contexts, thus the extraction of formal contexts must be achieved in a fairly flexible manner. Relation algebra provides a useful framework for modelling the context extraction for such databases. Results presented in previous research (Priss & Old, 2006) are extended and elaborated in this paper.

The paper continues (in Section 6) with a discussion of the FcaFlint software tool which in combination with FcaStone facilitates the application of RA operations to contexts stored in a variety of formats. FcaFlint allows to enter RA queries in a textual format. The implementation of FcaFlint is based on the description of a theoretical means for establishing an RA for formal contexts by Priss (2006). But the RA operations described in that paper are more theoretically than practically useful because they are inefficient (and to some degree incomplete). For practical implementations some modifications are needed as discussed in Section 7.

2 Relation Algebra

Relation algebra was invented and developed by Peirce, Tarski and others. Priss (2006) discusses the modelling of FCA with RA and provides some background and references which shall not be repeated in this paper. An introduction to using RA with FCA, which contains detailed examples of the operations is available elsewhere[4]. RA can serve as an alternative to the more common modelling of FCA operations with Peirce Algebraic Logic (PAL) or Relational Algebra[5] (RLA). The difference between RA and RLA is that RA operates on binary relations while RLA operates on n-ary relations. Both RA and RLA can be used for the modelling of many-valued contexts or power context families. Since concept lattices correspond to binary relations, at the visualisation stage a binary relation must be extracted from the many-valued data, for example, by using the process of conceptual scaling (Ganter & Wille, 1999). The difference is that with RLA the binary context is extracted at the end after the context operations have been applied to n-ary relations, whereas with RA (with the addition of a Fork algebraic operation) n-ary relations are encoded in a binary format from the start. The details of this process are described by Priss (2006). In any case, both RA and RLA are relevant for FCA and ideally there should be FCA software for RA and RLA operations. This paper focuses on RA operations, which can be seen as a generalisation of some of the context operations discussed by Ganter & Wille (1999).

A detailed formal notation of FCA with RA is quite complex because all operations need to consider the sets of formal objects, attributes and the matrices of the contexts. In this paper, we simplify the notation and focus mostly on the Boolean matrices representing the binary relations of the contexts. It is assumed that the operations are applied

[4] An “Introduction to using Relation Algebra with FCA” can be downloaded from: http://www.upriss.org.uk/fca/relalg.html

[5] The similarity in the names of Relation versus Relation**al** Algebra is unfortunate, but these are the names that are established in the literature. RLA is Codd’s algebra for relational databases.

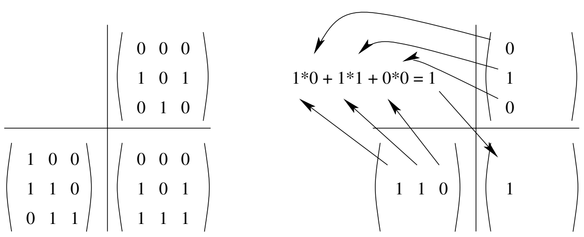


Fig. 1. Binary relational composition, $I \circ J$

sensibly with respect to the sets of formal objects and attributes. For example, in the case of union and intersection, the matrices need to have the same dimensions.

Definition 1. *A matrix-RA is an algebra $(R, \cup, ^-, one, \circ, ^d, dia)$ where R is a set of square Boolean matrices of the same size; one is a matrix containing all 1s; dia is a matrix, which has 1s on the diagonal and 0s otherwise; $\cup, ^-, \circ, ^d$ are the usual Boolean matrix operations. $\cap$ and nul are defined as $I \cap J := \overline{\overline{I} \cup \overline{J}}$ and $nul := \overline{one}$.*

Boolean (or binary) matrices are elaborated by Kim (1982). Boolean matrix operations are exactly like normal matrix operations except that the matrices contain only 0s and 1s and use Boolean OR, AND, and NOT for the componential operations. For example, Fig. 1 illustrates relational composition. The operations $\subseteq$ and $=$ are as usual: $I \subseteq J :\Longleftrightarrow I \cap J = I$ and $I = J :\Longleftrightarrow I \subseteq J$ and $J \subseteq I$. It can be shown that a matrix-RA is an RA and fulfills all the axioms of an RA, such as $(R, \cup, \cap, ^-, nul, one)$ is a Boolean algebra; $\circ$ is associative and distributive with $\cup$; dia is a neutral element for $\circ$ (but unique inverse elements need not exist); $(I^d)^d = I$; $(I \cup J)^d = I^d \cup J^d$; $(I \circ J)^d = J^d \circ I^d$; and so on (cf. Priss, 2006). RA has the expressive power of First Order Logic (FOL) with three variables. For finite matrices a transitive closure of composition can be defined: $I^{trs} := I \cup (I \circ I) \cup (I \circ I \circ I) \cup ....$ But this is not an RA operation and not FOL.

Boolean matrices have many interpretations. If they are interpreted as binary relations, they can be used to check properties. A binary relation is:

- symmetric if $I = I^d$,
- reflexive if $dia \subseteq I$,
- transitive if $I \circ I \subseteq I$,
- antisymmetric if $I \cap I^d \subseteq dia$,
- a partial ordering if $I \cap I^d = dia$ and $I \circ I = I$,
- surjective if $dia \subseteq I^d \circ I$ (or I has at least one 1 per column),
- injective if $I \circ I^d \subseteq dia$ (or I has at most one 1 per column).

If Boolean matrices are interpreted as the incidence relation of a directed graph, the transitive closure I^{trs} shows the reachability for travelling along the graph's edges. The transitive closure $(I \circ I^d)^{trs}$ shows the connectedness in an undirected graph. The next section discusses Boolean matrices interpreted as formal contexts.

3 Formal Contexts as Boolean Matrices

Ganter & Wille (1999) discuss context constructions and operations, some of which are RA operations, some are not. In their Definition 30, the complementary context K^c ($\overline{I}$ in our notation) and the dual context are RA operations. Ganter & Wille write X and Ø instead of one and nul, respectively. Ganter & Wille's apposition and subposition are non-RA operations, but are frequently used with FCA (see the next section). Ganter & Wille's disjoint union is not an RA operation because it involves apposition and subposition.

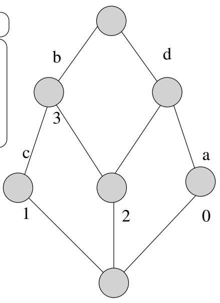
As mentioned above, in order for RA formalisms to be meaningful for FCA, matrix operations should only be applied if the involved contexts have appropriate sets of objects and attributes. For example in the case of composition, the set of attributes of the left matrix should correspond to the set of objects of the right matrix. This means that the sets need to contain the same elements and need to be in the same linear order. In the rest of this paper, it shall always be assumed that the RA operations are applied sensibly without explicitly mentioning the details about the sets of objects and attributes. Furthermore, the RA operations are also applied to non-square matrices and it is silently assumed that the dimensions of the matrices are compatible as needed. The matrix dia always needs to be square, but nul and one adjust their dimensions to the matrices they are unioned or intersected with.

An interesting question, which as far as we know has not yet been discussed elsewhere, is to investigate the expressiveness of RA with respect to FCA. Because the number of concepts in a concept lattice tends to be different from the numbers of objects and attributes, but RA operations do not radically change the matrix dimensions, it is immediately apparent that it is not possible to generate concept lattices from formal contexts using RA operations. But, as will be shown below, it is possible to calculate basic FCA facts about the objects and attributes and their ordering using just RA operations.

This paper does not contain an introduction to FCA (which can be found in Ganter & Wille (1999)). But a few notions shall be mentioned here: for a formal context (G, M, I), the *prime operator* retrieves *intensions* of concepts if applied to sets of objects and *extensions* of concepts if applied to sets of attributes. For $G_1 \subseteq G$, $G_1' := \{m \in M \mid gIm \text{ for all } g \in G_1\}$. Dually, for $M_1 \subseteq M$, $M_1' := \{g \in G \mid gIm \text{ for all } m \in M_1\}$. The *plus operator* uses an EXISTS-quantifier instead of an ALL-quantifier: $G_1^+ := \{m \in M \mid gIm \text{ for one } g \in G_1\}$. An *object concept* γg is the smallest concept to which that object belongs. It is the concept which is labelled by the object in the usual graphical representation of a concept lattice. An *attribute concept* μm is the largest concept to which that attribute belongs. The *object order* $\leq_o$ of a concept lattice is an order on the set of objects derived from the order among the object concepts in the concept lattice: $g_1 \leq_o g_2 \iff \gamma g_1 \leq \gamma g_2$. The *attribute order* is defined analogously.

For a formal context (G, M, I), the following examples indicate what kind of information can be derived about the conceptual structure by using just RA operations (described here for the set of objects, but dually for the set of attributes):

	a	b	c	d
0	X			X
1		X	X	
2		X		X
3		X		



	0	1	2	3
0	X			
1		X		X
2			X	X
3				X

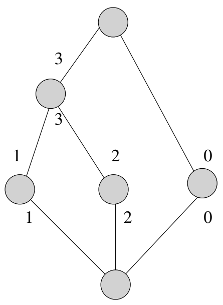


Fig. 2. A formal context (left) and its object order (right), $\overline{\overline{I} \circ I^d}$

1. Sets of objects that share at least one attribute with each other: $(G, G, I \circ I^d)$. This matrix is symmetric: $(I \circ I^d)^d = (I^d)^d \circ I^d = I \circ I^d$. Each row or column contains g^{++}. It represents applying the plus operator twice.
2. An equivalence relation on the objects based on the horizontal decomposition of the lattice (see Priss & Old, 2006): $(G, G, (I \circ I^d)^{trs})$.
3. Objects which have no attributes in common: $(G, G, \overline{I \circ I^d})$.
4. The object order: $(G, G, \overline{\overline{I} \circ I^d})$. Each column in this context shows the extension of the object concept of that object: g''. It represents applying the prime operator twice. The lattice of this context displays the object order. Fig. 2 shows an example.

Thus the RA operations are able to extract some information about the relationships among the objects (and attributes, respectively) but not the full lattice. Also, although the arrow relations (Ganter & Wille, 1999) are representable within a formal context, it is unlikely that these can be generated with RA operations because two different formal contexts with the same object orders can have different arrow relations. Thus, if RA can only extract the object and attribute orders, it is not sufficient to calculate the arrow relations.

4 Examples of Context Schemata

This section and the next one show some more examples of how RA operations can be used for FCA applications. Apposition and subposition are not RA operations, but they can be used to iteratively build a context from smaller contexts. A *context schema* consists of four (or nine etc) contexts built via apposition and subposition, such as the examples in Table 1 and Fig. 3. Quite often some of the contexts in a context schema are functionally dependent on other ones in the same schema. For example, in schemata 5 and 6 in Table 1 the context in the lower right corner is derived by composition. In general, the functions used for building context schemata are not restricted to RA operations. If the functions used for building a context schema are solely RA operations and nul, one, dia, these are called *RA context schemata*. It might be an interesting research question, to examine the algebraic structure that is formed by RA context schemata.

Table 1 shows examples of RA context schemata. The simplest examples (1-3) are listed in Ganter & Wille (1999) and yield the horizontal and vertical sums and the

Table 1. RA context schemata

Context schema	Effect on the lattices
1) $\begin{array}{c\|c} nul & I \\ \hline J & nul \end{array}$	Horizontal sum (Ganter & Wille, 1999).
2) $\begin{array}{c\|c} one & I \\ \hline J & nul \end{array}$	Vertical sum (Ganter & Wille, 1999).
3) $\begin{array}{c\|c} one & I \\ \hline J & one \end{array}$	Direct product (Ganter & Wille, 1999).
4) $\begin{array}{c\|c} dia & I \\ \hline nul & dia \end{array}$	All object and attribute concepts are turned into atoms/co-atoms.
5) $\begin{array}{c\|c} dia & J \\ \hline I & I \circ J \end{array}$	Composition of two lattices.
6) $\begin{array}{c\|c} dia & I^d \\ \hline I & I \circ I^d \end{array}$	Vertical gluing of a lattice and its dual.

direct product of concept lattices. Many other constructions of RA and non-RA context schemata are known (e.g. Ganter & Wille, Section 1.4).

The following examples demonstrate how classification can be represented with context schemata and RA operations. In Fig. 3, one context (J) is a classification on the attributes of another context (I). The relational composition $I \circ J$ means that the objects of the first context are classified with the attributes of the second context. This is an example of a context schema of type 5 from Table 1. In this case, the first context I contains types of living beings as objects and food sources as attributes. The second context J contains a classification of the food sources into food types. Using the dia matrix in the upper left corner has the effect that both lattices are glued together along the object-attribute concepts of the shared set. The lattice of J can be embedded join-preservingly into the combined lattice, which retains the attribute order of J. The lattice of I is meet-preservingly mapped and retains the object order of I. In the lattice diagram in Fig. 3, the concepts of I (except top and bottom) can be identified by their thicker node-borders. The concepts of J have been shaded in grey.

Wille proposed a different modelling of the relational composition[6] of two contexts, which represents a classification. Instead of the dia matrix, a context is used that represents the union of the object/attribute orders. Translated into RA notation, this context is $\begin{array}{c|c} (\overline{I^d} \circ I)^d \cup \overline{J} \circ J^d & J \\ \hline I & I \circ J \end{array}$. Fig. 4 shows this lattice and a dual/invers construction, which was not mentioned by Wille. The difference between the left-hand side lattice in Fig. 4 and the lattice in Fig. 3 is that Fig. 4 maintains the object and attribute orders of I and J. This usually means that the lattice has fewer concepts.

The dual construction on the right-hand side assumes that the relational composition uses an implicit ALL-quantifier. In this case, an object of context I only belongs to a class (i.e. an attribute in J), if *all* of its attributes in I are in that class. In this case, neither the object concepts, nor the attribute concepts can be mapped using order-preserving mappings. The object and attribute concepts are still highlighted in the lattice diagram as before. Together the lattices in Fig. 4 show the differences in the interpretations of the classifications. For example, although humans eat some instances of each of

[6] Rudolf Wille suggested this in a seminar presentation in 1995. I am not sure if this has ever been published.

	meat	French fries	milk	mice	gras	ANIMAL PRODUCT	VEGETARIAN FOOD	FAST FOOD	MEAT PRODUCT	PLANT-BASED
meat	×					×			×	
French fries		×					×	×		×
milk			×			×	×			
mice				×		×			×	
gras					×					×
human	×	×	×			×	×	×	×	×
vegetarian		×	×			×	×	×		×
cat	×		×	×		×	×		×	
eagle	×			×		×			×	
cow					×					×

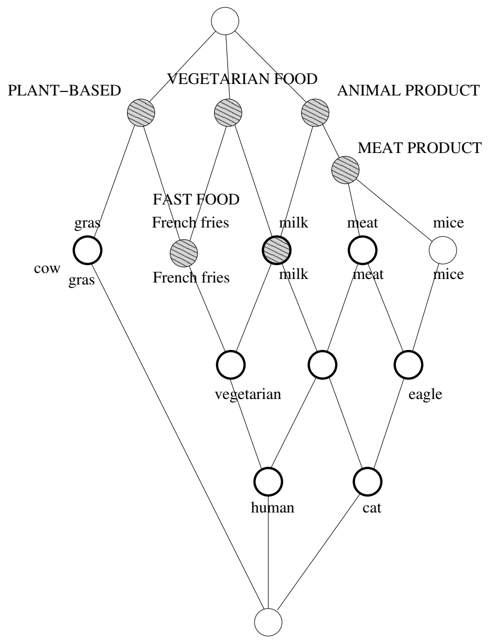


Fig. 3. An example of a context schema for relational composition: $\begin{array}{c|c} dia & J \\ \hline I & I \circ J \end{array}$

the different food types (left lattice), humans eat all instances of fast food and vegetarian food in this context, but not all instances of meat, plant-based and animal products (right lattice).

The relationship between the EXISTS- and the ALL-quantifier is also explored in the next example, but in a single lattice. Fig. 5 shows another example of classification. In this case, a formal context (C, C, I) represents a classification (bird, mammal, vertebrate and songbird). Another context contains some instances (G, C, J) of that classification (penguin, sparrow, kiwi and bat). The instances have properties (egg-laying, wings, flightless, flippers and nocturnal) represented by a third context (G, M, K). The properties are generalised to the classes using an ALL-quantifier. A class has a property if all of its instances have that property $(C, M, \overline{(\overline{J} \circ I)^d} \circ K)$. The resulting concept lattice in Fig. 5 shows the wider and narrow senses of each class. This is indicated by the dashed lines for the class BIRD. The concepts under the attribute concept of a class name show the wider sense of the class which contains any instances that have some of the class attributes. The concepts under the object concept of a class name show the narrower sense of the class which contains only instances which have properties common to all instances in the class. The prototypical instances of each class are given by $\overline{(\overline{J} \circ I)^d}$; in this case: the prototypical bird and songbird is sparrow; the prototypical mammal is bat. It is not claimed that this construction is novel (because it is similar to common FCA-models of Fuzzy and Rough Set Theory), but it is demonstrated how this can easily be expressed with RA operations.

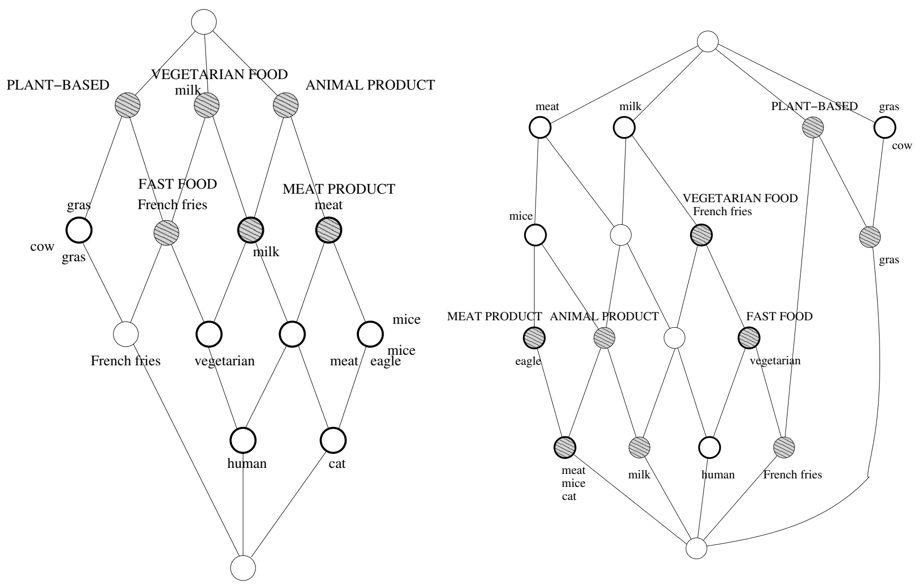


Fig. 4. Left-hand side: $\begin{array}{c|c} \overline{\overline{I}^d \circ I}^{d} \cup \overline{\overline{J} \circ J^d} & J \\ \hline I & I \circ J \end{array}$. Right-hand side: $\begin{array}{c|c} I^d \circ I \cap J \circ J^d & J \\ \hline I & \overline{I} \circ J \end{array}$

	BIRD	MAMMAL	VERTEBRATE	SONGBIRD	egg laying	wings	flightless	flippers	nocturnal
bird	×		×		×	×			
mammal		×	×			×			×
vertebrate			×						
songbird	×		×	×	×	×			
penguin	×		×		×		×	×	
sparrow	×		×	×	×	×			
kiwi	×		×		×	×	×		×
bat		×	×			×			×

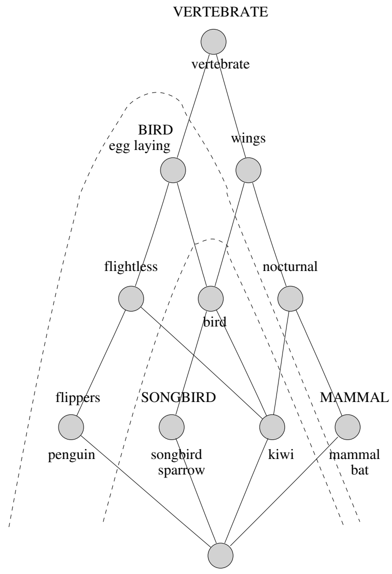


Fig. 5. The wider and narrower senses of a class, $\begin{array}{c|c} I & \overline{\overline{J} \circ I}^{d} \circ K \\ \hline J & K \end{array}$

In this example in Fig. 5, if each class has at least one instance that is distinguished from all other instances, then I is the attribute order of J, i.e. $I = \overline{(\overline{J^d} \circ J)^d}$. Turning this argument around, if the set of attributes in a formal context can be partitioned into a set of class C of type indicators and a set M of other attributes (i.e., the context is of the form $(G, C \cup M, J|K)$), then the narrower and wider senses can be calculated for these classes using $\begin{array}{c|c} \overline{(\overline{J^d} \circ J)^d} & \overline{\overline{\overline{J^d} \circ J} \circ \overline{J^d}} \circ K \\ \hline J & K \end{array}$.

5 RA for Modelling of Neighbourhood Lattices in Lexical Databases

Lexical databases are usually too large to be visualised as a single concept lattice. For example, a lexical database, such as WordNet or Roget's Thesaurus, contains more than 100,000 words. Therefore techniques are required that allow to extract smaller formal contexts from the lexical database. Priss & Old (2006) discuss the use of neighbourhood lattices, which use the plus operator to extract a neighbourhood around a word in Roget's Thesaurus. Starting with one word (or small set of words), all senses of this word are extracted, and then all other words that have the same senses. Priss & Old (2006) show how this process can be modelled with RA operations.

Simple neighbourhood lattices are not always the best way to represent the data because they can be too broad and include too many unrelated words. Simple neighbourhood lattices tend to include homographs which are words that have the same spelling, but are otherwise unrelated, such as "lead" as a verb and as a metal. A mechanism for excluding homographs is to use *restricted neighbourhood lattices*. These were invented by Old and shown by Priss & Old (2008) to be the most useful types of concept lattices for Roget's Thesaurus compared to other methods of reducing the complexity of concept lattices. Independently, a lattice-based model of Roget's Thesaurus was invented in the 1950's by Margaret Masterman and shown to be very similar to restricted neighbourhood lattices by Priss & Old (2009). Masterman's work precedes FCA and does not use formal contexts or concept lattices. Her algorithms are not mathematically formulated, but described semi-formally and with examples. Thus, the relationship between her work and FCA is not entirely obvious without careful analysis and was only recently discovered (Priss & Old, 2009). Nevertheless, the fact that Old and Masterman independently and starting from different theoretical backgrounds discover a similar method for modelling Roget's Thesaurus, seems to indicate that this method is appropriate for its domain.

Because Priss & Old (2006) describe neighbourhood lattices with RA, but not restricted ones, the RA modelling of restricted neighbourhood lattices is added here. For a context (G, M, I), neighbourhoods of objects are computed as $(I \circ I^d) \circ (I \circ I^d) \circ ...$, and for attributes as $(I^d \circ I) \circ (I^d \circ I) \circ ....$ Each additional $(I \circ I^d)$ corresponds to applying the plus operator two more times. The resulting matrices are symmetric. Thus the rows or columns of these matrices show which objects (or attributes) belong to the same neighbourhood at that stage. The plus operator is not a closure operator, but the transitive closure can be formed. The transitive closures $(I \circ I^d)^{trs}$ and $(I^d \circ I)^{trs}$ show the neighbourhood closures for the sets of objects and attributes, respectively. If all objects are connected to all attributes via transitive closure then both matrices are one. Otherwise, the matrices show equivalence relations on the objects and attributes. Priss & Old (2006) show that $(I \circ I^d)^{trs} \circ I = I \circ (I^d \circ I)^{trs}$. This is a matrix which contains the neighbourhood closures of the objects as columns and of the attributes as rows.

A restricted neighbourhood context of degree n means that instead of "at least one" as for the plus operator, an object must have at least n attributes: $G_1^{(I,\geq n)} := \{m \in M \mid gIm$ for at least n elements $g \in G_1\}$. For attributes: $M_1^{(I,\geq n)} := \{g \in G \mid gIm$ for at least n elements $m \in M_1\}$. We use the notation $\circ^{\geq n}$ to express that at least n elements need to be in common in the composition. Thus, $I \circ^{\geq 2} I^d$ shows objects that have at least two attributes in common. It is not possible to model this with just RA operations because this is equivalent to an FOL expression with more than 3 variables: $\exists_{a,b,x,y} : (a,b), (a,x), (b,x), (a,y), (b,y), x \neq y$. It is possible to express the condition that each row or column must have at least two 1s using RA: $((I \circ \overline{dia} \circ I^d) \cap dia) \circ I$ selects rows which have at least two 1s. $(((I \circ \overline{dia} \circ I^d) \cap dia) \circ I) \cap (I \circ ((I^d \circ \overline{dia} \circ I) \cap dia)) = (I \circ \overline{dia} \circ I^d) \circ I \circ (I^d \circ \overline{dia} \circ I)$ keeps only those 1s from the matrix I which belong to rows with at least two 1s and columns with at least two 1s. This is the form of a restricted neighbourhood context which has been shown to be useful for Roget's Thesaurus (Priss & Old, 2008).

6 FcaFlint

The FcaFlint software, which will be bundled with the next edition of FcaStone, implements all of the RA operations discussed in this paper. The operations are applied to a context stored in an input file (e.g. input.cxt) and the result is saved in a new file (e.g. output.cxt). The default format of the contexts is the Burmeister format, but in combination with FcaStone, any context format can be used that is supported by FcaStone. The RA operations are entered as functions. For example, $\overline{I} \circ I^d$ is executed from the command-line as:

```
fcaflint 'compos(invers(input.cxt),dual(input.cxt))' output.cxt
```

The matrices $one, dia, nul, \overline{dia}$ are entered as "<ONE>","<DIA>","<NUL>"and "<AID>". The dimensions of these matrices are automatically determined where possible. For example, in a union or intersection, one will use the same dimensions as the matrix it is unioned or intersected with. FcaFlint also provides the non-RA operations apposition, subposition, equality and transitive closure of composition. The $one, dia, nul, \overline{dia}$ matrices can be used for apposition and composition with other matrices, but not in combination with each other. This is because in that case, the dimensions of the matrices would be unknown. The 'equal()' function determines whether two matrices are equal. The 'trans()' function calculates the transitive closure of a matrix with respect to composition ($\circ$). The composition function also implements the (non-RA) operations of requiring at least n values to be shared in the comparison (written as $\circ^{\geq n}$ in the previous section).

Because context schemata are used frequently in FCA-based modelling, FcaFlint should be useful in many applications. FcaFlint has been tested on matrices of sizes of up to 50×400. It returns reasonably fast results, with the exception of the transitive closure function, which should only be used for smaller matrices. It should be stressed that FcaFlint is not aimed at end users (because using relation algebra requires some expertise) but is meant as an intermediate representation - as a formal language for conceptual modelling. In the future, it is intended to produce a graphical interface that allows users to enter certain natural language queries which are then translated internally into a relation algebra representation used for further processing.

7 Implementing RA Operations for FCA Contexts

According to Definition 1, square Boolean matrices form an RA, but FCA contexts are not usually square (i.e. have $G = M$.) Non-square matrices cannot form an RA because, for example, union and intersection require the matrices to have the same dimensions. Furthermore, the nul, one and dia matrices need to change their dimensions depending on which matrices they are used with. But it can be shown that if RA operations are applied to non-square matrices of appropriate dimensions then they fulfill the RA axioms. Thus, non-square Boolean matrices almost form an RA and the RA operations can be meaningfully used with such matrices.

FcaFlint intends to implement two modes for FCA contexts. In the first mode, only the matrices of the contexts are considered, not the sets of objects and attributes; and the RA operations are just matrix operations. FcaFlint only checks whether matrices have appropriate dimensions. If yes, the operations are executed. Otherwise a warning is printed. In this mode it is up to the user of FcaFlint to make sure that the RA operations are actually meaningful with respect to the formal objects and attributes.

The second mode for FcaFlint attempts to implement RA operations which consider objects and attributes as well as the matrices. Priss (2006) describes context-RAs and context algebraic structures (CAS) as a means for using RA with FCA. A context-RA assumes an *active domain* which is a set of all objects and attributes of all contexts of an application. Each context is then transformed into a square, $|\mathcal{A}|$-dimensional matrix. A context-RA is a matrix-RA where each row and column corresponds to an element of the active domain. This context-RA is only of theoretical interest because it is not practically useful to create $|\mathcal{A}|$-dimensional matrices. Apart from the size problems, such matrices would need to be recalculated each time new data is added to an application.

Priss (2006) therefore also describes CAS as a means of implementing RA operations for FCA contexts in a more efficient manner. For example, in order to calculate the union of two contexts their sets of objects and attributes are unioned (using a normal union, not a disjoint union as suggested by Ganter & Wille (1999)). The problem with this approach is that a CAS does not form an RA and, in fact, if the operations are used in combination with negation, the CAS operations may yield different results from the context-RA operations (which is undesirable).

The problem with the CAS operations is that because they enlarge contexts as needed, rows and columns may need to be added to a context. These rows and columns are usually filled with 0s, but if a context was previously negated once, they should be filled with 1s. The idea for FcaFlint is to consider both the *inside* of a formal context (which consists of the relation between objects and attributes that is currently defined for the context) and the *outside* of the context (which collects conditions about potential elements from the active domain that might be added to the context at a later stage). Only the inside is stored as a matrix. The outside is stored as a set of conditions (e.g., “all 0”, “all 1”) without having a complete list of which elements belong to the outside. This is much more efficient than using context-RAs and allows to add new elements to an active domain over time.

All contexts start out with their outside containing all 0s. If the context is once negated, the outside contains all 1s. For union and intersection, both inside and outside conditions contribute to the new inside of the context. It can be shown that the axioms of a Boolean algebra are fulfilled by union, intersection, negation, one, and nul applied to formal contexts as defined for CAS and by additionally keeping track of whether the outside of a context is filled with all 0s or all 1s.

Unfortunately, composition can change the outside of a context into conditions which are more complicated than “all 1” or “all 0”. Thus, it is still not easy to create an RA in this manner. But the complexity of these conditions increases slowly. For many applications, the outside conditions of the contexts will be simple or irrelevant. For now, the approach that is taken with FcaFlint is to store simple conditions and to stop the program with a warning if the conditions are getting too complex. A warning would tell users that they need to manually check the sets of objects and attributes of their formal contexts and to verify whether the CAS operations that they are attempting to use are actually meaningful.

8 Conclusion

In summary, this paper presents a discussion of the use of RA operations on formal contexts. The paper was motivated by the development of the FcaFlint software which allows to apply RA applications directly to formal contexts stored in a variety of formats. With FcaFlint, RA operations can be used for conceptual modelling in the same way as RLA operations can be used using the Tupleware software[7]. But the implementation of FcaFlint highlighted some problems with previous models of RA/FCA that needed to be overcome. This paper also provides some insights into the expressivity of RA for formal contexts and discusses examples in the area of classification and lexical databases. A more systematic evaluation of RA operations on formal contexts might be of interest, but this is left for future research.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Heidelberg (1999)
2. Kim, K.H.: Boolean Matrix Theory and Applications. Marcel Dekker Inc., New York (1982)
3. Priss, U.: An FCA interpretation of Relation Algebra. In: Missaoui, R., Schmidt, J. (eds.) ICFCA 2006. LNCS, vol. 3874, pp. 248–263. Springer, Heidelberg (2006)

[7] `http://tockit.sourceforge.net/`

4. Priss, U., Old, L.J.: An application of relation algebra to lexical databases. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 388–400. Springer, Heidelberg (2006)
5. Priss, U., Old, L.J.: Data Weeding Techniques Applied to Rogets Thesaurus. In: Knowledge Processing in Practice (in publication) (2008)
6. Priss, U., Old, L.J.: Revisiting the Potentialities of a Mechanical Thesaurus. In: Ferre, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 284–298. Springer, Heidelberg (2009)

Conceptual Graphs and Datatypes

Thomas Raimbault, David Genest, and Stéphane Loiseau

University of Angers - LERIA laboratory of Computer Science
2 boulevard Lavoisier 49045 Angers Cedex 01 - France
{thomas.raimbault,david.genest,stephane.loiseau}@info.univ-angers.fr

Abstract. *Datatypes*, like numbers or strings, are widely used in Knowledge Representation (*e.g.* in RDF(S)/OWL or UML languages). The usual model of simple conceptual graphs does not support datatypes. Some extensions of conceptual graphs have been proposed for using datatypes, however these extensions often wander from initial model of conceptual graphs by introducing for instance procedural relations between nodes. This paper proposes a datatype extension for the simple conceptual graph model. Our contribution is threefold. First, we allow the use of datatypes for typing concept nodes. Second, we define two families of conceptual graphs: *factual graphs* and *query graphs*, both close to initial model. Factual graph is used to represent factual knowledge, including values of datatypes. Query graph may contain concept nodes that represent conditional queries on values of datatypes; these conditions are expressed by regular operators on datatypes. Third, we adapt projection to operate from a query graph to a factual graph.

1 Introduction

The notion of *datatypes*, like numbers, dates or strings, is widely used in Knowledge Representation, especially in semantic web languages such as RDF/RDFS [1,2] and OWL [3] or in UML [4], an object oriented modeling language. However, the model of *simple conceptual graphs* (SG model), which is a formalization by Chein & Mugnier [5,6,7] of Sowa's conceptual graph model [8], does not support datatypes. The conceptual graphs' inability to model datatypes and their values is mainly due to the fact that individual markers are not suitable to represent values of datatypes and that projection operation is not fitting for operating on such values, like for querying data. Indeed, to extend SG model it is necessary to use specific operators to properly operate on values of datatypes, *e.g.* to answer the following query: "Who are people over 18 years old?".

Our contribution is threefold. Firstly, we extend SG model in order to represent datatypes into the support (*i.e.* into ontological level of modeled knowledge) and to represent values of datatypes into conceptual graphs. In other words, we allow the possibility of a concept node to have a datatype as type. Secondly, we define two families of conceptual graphs: *factual graphs* and *query graphs*. In both cases, values of datatypes can be represented. While a factual graph represents known facts of modeled knowledge, a query graph allows the opportunity to represent a data by a set of values - of the same datatype - with special operators, called *comparison operators*. For example, all the values over 18 can be represented using the greater-than operator ($>$). Thirdly, we adapt projection operation so that it operates from a query graph to a factual graph and it takes into account values of datatypes.

Some extensions of conceptual graph model (especially SG model) have been proposed in the literature for using datatypes to type some nodes, such as the use of procedural relations between nodes in [9] extended in [10]. Our approach is different in two main points from extension in [10], which it is wandered from initial SG model. First, a conceptual graph in [10] includes four families of nodes (concept nodes, relation nodes, operator nodes and data nodes), while we maintain the bipartite composition of a conceptual graph. Second, [10] systematically allows representation of data by a set of values that expresses a given condition, for instance the existence of a number under than 20. We chose not to allow conditional values into a conceptual graph that models factual knowledge of a given modeling. We only allow it into a conceptual graph that models a query, where such conditions on values are still represented in a bipartite graph. Otherwise, what would be for example the answer to the query "What are people over 18 years old?" on some knowledge that contains the following information: "Is There a person whose age is under 20 years old"? Thus, our adaptation of projection operation is close to initial SG model. To keep interest of projection we adapt too the definition of normal form of a graph, and we introduce a special transformation of a factual graph called *datatypes specialization form.*

This paper is organized as follows. Section 2 presents our extension of the definition of support in order to represent datatypes and the definition of factual graphs in order to represent values of datatypes. Section 3 presents query graphs and a comparison operator set. Section 4 presents an adaptation of projection operation, an adaptation of normal form and the definition of datatypes specialization form.

2 Datatype Extension

2.1 Some Notes on Simple Graphs

In this article, we consider the so-called model of *simple conceptuals graphs* (SGs) [5,6,7]. To simplify reading, fundamental definitions that constitute SG model are reported in Appendix (Definitions 10 to 15).

We recall here the main lines of *conjunctive types* that are used in SG model and that may be less-known.

In *basic conceptual graph model* (see in [7]), to express the fact that a concept node has several types $t_1, t_2, \ldots, t_n$ one needs to represent and use a concept type that is the subtype of all types $t_1, t_2, \ldots, t_n$. Giving in extension all acceptable types is not conceivable in practice, therefore [6] defines conjunctive types.

The set of conjunctive types, noted $T^{\sqcap}$, is defined in intension by a hierarchy of primitive concept types (*primitive* adjective is used to refer "standard" concept types with SGs). Any subset of incomparable primitive concept types defines a conjunctive type (Definition 11). However not all conjunctions of types have a meaning. Thus unexpected conjunctive type is *banned*, i.e. the primitive types of the banned conjunction cannot have a common subtype. If B refers the set of defined banned types, $\downarrow B$ denotes the set of all banned types (Definition 12). Note that a primitive type can be seen as a conjunctive type with one type. Conjunctive types are provided with a natural partial order, which extends the one on primitive types (Definition 13): $t \leq s$, where $t = \{t_1, \ldots, t_n\}$ and $s = \{s_1, \ldots, s_p\}$, if for all primitive types $s_j \in s$ there is a primitive type $t_i \in t$ such as $t_i \leq s_j$.

Concerning normal form of a SG, if an individual is represented by a node of type $t = \{t_1, \ldots, t_n\}$ and by a node of type $t' = \{t'_1, \ldots, t'_p\}$ then the individual is exactly represented by a node of conjunctive type $t \cup t'$. For example, a conceptual graph that has the two following concepts nodes [Student: Mike] and [Professor: Mike] may express the fact that student Mike is also a professor (*e.g.* a PhD student who has a teaching assistant status). Thus, individual Mike is represented, into the conceptual graph in normal form, by the single concept node [Professor,Student: Mike] that is typed by the conjunctive concept type {Professor,Student}, *i.e.* by both Professor and Student. Note that if a conjunctive type is composed by two comparable primitive types, only the most specialized type is kept.

2.2 Set of Literals and Extension of Support

Let us recall that in SG model, an individual marker is a reference and only a reference for identifying a given entity. Thus the set of individual markers is not suitable to represent values of datatypes. Even if individual markers are usually written as strings, one of these strings do not have the meanning of "a string" (as a value of datatype *string*): they are comparable to URI (in semantic web) or names of constants. For this reason, we keep unchanged the set of individuals but we introduce a new set representing literals. This literal set is defined in intension, unlike individual markers that are given in extension.

Definition 1 (Set of literals). *Let Σ be all printable characters (including blank) of the Unicode standard*[1]*. A* literal *is a finite sequence, possibly empty, of characters; the literal formed by characters* a*,* b *and* c *is noted* abc*. The* set of literals *that can be formed from Σ is denoted by Σ^*.*

We now extend the definition of support from SG model. The concept type set is adapted and we introduce a function for typing literals *i.e.* it associates a (conjunctive) concept type for each literal.

Definition 2 (Extended support). *An* extended support *is a 7-uplet $S = (T_C, B, T_R, \sigma, I, \tau, \mu)$, as T_C, B, T_R, σ, I and τ are defined in Definitions 10, 11 et 12, and where:*

[1] Unicode specifies a set of characters independent of any coding, `http://www.unicode.org/`. The choice of Unicode is only to explicitly designate a set of characters, and thus to formally define Σ^*.

– *T_C is structured as follows:* Thing *and* Literal *are the only two direct subtypes of $\top$, with* {Thing,Literal} *a banned conjunctive type. Both of these concept types form the skeleton of every set of primitive concept types (Figure 1).*
– *$I \cap \Sigma^* = \emptyset$, with Σ^* the set of literals (see Definition 1).*
– *the type t associated by τ with an individual marker m is such as $t \leq$* Thing.
– *μ is a function, called* typing literals function*, such as $\mu : l \in \Sigma^* \rightarrow t \in T^{\sqcap} \mid t \leq$* Literal
– *$\#$ is a* generic value*, as non-valued value. $*$ and $\#$ are not comparable.*

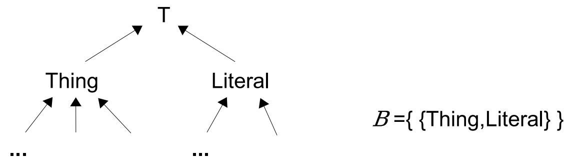


Fig. 1. Skeleton of every support

The constraint stipulating that only Thing and Literal are direct subtypes of $\top$ enables to explicitly separate concept types corresponding to *datatypes* (subtypes of Literal) from concept types corresponding to *classes* (subtypes of Thing). These two sets of concept types are obviously disjoint: the conjunctive concept type {Thing,Literal} is banned. So for a given modeling, user will be prompted to create its own concept types, either subtypes of Thing or subtypes of Literal.

The typing literals function is necessary because it is not conceivable in practice to define by extension all values of a datatype; corresponding datatype(s) of a value is thus given by intension, contrary to what is done by τ concerning individual markers. It is important to note that μ is a function of our extented SG model, but the designer of a given modeling is assigned to define μ according to this modeling. For instance, a first typing literals function μ_1 may associate literal 42 with conjunctive type {integer,string} and may associate literal foo with type string, whereas a second typing literals function μ_2 (of another modeling) may associate literal 42 with type integer and literal "foo" with type string.

Example 1. Figure 2 shows an example of extended support. Concept types Person, Professor, Student and Lesson represent some *classes*. So, they are defined as subtypes of Thing. Concept types string, real and int, which are subtypes of Literal,

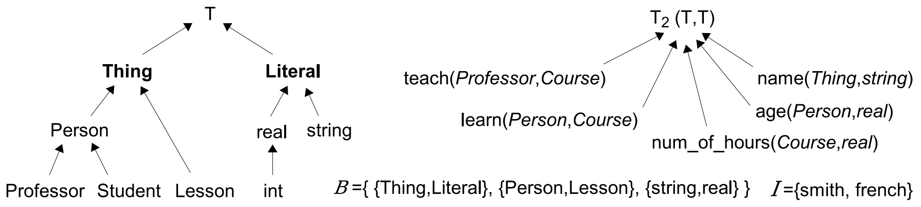


Fig. 2. Example of extended support

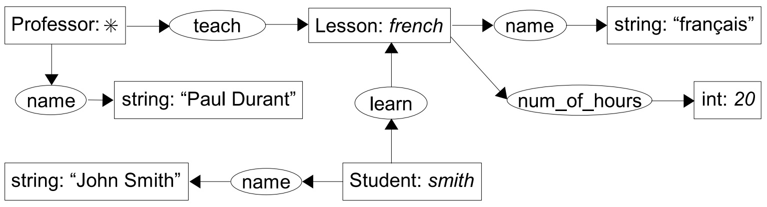


Fig. 3. Example of *factual graph*, based on extended support in Figure 2

represent some datatypes. Both datatypes and classes may be hierarchically organized with a *kind-of* relation, as with SG model. For instance, datatype int is a kind-of datatype real, Student is a specialization of Person. Some relation types are given between two classes, like teach between Professor and Lesson, or between a class and a datatype, like age between Person and real. In addition to the banned concept type {Thing,Literal}, which is defined into skeleton's extended support, other banned concept types are defined here: {Person,Lesson} and {string,real}.

2.3 Factual Graph

A factual graph (FG) represents factual knowledge of a given modeling. It is based on an extended support that represents ontological knowledge.

Definition 3 (Factual graph). *A* factual graph $F = (C, R, U, lab)$ *based on an extended support* $S = (T_C, B, T_R, \sigma, I, \tau, \mu)$ *is a labelled bipartite graph as it is defined in Definition 14, and where:*

- *lab satisfies:*
  - $\forall c \in C,\ lab(c) \in T^{\sqcap} \setminus \{\top\} \times (I \cup \{*\} \cup \Sigma^* \cup \{\#\})$. *Label of a concept node c is a pair (type(c), ref(c)). A concept node c such as*[2] *ref(c)* $\in \Sigma^*$ *is called a* valued data concept node, *if ref(c)* $= \#$ *then c is called an* unvalued data concept node.
- *Moreover, lab respects:*
  - *constraints fixed by* μ*:*
    - ∘ $\forall c \in C$ *if* $ref(c) \in \Sigma^*$ *then* $\mu(ref(c)) \leq type(c)$.

This definition differs from the one of SG model, because the label of a concept node in a factual graph can be made up of a marker (individual or generic) or of a literal (possibly generic).

Example 2. Figure 3 presents an example of factual graph, which is based on extended support that is defined in Figure 2. Let us see in particular the presence of three valued data concept nodes of type string where values are the strings "John Smith", "Paul Durant" and "français", and the data concept node of type int where value is integer 20. This factual graph expresses following knowledge: "A student (ID *smith*) named "John Smith" is taking part in lecture named "français" of 20 hours long, identified by *french* and given by a professor named "Paul Durant"."

[2] From Definition 14, one recalls that if ref(c) $\in I$ then c is said to be an *individual concept node*, and if ref(c) $= *$ then c is called a *generic concept node*.

3 Query Graph

We define here a second family of conceptual graphs (in addition to factual graphs): query graphs (QGs). Into a query graph, a label of concept node has three parts such as one is a *comparison operator*. Comparison operators are used to express some conditions on values.

Definition 4 (Query graph). *A* query graph $Q = (C, R, U, lab)$ *based on an extended support* $S = (T_C, B, T_R, \sigma, I, \tau, \mu)$ *is a labelled bipartite graph as it is defined in Definition 3, and where:*

- *lab satisfies:*
  - $\forall c \in C,\ lab(c) \in T^{\sqcap} \setminus \{\top\} \times Op \times (I \cup \{*\} \cup \Sigma^* \cup \{\#\})$. *Label of a concept node c is a* ***triplet*** *(type(c), operator(c), ref(c)).*
    Op is a set of binary operators, called comparison operators *(see for instance Table 1). Each operator has a domain of definition, given by the function domain* $: o \in Op \rightarrow t \in T^{\sqcap}$. $\forall c \in C,\ type(c) \leq domain(operator(c))$.

In Table 1, the three first comparison operators exist in every case; the user/designer of a given modeling is assigned to define other operators that are needed to express queries on this modeling. Domain of definition of an operator is obviously the greatest as possible (primitive or conjunctive) concept type that can be used with this operator. For example, the domain definition of operator $<$ is the datatype real rather than int, thus $<$ may be used between reals or integers (int is a subtype of real) and not just between integers.

It is important to note in Table 1 that operator $\preceq$ and $<=$ have nothing in common, since set of markers and set of literals are disjoint. Indeed, $\preceq$ is used to test order between two (individual or generic) markers whereas $<=$ is used to test order between real numbers.

Example 3. Figure 4 shows an example of a query graph, which is based on the extended support that is defined in Figure 2. This graph expresses the following query: "What is the name of the lesson, more than 10 hours long, that is learned by the person whose name (not necessarily complete) is "Smith"?". The valued data concept node [int: {>} 10] represents an integer upper than 10. The valued data concept node [string: {like} "Smith"] queries the existence of a string containing the substring "Smith". The unvalued data concept node [string: {eq} #] represents any string, *i.e.* any value of datatype string.

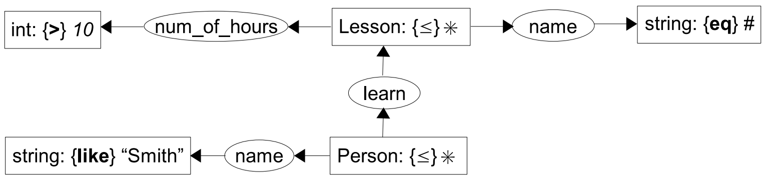


Fig. 4. Example of *query graph*, based on extended support in Figure 2

Table 1. A set of *comparison operators* that may compose labels of (valued or unvalued) data concept nodes into query graphs

(binary) Operator	Domain of definition	comments
$\preceq$	$I \cup \{*\}$	true iff individual makers are the same or if left operand is the generic marker $*$.
$=$	$\Sigma^* \cup \{\#\}$	true iff there is strictly equality between literals or if left operand is the generic value $\#$.
$<>$	$\Sigma^* \cup \{\#\}$	true iff there is not equality between literals or if left operand is $\#$.
$==$	real $\cup \{\#\}$	true iff there is numerical equality between two operands or if left operand is $\#$.
$!=$	real $\cup \{\#\}$	true iff there is not numerical equality between two operands or if left operand is $\#$.
$<$	real $\cup \{\#\}$	true iff left operand is strictly lower than right operand or if left operand is $\#$.
$<=$	real $\cup \{\#\}$	true iff left operand is lower or equal than right operand or if left operand is $\#$.
$>$	real $\cup \{\#\}$	true iff left operand is strictly upper than right operand or if left operand is $\#$.
$>=$	real $\cup \{\#\}$	true iff left operand is upper or equal than right operand or if left operand is $\#$.
eq	string $\cup \{\#\}$	true iff there is strictly equality between two operands or if left operand is $\#$.
neq	string $\cup \{\#\}$	true iff there is not equality between two operands or if left operand is $\#$.
ll	string $\cup \{\#\}$	true iff left operand is strictly lower (as alphanumeric comparison) than right operand or if left operand is $\#$.
leq	string $\cup \{\#\}$	true iff left operand is lower or equal (as alphanumeric comparison) than right operand or if left operand is $\#$.
gg	string $\cup \{\#\}$	true iff left operand is strictly upper (as alphanumeric comparison) than right operand or if left operand is $\#$.
geq	string $\cup \{\#\}$	true iff left operand is upper or equal (as alphanumeric comparison) than right operand or if left operand is $\#$.
like	string $\cup \{\#\}$	true iff left operand is a strictly substring of right operand or if left operand is $\#$.
cil	string $\cup \{\#\}$	true iff left operand is a substring (as case insensitive comparison) of right operand or if left operand is $\#$.

4 Operations

4.1 Projection

In our extension of SG model, the projection operation requires a minor adaptation due to the fact that projection is now applied from a query graph to a factual graph. Indeed, criteria of comparison of projection between two concept nodes

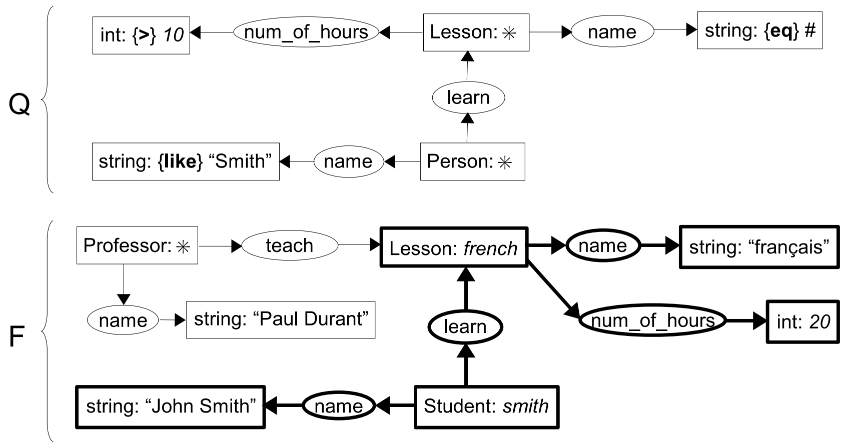


Fig. 5. Factual graph F (from Figure 3), query graph Q (from Figure 4), and in bold image of Q in F by application of our adapted projection operation from Q to F

(one from query and one from fact) needs a revision to conclude if a concept node from fact may be an image by projection of a concept node from query.

Definition 5 (Compatibility on labels). *Let $Q = (C_Q, R_Q, U_Q, lab_Q)$ be a query graph and $F = (C_F, R_F, U_F, lab_F)$ be a factual graph, both based on a same extended support $S = (T_C, T_R, \sigma, I, \tau, \mu)$. Let $e = (t, \nabla, m)$ and $e' = (t', m')$ be labels of two concepts nodes where $c \in C_Q$ and $c' \in C_F$ with $lab_Q(c) = e$ and $lab_F(c') = e'$.*

Label e is said to be compatible *with e' if and only if $t' \leq t$ and $m' \nabla m$ is true.*

Definition 6 ((adapted) Projection). *A* projection *from a query graph $Q = (C_Q, R_Q, U_Q, lab_Q)$* to a factual graph *$F = (C_F, R_F, U_F,\ lab_F)$, both based on a same extended support $S = (T_C, T_R, \sigma, I, \tau, \mu)$, is a pair of mappings $\Pi = (f, g)$, such as it is defined in Definition 15, and where: the* compatibility *used between two concept nodes is the one defined in Definition 5.*

Here, comparison between two concept nodes, the first from a query graph and the second from a factual graph, considers what is done in Definition 5 and not just $\leq$ such as it is used in Definition 15 (we recall that in SG model, $\leq$ is used to compare two labels of two concept nodes).

Example 4. In Figure 5, the answer of query graph Q on factual graph F is the bold-marked subgraph of F. This only answer, given by projection[3] from Q to F, expresses that "The lesson named "français" is 20 hours long and is learned by student "John Smith"". This answer is the image by the only projection from Q to F. In particular the image of valued data concept node [string: {*like*} "Smith"] is the valued data concept node [string: "John Smith"], the image of [int: {>}10] is

[3] The projection operation to consider is obviously the one defined in Definition 6.

[int: 20], and the image of unvalued data concept node [string: $\{eq\}$ #] is [string: "français"].

Let us note that operator of an individual or a generic concept node of a query is necessarily the operator $\preceq$, because each operand is either an individual marker or the generic marker. So, operators of such concept nodes may be omitted within nodes' representations, as it is shown in query graph Q in Figure 5 (that is the same as the query graph in Figure 4). For instance, individual concept node [Person: $*$] is equivalent to [Person: $\{\preceq\}$ $*$].

In the same way, if the operator of a (valued or unvalued) data concept node is omitted that is a shortcut to express the operator $=$.

4.2 Normalization

Normal form of a FG remains almost the same as a SG, literals are considered in addition. A factual graph is said to be in *normal form* if an individual marker or a literal are used into one and only one concept node.

Definition 7 (Normal form of factual graph). *Let $F = (C_F, R_F, U_F, lab_F)$ be a factual graph, based on an extended support $S = (T_C, B, T_R, \sigma, I, \tau, \mu)$. The* normal form *of F, denoted by $\mathrm{nf}(F)$, is obtained as follows: for any set $\{c_1, \ldots, c_k\}$ / $c_i \in C_F$, $ref(c_i) = x$, $1 \leq j \leq k$ and $x \in I$ (respectively $x \in \Sigma^*$), all c_i are merged into one individual concept node c (respectively into one valued data concept node c) with $type(c) = type(c_1) \cup \ldots \cup type(c_k)$ and $ref(c) = x$.*

Example 5. Let us consider in Figure 6 query graph Q_1 to interrogate factual graphs F_1 and F_2. F_1 represents the following knowledge: "mary and bob have the same age that is 20"; F_2 represents the fact: "mary is 20 years old and bob is 20 years old". Q_1 expresses the query: "Are there two people who are the same age, which is greater than (>=) 18?".

Q_1 to interrogate F_1 has an answer, which is F_1 as whole, whereas Q_1 to interrogate F_2 has no answer through an answer is expected. This example presents the interest of using the normal form of a factual graph. Namely, Q_1 has an answer to interrogate F_2 where F_2 is $\mathrm{nf}(F_1)$.

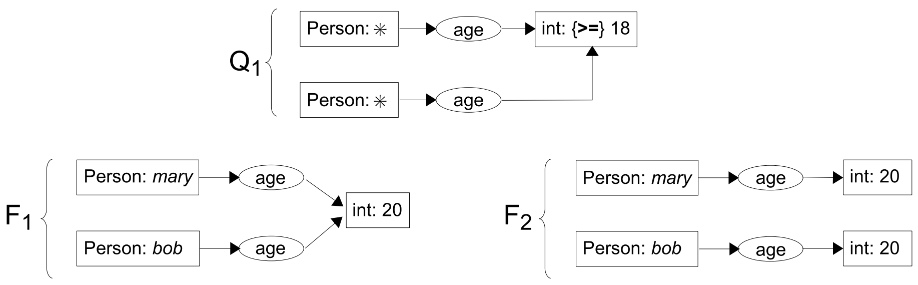


Fig. 6. Interest of a factual graph to be in *normal form*

4.3 Datatype Specialization Form

We recall that the typing literals function μ of an extended support (see Definition 2) is a function that associates a concept type for each literal; μ depends on defined datatypes, *i.e.* depends on a given modeling. For example, literal 18 may be associated by a given function μ_1 with conjunctive concept type {real,int,string}, or just type {int,string} due to int $\leq$ real (see Definition 11).

We define here the *value-datatypes mapping operator* (VDM operator). For a valued data concept node $[t : v]$ into a factual graph, VDM operator creates a conjunctive concept type composed by primitive concept types (as datatypes) from μ w.r.t. v that are subtypes of t (t included).

Definition 8 (Value-datatypes mapping operator). *Let c be a valued data concept node of a factual graph, where $t = type(c)$ and $v = ref(c)$. The* value-datatypes mapping operator *returns a conjunctive concept type in accordance with $ref(c)$, noted* vdm(c)*, such as: $vdm(c) = \mu(v) \cap \{t' \mid t' \leq t\}$.*

For example, the conjunctive concept type that is returns by VDM operator from the valued data concept node [real: 18] is {int}. Indeed, literal 18 is associated by μ_1 with conjunctive concept type {int,string}, and according to the support int $\leq$ real and string is incomparable with real.

We now define *datatype specialization form* of a factual graph. This special form is motivated to compensate for the fact that given some comparable datatypes (see in Definition 2 order among concept types subtypes of Literal), values of such datatypes may not be comparable by comparison operators whereas they have to be; e.g. see example 6.

Definition 9 (Datatypes specialization form). *A factual graph F is said to be in* datatype specialization form *if each valued concept node c of F is such as $type(c) = vdm(c)$.*

The datatypes specialization form of a factual graph F, denoted dsf(F)*, is obtained as follows: for any valued concept node c of F, $type(c)$ is remplaced with $vdm(c)$.*

Example 6. Let us consider in Figure 7 the query graph Q_2 and the two factual graphs F_3 and F_4. Q_2 expresses the query “Is There a person who is older than 18 (as an integer)?”. F_3 represents the knowledge “The person mary is 20 (as a real number) years old”. F_4 represents the knowledge “Person mary is 20 (as a real) and bob is 20 (as an integer)”.

It is interesting to see that Q_2 to interrogate F_3 has no answer, whereas if knowledge of F_3 are completed such as that is represented in F_4 then Q_2 to interrogate nf(F_4) (we have discussed in the previous section the interest of normal form) has two answers including “mary is 20 years old”. So, datatypes specialization form of F_3 has to be used to settle problem of this unwanted lack of answer: Q_2 to interrogate dsf(F_3) has the expected answer “mary is a person who is 20 (number equally represented as a real or as an int) years old”.

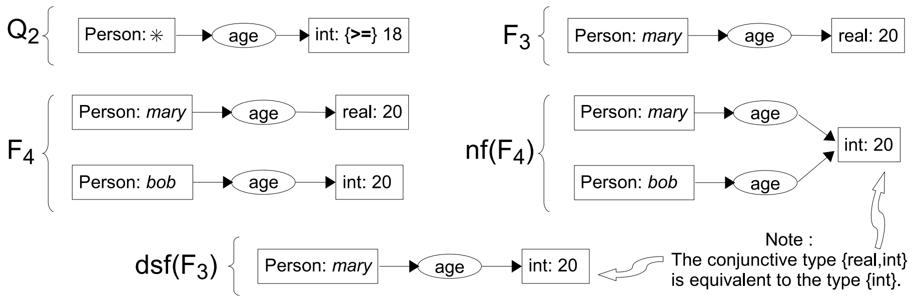


Fig. 7. Interest of a factual graph to be in *datatypes specialization form*

5 Conclusion

We have proposed an extension of SG model. First, our approach takes into account datatypes and their values into a model close to initial SG model. Two families of graphs, factual graphs and query graphs, were defined for different use cases: for representing factual knowledge or for expressing a query. Second, the complexity of our adaptation of projection operation remains the same as the one in SG model. Indeed, all algorithms of usual projection can be used for our model. The only change is the compatibility for valued data concept nodes. Moreover, the cost of comparison operator does not depend on size of graphs. So, the extra cost of projection's execution is negligible. Third, our approach is easy to implement on top of a given conceptual graph tool; we have implemented a prototype on top of Cogitant [11].

We are now working on the possibility of using rules [12] and constraints [13] with our SG extended model. Namely, definitions of a rule and of a positive or negative constraint may be adapted as follows. For a rule, the hypothesis may be a query graph while the conclusion may be a factual graph. For a constraint, the condition, the obligation (of a positive constraint) and the prohibition (of a negative constraint) may be query graphs. Applying a rule or checking a constraint on a factual graph will be done by using our adaptation of the projection operation.

References

1. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C Recommendation (2004), http://www.w3.org/TR/rdf-concepts/
2. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C Recommendation (2004), http://www.w3.org/TR/rdf-schema/
3. Dean, M., Connolly, D., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Technical report, W3C Recommendation (2004), http://www.w3.org/TR/owl-ref/

4. Booch, G., Jacobson, C., Rumbaugh, J.: The Unified Modeling Language - a reference manual. Addison-Wesley, Reading (1998)
5. Mugnier, M.L.: Knowledge Representation and Reasoning based on Graph Homomorphism. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 172–192. Springer, Heidelberg (2000)
6. Chein, M., Mugnier, M.L.: Concept types and coreference in simple conceptual graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 303–318. Springer, Heidelberg (2004)
7. Mugnier, M.L., Chein, M.: Graph-based Knowledge Representation. In: Advanced Information and Knowledge Processing. Springer, London (2009)
8. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
9. Lukose, D., Mineau, G.W.: A comparative study of dynamic conceptual graphs. In: Proc. of KAW 1998 (1998)
10. Baget, J.F.: A Datatype Extension for Simple Conceptual Graphs and Conceptual Graphs Rules. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 83–96. Springer, Heidelberg (2007)
11. Genest, D.: CoGITaNT 5.1.93 - Reference Manual (2008), http://cogitant.sourceforge.net
12. Salvat, É.: Theorem Proving Using Graph Operations in the Conceptual Graph Formalism. In: Proc. of ECAI 1998, Brighton, UK (1998)
13. Baget, J.F., Mugnier, M.L.: Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints. JAIR 16(12), 425–465 (2002)

Appendix

Following definitions are fundamental definitions of SG model [5,6,7,8].

Definition 10 (Support [5,6,7]). *A* support *of conceptual graphs is a 6-uplet* $S = (T_C, B, T_R, \sigma, I, \tau)$*, where:*

- T_C *is the partial ordered set of primitive concept types, whose greatest element is* $\top$ *(the universal type).*
- B *is the set of banned primitive concept types (see Definition 12).*
- T_R *is the set of relation types, with a partition:* $T_R = T_{R_{i_1}} \cup \cdots \cup T_{R_{i_p}}$ *where* $T_{R_{i_j}}$ *is a set of* i_j*-ary relation types,* $i_j > 0$*. Each* $T_{R_{i_j}}$ *is partially ordered and has a greatest element* $\top_{i_j}$*. The partial orders on* $T^{\sqcap}$ *(see Definition 13) and on* T_R*, noted* $\leq$*, correspond to a* kind-of *relation between types.*
- σ *is a mapping, which associates a signature with each relation type; the signature of a relation specifies the arity and greatest possible (primitive or conjunctive) concept type for each argument. More precisely, for any* $t_r \in T_{R_{i_j}}$*,* σ *associate an uplet* $\sigma(t_r) \in (T^{\sqcap})^{i_j}$ *and verifies: for all* t_{r_1} *and* t_{r_2} *of* $T_{R_{i_j}}$*, if* $t_{r_1} \leq t_{r_2}$ *then* $\sigma(t_{r_1}) \leq \sigma(t_{r_2})$ *where order on signature is the product order* [4] *on* $(T^{\sqcap})^{i_j}$*. The* i^{th} *argument of* $\sigma(t_r)$ *is noted* $\sigma_i(t_r)$*.*

[4] Namely, the type that is associated with the k^{th} argument of t_{r_1} is lower or equal than the type that is associated with the k^{th} argument of t_{r_2}.

– *I is the set of individual markers. In addition there is a generic marker, noted $*$, as non-specified individual. $I \in \{*\}$ is provided with an order, such that $*$ is greater than every individual marker, and individual markers are pairwise non-comparable.*
– *τ is a mapping from I to T_C, called* conformity mapping*, which for each individual marker associates a concept type.*

Definition 11 (Set of conjunctive concept types [5]). *A* conjunctive concept type *is given by a (non-empty) set of incomparable primitive concept types $\{t_1, \ldots, t_n\}$. This type is denoted by the set itself or by $t_1 \wedge \ldots \wedge t_n$; i.e. an antichain of the set of primitive types.*

Let T_C be a set of primitive concept types. $T^{\sqcap}$ denotes the set of all conjunctive concept types *over T_C . It is provided with the following partial order, which extends the partial order on T_C: given two types $t = \{t_1, \ldots, t_n\}$ and $s = \{s_1, \ldots, s_p\}$, $t \leq s$ if for every primitive type $s_j \in s$, $1 \leq j \leq p$, there exists a primitive type $t_i \in t$, $1 \leq i \leq n$, such that $t_i \leq s_j$.*

The set of acceptable conjunctive types can be exponentially bigger than the primitive type set. That is why the concept type hierarchy is not defined in extension but in intension of the primitive type set and a set of assertions stating which types are banned. The set of acceptable types is obtained from $T^{\sqcap}$ by removing the banned types ($\downarrow B$).

Definition 12 (Banned type set [5]). *Let B denote a subset of non comparable conjunctive types. An element of $T^{\sqcap}$ is said to be banned w.r.t. B if it is less or equal to an element of B.*
$\downarrow B$ denotes the set of all banned types*; i.e. $\downarrow B = \{t \in T^{\sqcap} \mid \exists t' \in B, t \leq t'\}$.*

Definition 13 (Concept type hierarchy [5]). *A concept type hierarchy T is given by a couple (T_C, B) where:*
– *T_C is the set of primitive concept types,*
– *B is the set of basic banned conjunctive types,*
– *$\forall b \in B, \nexists t \in T_C \mid t \leq b$; i.e. $\downarrow B \cap T_C = \emptyset$.*

T is defined as the set[5] $T^{\sqcap} \setminus \downarrow B$. $T^{\sqcap}$ is thus partitioned into the acceptable types T and the banned types $\downarrow B$.

Using conjunctive types, the label of a concept node is a pair (type(c),ref(c)) where type(c) is in the conjunctive concept type set (rather than the primitive concept type set). type(c) is represented by a list of primitive types, which are separated by a comma.

Definition 14 (Conceptual graph [5,6,7]). *A* conceptual graph *$G = (C, R, U, lab)$, based on a support $S = (T_C, B, T_R, \sigma, I, \tau)$, is a finite, undirected and bipartite multigraph where:*
– *C is the set of concept nodes, R is the set of relation nodes. C and R are disjoint ($C \cap R = \emptyset$).*

[5] A primitive type can be seen as a conjunctive type with one type.

- *U is the set of edges.*
- *lab is a mapping that associates a label for every nodes and every edges of G, such as:*
  - $\forall r \in R,\ lab(r) \in T_R$
  - $\forall c \in C,\ lab(c) \in T^{\sqcap} \times (I \cup \{*\})$. *A concept node c is labelled by a pair (type(c), ref(c)). A concept node c where ref(c) $\in I$ is called an individual concept node, else (ref(c) $= *$) it is called a generic concept node.*
    $\forall c_1, c_2 \in C$, $lab(c_1) \leq lab(c_2)$ *if and only if*[6] $type(c_1) \leq type(c_2)$ *and* $ref(c_1) \leq ref(c_2)$.
  - $\forall e \in U, lab(e) \in \mathbb{N}$
- *Moreover, lab satisfies*
  - *constraints from σ et τ:*
    - *Edges incident on a relation node are totally ordered, they are numbered from 1 to the degree of the relation node. The i^{th} neighbor of a relation r is denoted by $G_i(r)$.*
    - $\forall r \in R,\ type(G_i(r)) \leq \sigma_i(type(r))$.
    - $\forall c \in C$ *if* $ref(c) \in I$ *then* $\tau(ref(c)) \leq type(c)$.
  - *constraints from conjunctive type definition:*
    $\forall c_1, c_2 \in C$ *where* $ref(c_1), ref(c_2) \in I$, *let us note* $t_1 = type(c_1)$ *and* $t_2 = type(c_2)$, *if* $ref(c_1) = ref(c_2)$ *and* $t_1 \neq t_2$ *then the conjunctive type* $t_1 \cup t_2 \notin \downarrow B$ *; with* $\downarrow B$ *the banned conjunctive concept type set.*

Definition 15 (Projection [5,6,7]). *Let* $H = (C_H, R_H, U_H, lab_H)$ *and* $G = (C_G, R_G, U_G, lab_G)$ *be two conceptual graphs, both based on a same support* $S = (T_C, B, T_R, \sigma, I, \tau)$. *A* projection *from H to G*[7] *is an ordered pair of mappings* $\Pi = (f, g)$, *with* $f : C_H \rightarrow C_G$ *and* $g : R_H \rightarrow R_G$, *such that Π preserves edges and may decrease concept and relation labels:*

- $\forall c \in C_H,\ lab_H(c)$ ***is compatible with*** $lab_G(f(c))$ *(in usual SG model,* compatible *means:* $lab_H(c) \leq lab_G(f(c))$*),*
- $\forall r \in R_H,\ lab_G(g(r)) \leq lab_H(r)$.
- *for every edge rc of U_H, $\Pi(rc)$ is an edge of U_G with the same label. Moreover, if $c = H_i(r)$, then $f(c) = G_i(g(r))$.*

[6] This order on concept node labels is not use just like that in our extension of SG model.

[7] Projection operation has an interest especially if G is in normal form [5].

Towards the Complexity of Recognizing Pseudo-intents

Barış Sertkaya[⋆]

TU Dresden, Germany
sertkaya@tcs.inf.tu-dresden.de

Abstract. Pseudo-intents play a key rôle in Formal Concept Analysis. They are the premises of the implications in the Duquenne-Guigues Base, which is a minimum cardinality base for the set of implications that hold in a formal context. It has been shown that checking whether a set is a pseudo-intent is in coNP. However, it is still open whether this problem is coNP-hard, or it is solvable in polynomial time. In the current work we prove a first lower bound for this problem by showing that it is at least as hard as TRANSVERSAL HYPERGRAPH, which is the problem of identifying the minimal transversals of a given hypergraph. This is a prominent open problem in hypergraph theory that is conjectured to form a complexity class properly contained between P and coNP. Our result explains why the attempts to find a polynomial algorithm for recognizing pseudo-intents have failed until now. We also formulate a decision problem, namely FIRST PSEUDO-INTENT, and show that if this problem is not polynomial, then, unless P = NP, pseudo-intents cannot be enumerated with polynomial delay in a specified lexicographic order.

1 Introduction

Pseudo-intents play a key rôle in Formal Concept Analysis (FCA) [7]. They form the premises of the Duquenne-Guigues Base [9], which is a minimum cardinality base for the set of implications that hold in a formal context. Computational problems related to pseudo-intents have been of major interest to the FCA community since their introduction. Among these problems, the most central one, namely recognizing pseudo-intents, which is the problem of checking whether a given set is a pseudo-intent of a given formal context, has been shown to be in coNP [15,16]. However, so far neither a polynomial time algorithm that solves this problem, nor a proof that this problem is intractable has been found.

In another field of discrete mathematics, namely hypergraph theory [1], problems that show a similar computational behaviour exist as well. The problem known as TRANSVERSAL HYPERGRAPH [3], which is the problem of checking whether the edges of a given hypergraph are precisely the minimal transversals of another given hypergraph, is one such problem. Like the problem of recognizing pseudo-intents, this problem is also known to be in coNP, however whether it

⋆ Supported by the German Research Foundation (DFG) under grant BA 1122/12-1.

is coNP-hard or it is solvable in polynomial time has now been open for more than 20 years. Moreover, many other problems from various fields of computer science have been shown to be computationally equivalent to this problem. Some of these problems are: from relational databases the problem FD-RELATION EQUIVALENCE, which is checking whether a given set of functional dependencies that is in Boyce-Codd Normal Form is a cover of a given relation instance [3], the problem ADDITIONAL KEY for relation instances, which is the problem of checking whether an additional key exists for a given relation instance and a set of minimal keys thereof [3], and from logic the problem MONOTONE DUAL, which is checking whether two monotone Boolean functions given in CNF are mutually dual [4]. Related problems from artificial intelligence can be found in [13], problems from data mining can be found in [10], and a comprehensive survey on these problems can be found in [4]. In a landmark paper Fredman and Khachiyan proved in [5] that TRANSVERSAL HYPERGRAPH can be solved in $n^{o(\log n)}$ time, which implies that this problem is most likely not coNP-hard. It is conjectured that this problem, together with the computationally equivalent problems mentioned above, forms a class properly contained between P and coNP.

The present paper is the first step of an ongoing work investigating whether the problem of recognizing pseudo-intents is computationally equivalent to the abovementioned problems. We show that it is at least as hard as TRANSVERSAL HYPERGRAPH. However, whether it is TRANSVERSAL HYPERGRAPH-complete remains open. Our result explains why the attempts in the FCA community to find a polynomial time algorithm for this problem have failed until now. We also formulate a decision problem, namely FIRST PSEUDO-INTENT, and show that if this problem is not solvable in polynomial time, then, unless P = NP, pseudo-intents cannot be enumerated with polynomial delay in a specified lexicographic order.

2 Preliminaries

2.1 Formal Concept Analysis

Formal Concept Analysis (FCA) [7] is a field of mathematics that is based on a lattice-theoretic formalization of the notions of a concept and a conceptual hierarchy. It facilitates the use of mathematical reasoning for conceptual data analysis and knowledge processing.

In FCA, one represents data in a *formal context*, which in its simplest form is a way of specifying which attributes are satisfied by which objects. A formal context is usually denoted by $\mathbb{K} = (G, M, I)$ where G is the set of *objects*, M is the set of *attributes*, and I is the *incidence relation* between the objects and the attributes. A formal context is usually visualized as a cross table, where the rows represent the objects, and the columns represent the attributes of the context. A cross in column m of row g means that the object g has the attribute m, and the absence of a cross means that g does not have the attribute m. For a set of objects $A \subseteq G$, the *derivation operator* applied to A, which is denoted with A', defines the set of attributes that are satisfied by all objects in A. Similarly, for a set of attributes $B \subseteq M$, the derivation operator applied to B, which is denoted with B', defines the set of objects that satisfy all attributes in B. Double application of the derivation operator yields the *closure operator* $(\cdot)''$. The subsets of M closed under $(\cdot)''$ are called the *concept intents* of $\mathbb{K}$, and when ordered w.r.t. inverse set inclusion, they yield a complete lattice called the *concept lattice* of $\mathbb{K}$. Given a formal context $\mathbb{K} = (G, M, I)$ and an *implication* $P \rightarrow Q$, where $P, Q \subseteq M$, we say that $P \rightarrow Q$ *holds* in $\mathbb{K}$ if the objects that have the attributes in P also have the attributes in Q, i.e., $P' \subseteq Q'$. We denote the *implicational theory* of $\mathbb{K}$, i.e, the set of all implications that hold in $\mathbb{K}$, with $Imp(\mathbb{K})$.

The implicational theory of a formal context $\mathbb{K}$ can be large. Thus, one is interested in small *implicational bases* generating $Imp(\mathbb{K})$. In [9] a canonical implicational base, which is called the *Duquenne-Guigues Base*, of a given formal context has been characterized, and it has been shown that there cannot be another base with fewer implications. The premises of the implications in a Duquenne-Guigues Base are called the pseudo-intents of the underlying formal context. A set $P \subseteq M$ is called a *pseudo-intent* if $P \neq P''$ and $Q'' \subsetneq P$ holds for every pseudo-intent $Q \subsetneq P$. Equivalently, a set $P \subseteq M$ is called a pseudo-intent if $P \neq P''$, it is a quasi-intent, and for every quasi-intent $Q \subsetneq P$, $Q'' \subsetneq P$ holds, where a *quasi-intent* is defined as a set $Q \subseteq M$ that satisfies $R'' \subseteq Q$ or $R'' = Q''$ for any $R \subseteq Q$.

2.2 Hypergraphs

Hypergraph theory [1] is a field of discrete mathematics with many important applications in both theoretical and applied computer science.

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ is a pair consisting of a set of *vertices* $V = \{v_i \mid 1 \leq i \leq n\}$, and a set of *(hyper)edges* $\mathcal{E} = \{E_j \mid 1 \leq j \leq m\}$ where $E_j \subseteq V$. A hypergraph is called *simple* if none of its edges contains another edge, that is, $\forall E, F \in \mathcal{E}.\ E \subseteq F \Rightarrow E = F$. A set of vertices $W \subseteq V$ is called a *transversal* of $\mathcal{H}$ if it intersects every edge of $\mathcal{H}$, i.e., $\forall E \in \mathcal{E}.\ E \cap W \neq \emptyset$. A transversal is called *minimal* if no proper subset of it is a transversal. The set of all minimal transversals of $\mathcal{H}$ constitute another hypergraph on V called the *transversal hypergraph* of $\mathcal{H}$, which is denoted by $Tr(\mathcal{H})$. Generating $Tr(\mathcal{H})$ is an important problem which has applications in many fields of computer science. The well known decision problem associated to this computation problem is defined as follows:

Problem: TRANSVERSAL HYPERGRAPH (TRANS-HYP)
Input: Two hypergraphs $\mathcal{H} = (V, \mathcal{E}_{\mathcal{H}})$ and $\mathcal{G} = (V, \mathcal{E}_{\mathcal{G}})$.
Question: Is $\mathcal{G}$ the transversal hypergraph of $\mathcal{H}$, i.e., does $Tr(\mathcal{H}) = \mathcal{G}$ hold?

We say that a decision problem Π is TRANS-HYP-hard if TRANS-HYP can be reduced to Π by a standard polynomial transformation. We say that Π is TRANS-HYP-complete if it is TRANS-HYP-hard and Π can be reduced to TRANS-HYP by a polynomial transformation.

3 Related Work and Previous Results

Pseudo-intents and computational problems related to them has attracted great attention among the researchers in the FCA community since their introduction. It is well known that for a formal context $\mathbb{K} = (G, M, I)$ the number of pseudo-intents can be exponential in $|M|$. This is for instance the case when object intents are precisely all possible subsets of M with cardinality $|M|/2$. However, in this case $|G|$ as well as $|I|$ are also exponential in $|M|$, thus the number of pseudo-intents is polynomial in $|I|$. In [14] Kuznetsov has given an example of a context where the number of pseudo-intents is exponential in the size of the incidence relation $|I|$. Moreover, he has shown that determining the number of pseudo-intents of a formal context is a #P-hard problem. Given the fact that the number of pseudo-intents can be exponential in the size of the input context, it is clearly not possible to enumerate all pseudo-intents in time polynomial time in the size of this formal context. In complexity theory, for analyzing the performance of enumeration algorithms where the number of solutions can be exponential in the size of the input, one considers other measures. One such measure is to take into account not only the size of the input, but also the size of the output when defining a notion of performance. An algorithm is said to run in *output polynomial time* (or *polynomial total time*) [12] if it outputs all solutions in time polynomial in the size of the input *and the output.* One advantage of an output polynomial algorithm is that it runs in polynomial time (in the size of the input) when there are only polynomially many solutions.

For enumerating pseudo-intents, currently no output polynomial algorithm is known. The most well known algorithm *next closure* [6] by Ganter, as a by-product, enumerates the concept intents as well. That is, its running time depends not only on the number of pseudo-intents but also on the number of concept intents. Since the number of concept intents can be exponential in the number of pseudo-intents, this algorithm in general does not run in output polynomial time. Similarly, the *attribute-incremental algorithm* [17] by Duquenne and Obiedkov has also time complexity depending on both the number of pseudo-intents and the number of concept intents. Recently in [18] we have shown that enumerating pseudo-intents is at least as hard as computing the minimal transversals of a given hypergraph. There we have also identifed a class of formal contexts for which these two problems are computationally equivalent. In [11], for the special case where the concept lattice is meet-semidistibutive, Janssen and Nourine have shown that there are at most polynomially many pseudo-intents, and they can be enumerated in polynomial time. For the special case where the concept lattice is modular, Wild has shown in [19] that an optimal base, i.e., a base that not only contains the minimum number of implications, but also contains the minimum number of attributes, can be computed in polynomial time. In [2] Duquenne has shown that for locally distributive lattices a minimum cardinality base can be computed in polynomial time.

4 Complexity of Recognizing Pseudo-intents

Apart from enumerating and counting pseudo-intents, recognizing them is another important computatinal problem. Kuznetsov and Obiedkov has shown in [15,16], that this problem is in coNP. However, neither a polynomial time algorithm, nor a proof of coNP-hardness has been found so far. In the following we prove a first lower bound for this problem.

First we need to introduce some more notions from hypergraphs. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called *saturated* [3] if every subset of V is contained in at least one of the edges of $\mathcal{H}$, or it contains at least one edge of $\mathcal{H}$, i.e., for every $W \subseteq V$, $W \subseteq E$ holds, or $E \subseteq W$ holds for some $E \in \mathcal{E}$. It has been shown in [3] that checking whether a hypergraph is saturated is coNP-complete. There, a special case of the problem where the given hypergraph is restricted to be simple, has also been considered. It is the following problem:

Problem: SIMPLE HYPERGRAPH SATURATION (SIMPLE-H-SAT)
Input: A simple hypergraph $\mathcal{H} = (V, \mathcal{E})$, i.e., $\forall E, F \in \mathcal{E}.\ E \subseteq F \Rightarrow E = F$.
Question: Is $\mathcal{H}$ saturated, i.e., is it true that for every $W \subseteq V$, $W \subseteq E$ holds or $E \subseteq W$ holds for some $E \in \mathcal{E}$?

It is not difficult to see that this problem is in coNP. However, like the problem of recognizing pseudo-intents, neither a polynomial time algorithm that solves this problem, nor a proof that it is coNP-hard has been found so far. It has been shown in [3] that this problem is under polynomial transformations computationally equivalent to TRANS-HYP. In the following we show that recognizing pseudo-intents is at least as hard as this problem. We start with a formal definition of our problem:

Problem: PSEUDO-INTENT (PSI)
Input: A formal context $\mathbb{K} = (G, M, I)$, and a set $P \subseteq M$.
Question: Is P a pseudo-intent of $\mathbb{K}$?

Now we show that PSI is SIMPLE-H-SAT-hard.

Theorem 1. PSI *is* SIMPLE-H-SAT-*hard.*

Proof. Let an instance of SIMPLE-H-SAT be given with the simple hypergraph $\mathcal{H} = (V, \mathcal{E})$, where $V = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{E_1, \ldots, E_m\}$. From $\mathcal{H}$ we construct an instance of PSI, i.e., a formal context $\mathbb{K}_\mathcal{H} = (G, M, I)$ and a set $P \subseteq M$, as follows: As attributes of $\mathbb{K}_\mathcal{H}$, we take the vertices of $\mathcal{H}$ and two new attributes a and b that do not already occur in V, that is, $M = V \cup \{a, b\}$. For every i, where $1 \leq i \leq m$, we construct an object g_i whose intent is $E_i \cup \{a\}$. In addition, for each i we construct the following objects: Consider edge E_i. For every $F \subsetneq E_i$ such that $|F| = |E_i| - 1$, we create an object with the intent F. E_i has $|E_i|$-many such subsets. We name these objects as $g_{i1}, \ldots, g_{i|E_i|}$. In total $\mathbb{K}_H$ contains $\sum_{i=1}^{m} |E_i| + m$ objects. Figure 1 demonstrates the context $\mathbb{K}_\mathcal{H}$. Finally we create a subset of M just by defining $P = V \cup \{a\}$. It is easy to see that both $\mathbb{K}_\mathcal{H}$ and P can be constructed in polynomial time. We know that $\mathcal{H}$ is simple,

	$v_1 \quad \cdots \quad v_n$	a	b
g_1	E_1	x	
$\vdots$	$\vdots$	$\vdots$	
g_m	E_m	x	
g_{11}	F_{11}		
$\vdots$	$\vdots$		
$g_{1\lvert E_1\rvert}$	$F_{1\lvert E_1\rvert}$		
$\vdots$	$\vdots$		
g_{m1}	F_{m1}		
$\vdots$	$\vdots$		
$g_{m\lvert E_m\rvert}$	$F_{m\lvert E_m\rvert}$		

Fig. 1. Formal context $\mathbb{K}_\mathcal{H}$ constructed from simple hypergraph $\mathcal{H}$

that is none of its edges is contained in another edge. Then $\mathbb{K}_\mathcal{H}$ has the following property:

($*$) Each E_i is contained in only one object intent, namely in $E_i \cup \{a\}$, so $E_i'' = E_i \cup \{a\}$. That is, E_i are not closed. Moreover, strict subsets of E_i are closed. In order to see this, consider a fixed E_i. $\mathbb{K}_\mathcal{H}$ contains the objects $g_{i1}, \ldots, g_{i|E_i|}$ whose intents are all strict subsets of E_i with cardinality $|E_i| - 1$. Every strict subset of E_i can be written as the intersection of such object intents, i.e, every strict subset of E_i is closed. This means that the edges of $\mathcal{H}$ are pseudo-intents of $\mathbb{K}_\mathcal{H}$.

Now we claim that P is a pseudo-intent of $\mathbb{K}_\mathcal{H}$ if and only if $\mathcal{H}$ is saturated. We are going to give a proof of the contrapositive of this claim, i.e., $\mathcal{H}$ is *not* saturated if and only if P is *not* a pseudo-intent of $\mathbb{K}_\mathcal{H}$.

($\Rightarrow$) If $\mathcal{H}$ is not saturated, then there exists a $W \subseteq V$ such that $W \not\subseteq E_i$ and $E_i \not\subseteq W$ for every $1 \leq i \leq m$. Then $W' = \emptyset$, and $W'' = M$ because W is not contained in any object intent. Assume without loss of generality that W is minimal with respect to set inclusion, that is for every $X \subsetneq W$, $X \subseteq E_i$ holds for some i. We know that $E_i \not\subseteq W$ holds for every i. Then every $X \subsetneq W$ is strictly contained in some E_i, that is $X \subsetneq E_i$ for some i. By property ($*$) above we know that strict subsets of E_i are closed, thus X is closed. That is W is not closed, but its all strict subsets are closed, i.e., W is a pseudo-intent of $\mathbb{K}_\mathcal{H}$. Since $W \subsetneq P$ and W'' is not strictly contained in P, P is not a pseudo-intent of $\mathbb{K}_\mathcal{H}$. Thus we have shown that if $\mathcal{H}$ is not saturated, then P is not a pseudo-intent of $\mathbb{K}_\mathcal{H}$.

($\Leftarrow$) If P is not a pseudo-intent of $\mathbb{K}_\mathcal{H}$, then $\mathbb{K}_\mathcal{H}$ has a pseudo-intent $W \subsetneq P$ such that W'' is not strictly contained in P (this is because P is not closed). It cannot be the case that $W'' = P$ since $\mathbb{K}_\mathcal{H}$ does not contain any object whose intent is P or a superset of P. This means that $W'' = M$, i.e., $W' = \emptyset$, that is W is not contained in any object intent. Assume $a \in W$. We know that W is not contained in any object intent. This implies that $W \setminus \{a\}$ is not contained in any object intent either, i.e., $(W \setminus \{a\})'' = M$. Note that $(W \setminus \{a\})$ is a quasi-intent, because every $X \subseteq (W \setminus \{a\})$ satisfies $X'' \subseteq (W \setminus \{a\})$, or $X'' = (W \setminus \{a\})'' = M$.

But this contradicts the fact that W is a pseudo-intent since a pseudo-intent is minimal among the quasi-intents that generate the same closure. Thus, $a \notin W$, i.e., $W \subseteq V$. Due to property $(*)$ each edge E_i of $\mathcal{H}$ is a pseudo-intent of $\mathbb{K}_{\mathcal{H}}$ and its closure contains the attribute a. Since a pseudo-intent contains the closure of all strictly smaller pseudo-intents and $a \notin W$, for every $1 \leq i \leq m$, E_i is not a strict subset of W. In addition W is different from every E_i since $W'' \neq E_i''$ for every i. That is, $E_i \not\subseteq W$ for every $1 \leq i \leq m$. Moreover, since W is a pseudo-intent and its closure $W'' = M$ is not contained in any E_i, W is not contained in any E_i, i.e., for every $1 \leq i \leq m$, $W \not\subseteq E_i$. Thus we have shown that if P is not a pseudo-intent of $\mathbb{K}_{\mathcal{H}}$, then there is a $W \subseteq V$ such that $W \not\subseteq E_i$ and $E_i \not\subseteq W$ for every $1 \leq i \leq m$, i.e., $\mathcal{H}$ is not saturated. This completes the proof of our claim. □

The following is an immediate consequence of Theorem 1 above and Theorem 4.12 in [3]:

Corollary 1. PSI *is* TRANS-HYP*-hard.*

Corollary 1 explains why attempts to find a polynomial time algorithm for solving PSI have failed until now. Because if such an algorithm exists, then this algorithm can also decide TRANS-HYP in polynomial time.

5 Enumerability with Polynomial Delay

One other notion for analyzing the performance of enumeration algorithms is polynomial delay. An algorithm is said to run with *polynomial delay* [12] if the time until the first solution is generated, and thereafter the time between any two consecutive solutions is bounded by a polynomial in the size of the input. Currently we do not know whether pseudo-intents can be enumerated with polynomial delay in a specified order. Assuming that the elements of M are linearly ordered, we say that a set $P \subseteq M$ is *lexicographically smaller* than $Q \neq P$ if the smallest element that distinguishes P and Q belongs to Q. The following decision problem is of crucical importance for the above question:

Problem: FIRST PSEUDO-INTENT (FIRST-PSI)
Input: A formal context $\mathbb{K} = (G, M, I)$, a pseudo-intent $P \subseteq M$ of $\mathbb{K}$, and a linear order on M.
Question: Is P lexicographically the first pseudo-intent of $\mathbb{K}$?

Because if FIRST-PSI cannot be decided in polynomial time, then pseudo-intents cannot be enumerated with polynomial delay in a specified lexicographic order.

Proposition 1. *If* FIRST-PSI *is not in* P*, then unless* P = NP*, pseudo-intents cannot be enumerated in lexicographic order with polynomial delay.*

Proof. It is not difficult to see this. Assume we have an algorithm that enumerates the pseudo-intents of a given formal context with polynomial delay. This means that given a formal context $\mathbb{K} = (G, M, I)$ and a $P \subseteq M$, it generates the lexicographically next pseudo-intent coming after P in polynomial time. If we run this algorithm with the input $\mathbb{K}$ and $P = \emptyset$, which is the lexicographically smallest subset of M, then in polynomial time it generates the lexicographically first pseudo-intent, thus solves FIRST-PSI in polynomial time. $\square$

Of course if PSI turns out to be coNP-hard, then unless P = NP, pseudo-intents cannot be enumerated with polynomial delay since the lexicographically first pseudo-intent cannot be generated in polynomial time. However, even if PSI turns out to be polynomial it can still be the case that FIRST-PSI is intractable. A similar case about maximal independent sets has been investigated in [12]. Although recognizing a maximal independet set is polynomial, there it has been shown that deciding whether a given set is the lexicographically last maximal independent set, is coNP-hard. Thus maximal independent sets cannot be enumerated in reverse lexicographic order with polynomial delay.

6 Concluding Remarks and Future Work

We have shown that recognizing pseudo-intents is at least as hard as recognizing the transversal hypergraph, which is a prominent open problem. This can be taken as a weak evidence that recognizing pseudo-intents is unlikely to be polynomial time solvable. As future work we are going to work further on investigating whether these problems are computationally equivalent, i.e., whether PSI is TRANS-HYP-complete. We are also going to work on solvability of PSI with limited non-determinism [8], and determining the complexity of FIRST-PSI.

References

1. Berge, C.: Hypergraphs. Elsevier Science Publishers B.V, North Holland (1989)
2. Duquenne, V.: The core of finite lattices. Discrete Mathematics 88, 133–147 (1991)
3. Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. SIAM Journal on Computing 24(6), 1278–1304 (1995)
4. Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems in logic and AI. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS, vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
5. Fredman, M.L., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. Journal of Algorithms 21(3), 618–628 (1996)
6. Ganter, B.: Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany (1984)
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
8. Goldsmith, J., Levy, M., Mundhenk, M.: Limited nondeterminism. SIGACT 27(2), 20–29 (1978)
9. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de données binaries. Mathématiques, Informatique et Sciences Humaines 95, 5–18 (1986)

10. Gunopulos, D., Khardon, R., Mannila, H., Toivonen, H.: Data mining, hypergraph transversals, and machine learning. In: Proceedings of the Sixteenth Symposium on Principles of Database Systems (PODS 1997), pp. 209–216 (1997)
11. Janssen, P., Nourine, L.: Minimum implicational basis for meet-semidistributive lattices. Information Processing Letters 99(5), 199–202 (2006)
12. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Information Processing Letters 27(3), 119–123 (1988)
13. Kavvadias, D.J., Papadimitriou, C.H., Sideri, M.: On horn envelopes and hypergraph transversals. In: Ng, K.W., Balasubramanian, N.V., Raghavan, P., Chin, F.Y.L. (eds.) ISAAC 1993. LNCS, vol. 762, pp. 399–405. Springer, Heidelberg (1993)
14. Kuznetsov, S.O.: On the intractability of computing the Duquenne-Guigues Base. Journal of Universal Computer Science 10(8), 927–933 (2004)
15. Kuznetsov, S.O., Obiedkov, S.A.: Counting pseudo-intents and #P-completeness. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS, vol. 3874, pp. 306–308. Springer, Heidelberg (2006)
16. Kuznetsov, S.O., Obiedkov, S.A.: Some decision and counting problems of the Duquenne-Guigues basis of implications. Discrete Applied Mathematics 156(11), 1994–2003 (2008)
17. Obiedkov, S.A., Duquenne, V.: Attribute-incremental construction of the canonical implication basis. Annals of Mathematics and Artificial Intelligence 49(1-4), 77–99 (2007)
18. Sertkaya, B.: Some computational problems related to pseudo-intents. In: Ferré, S., Rudolph, S. (eds.) Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA 2009). LNCS (LNAI), vol. 5548, pp. 130–145. Springer, Heidelberg (2009)
19. Wild, M.: Optimal implicational bases for finite modular lattices. Quaestiones Mathematicae 23, 153–161 (2000)

Another Reason Why Conceptual Graphs Need Actors

B.J. Smith and Harry Delugach

Computer Science Department
University of Alabama in Huntsville
Huntsville, AL 35899
smithbj@uah.edu, delugach@cs.uah.edu

Abstract. Conceptual graphs (CGs) are a knowledge representation formalism that models monotonic first-order logic. However, in the case of an active knowledge base and in other cases, it is necessary to modify a CG dynamically, rendering the preceding static first-order CG possibly inconsistent and in need of further analysis. In order to extend monotonic first-order logic to non-monotonic second-order computation, and therefore achieve all of the power of a modern computer, CGs need to use atomic actors to represent change. To illustrate the power of actors, we represent the well-defined Turing machine; this has the added effect of showing that CGs can represent any of the power of a modern computer. This addition to the CG theory will have other similar practical effects.

Keywords: conceptual graphs, actors, non-monotonic logic, sequence, alternative, iteration, context, Turing machine.

1 Introduction

Conceptual graphs, introduced in [1], are a knowledge representation formalism that models monotonic first-order logic. However, in the case of an active knowledge base and in other cases, it is necessary to modify a CG dynamically, rendering the preceding static first-order CG possibly inconsistent and in need of further analysis. In order to extend monotonic first-order logic to non-monotonic second-order computation, and therefore achieve all of the power of a modern computer, CGs need to use atomic actors as a representation of change. To illustrate the power of actors, we represent the well-defined Turing machine, introduced in [2]. This illustration has the added effect of showing that CGs can represent the power of a modern computer. The impetus of this problem originated in [3], when iteration was needed within CGs to ensure consistency among two different sets of data, which progressed to this paper, and we feel that this addition to the community will have other similar practical effects. In the long term, we feel that by showing that CGs can model Turing machines, we bridge the gap from CG to programming languages.

We begin by defining a Turing machine. Then we will model a Turing machine using CGs. While modeling a Turing machine, we will explain how CGs can represent the necessary features of a Turing machine; sequence, alternative, and iteration, within the existing framework of CGs, and give examples of each. We then discuss an issue that we found while modeling sequence, and conclude the paper.

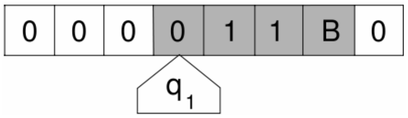


Fig. 1. A Turing machine with a finite tape

2 What is a Turing Machine?

A Turing machine consists of two basic parts. First, there is an infinite "tape" composed of cells. Second, there is a "read/write head" that moves up and down this infinite tape, based on commands represented in those cells, until the head hits a predefined stop command cell, usually represented by a blank (denoted by "B" here). Since CGs are finite, and a sheet of assertion is finite, we will ignore the case of the Turing machine with an infinite tape and will instead begin with a variant, a Turing machine with a finite tape. Fig 1 is a Turing machine with a finite tape, and this is the Turing machine variant that we will mainly consider when we are analyzing CGs in this paper.

It has been shown that a Turing machine can represent any kind of computation [4], and that a Turing machine consists of three basic features. These are (i) *sequence* whereby one step follows another, (ii) *alternative* where one sequence is selected from a set of possible sequences and (iii) *iteration* where a sequence may be repeated.

3 Modeling Turing Machines with CGs

When a Turing machine with a semi-infinite tape is represented by a CG, we represent it by an actor with a set of inputs from a repository of data external to the sheet of assertion; this repository can be considered as semi-infinite, as it has a beginning (the first element of a file), yet no known, a priori ending.

When a Turing machine with a finite tape is represented by a CG, we can represent it the same way as we represent a Turing machine with a semi-infinite tape. We can also represent it by an actor with a set of inputs from a repository of data *on* the sheet of assertion, by using a concept with a set of references, noted by using brackets, as shown in Fig 2 below.

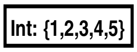


Fig. 2. A CG representation of a set of five distinct integers

The benefit of using a set of referents to represent the tape is that all data is on the sheet of assertion, but the disadvantage is that the tape must be finite. This is not a large disadvantage when dealing with CGs because CGs must be finite, but it should be considered when dealing with Turing machines. Next, we will discuss how the three features of a Turing machine, sequence, alternative, and iteration, are represented in CGs.

3.1 Sequence

To begin, we need to establish how sequence is indicated and enforced in CGs with actors. In the canon of CGs, if a relation is monadic or dyadic, no arc labeling is necessary. But if there are more than one or two arcs indicating a relation, the arcs are labeled from 1 to the number of arcs. Extending this idea, the framework for identifying sequence has already been established, by utilizing labeled links. In Fig 3, we show that in a triadic graph, labels become necessary.

We propose extending the use of these labeled arcs to include enforcing a rule to label all links that connect to actors, regardless of whether their definitions require it. Each actor is defined independently of other actors; therefore each actor has at least one labeled arc, either input or output, in its definition. In order for multiple actors to work together in a CG, a transformation has to occur that retains the definition of individual actors, yet extends to coordinate multiple actors acting together. We want to be able to maintain a relationship between a single actor as defined and an actor interacting with others.

The transformation is simple and extendable. From [1], we have the following definitions. A *source concept* is an input concept to one or more actors, but not an output concept of any actor. An *intermediate concept* is an output concept of exactly one actor and an input concept of one or more actors. A *sink concept* is an output concept of exactly one actor, but not an input concept of any actor. Utilizing these terms, we begin with all of the necessary definition actors (actors with *n* arcs labeled 1 through *n*) placed on the sheet of assertion.

All links are labeled, and we identify each *intermediate concept*, and combine them. The result is the creation of *intermediate concepts* from existing *source* and *sink concepts*, and this is shown in Fig 5. In the above example, [T: *b] is the only *intermediate concept.*

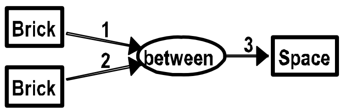


Fig. 3. A labeled, triadic relation

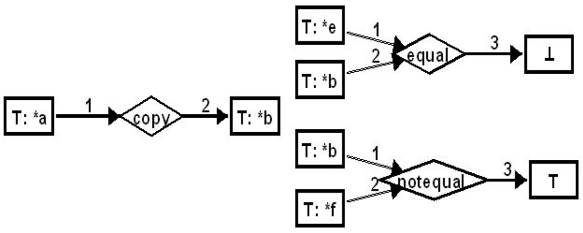


Fig. 4. Step 1 of transforming a CG, finding all needed individual actors

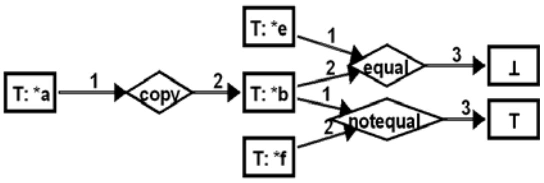


Fig. 5. Step 2 of transforming a CG, combining repeated concepts

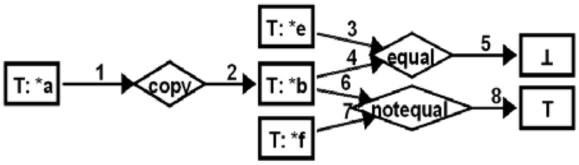


Fig. 6. An example of sequence that maintains the definition of an actor

Once all *intermediate concepts* are linked, the system identifies a critical path by identifying all concepts, relations, and actors involved from all *source concepts* to all *sink concepts*. There may be multiple critical paths of the same length. Now starting with the labeled link from each *source concept* to its actor, and using a depth first transversal, that link would assume the value of 1 and (if available) the *intermediate concepts* of that first actor would assume increasing integer values all the way to the first sink concept. Then the link from the next (if available) *source concept* to its actor would get labeled with the next increasing integer, resulting in the graph shown in Fig 6.

At the end of the process, we have a graph that, when followed in labeled order, reveals the most *sink concepts* in the fewest number of steps. Also, we have a graph that can easily be deconstructed to its original definition actors. An alternative process would involve labeling the links from *source concepts* to actors and *intermediate concepts* all the way to the eventual *sink concepts* in a breadth first transversal, but this would not reveal the most *sink concepts* in the fewest number of steps. To explain by way of an example, we will compare the depth first and breadth first traversals of these graphs. Consider two cases. In case 1, consider a *source concept* whose actor has *m* outgoing *sink concepts*. Using a depth first numbering algorithm and a breadth first numbering algorithm both yield the result of a *sink concept* in one step, each, and all *sink concepts* will be calculated in *m* steps. In this case, neither algorithm performs better than the other and we consider this the best case. In case 2, the worst case, we consider a *source concept* whose actor has *m* number of outgoing *intermediate concepts*, and the number of *intermediate concepts* and *sink concepts* is *n*. In order to calculate the $m * n$ nodes, using a depth first numbering algorithm, we would calculate *n* nodes before we got our first *sink concept*. Using a breadth first numbering algorithm, we would calculate $m * (n - 1)$ nodes before we got our first *sink concept*. Overall, we would still need to calculate $m * n$ nodes to calculate all of our sink nodes using either numbering algorithm, but using a depth first numbering algorithm returns up to the first $n - 1$ *sink concepts* quicker.

It could be argued that Fig 6 is not an example of sequence, but is instead an example of dependence. If this is the argument, then we contend that the sequence of any program that terminates is really just dependent on the final state of that program. We address another pathological case in Section 5.

3.2 Alternative

There are two methods to establish the notion of alternative in CGs. The first method has been shown in [6], but will be explained again here for completeness. Each actor needs an [Enabled] concept as an input. If the [Enabled] referent is true or "T", then the actor is enabled, and if the [Enabled] referent is false or "⊥", then the actor is disabled. With this framework in use, a simple comparison of a changeable concept can enable actors and disable them.

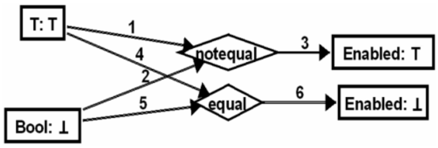


Fig. 7. One example of alternative

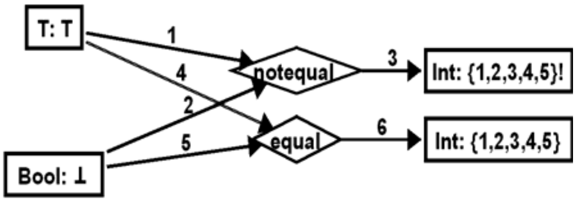


Fig. 8. A more coherent example of alternative

The above-mentioned method of using a type of Enable to control the activity of an actor might seem arbitrary. To create something more within the framework of CGs, we go back to basic principles. From [1], we know that there are *control marks* that determine when actors are executed to compute referents for their output concepts. The three different types of control marks are: the *request mark* "?", the *assertion mark* "!", and the *neutral mark* "°". We know that if all input concepts have referents, then an actor is ready, and that a ready actor will not fire unless it is enabled. We use this knowledge to utilize the *assertion mark* to indicate which concepts are ready for use. Currently, <notequal> and <equal> will output a [T], or a [null], for true and false, respectively. In the case where we want to *disable* an actor, we need another type of actor that will not modify the output referent for false and in the case that we want to *enable* an actor, we need another type of actor that will use the assertion mark for true, as shown in Fig 8.

These changes have the effect of enabling and disabling the actor, and functions within the framework of CGs better than using something like an input of type Enabled.

3.3 Iteration

To demonstrate iteration, we utilize a construct that has been used by C# and Perl already, the idea of a <foreach> actor. As demonstrated in Fig 9, the <foreach> actor takes a set of referents as its input, and outputs the matching type and a referent. The outputted referent is an individual of the input set. The <foreach> actor begins by outputting the first element of the input referent set. When all actions have completed downstream, the <foreach> actor updates its output's referent with the next referent in the set of the input.

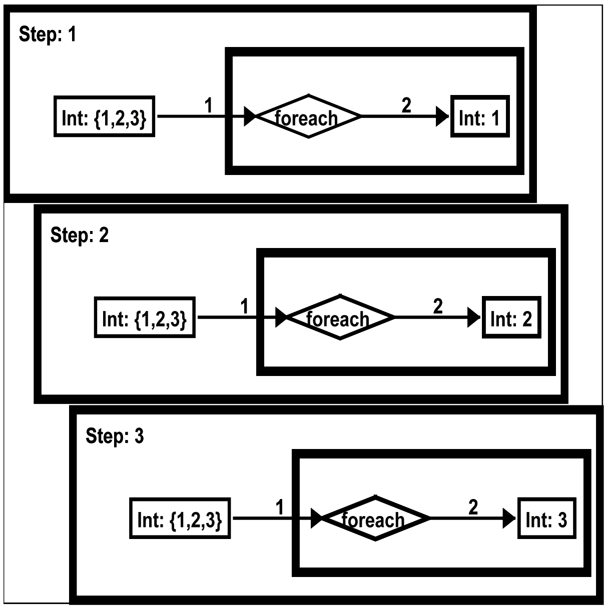


Fig. 9. An example of iteration at three different steps

The above figure shows the activation of <foreach> over three cycles. In [7] it is mentioned that the time structure can be mapped to discrete timestamps of the world, but we would like to further define this idea.

CGs are known for their strength in showing knowledge based on first order logic. A limitation of first order logic and an issue that we will attempt to ameliorate is that its representations are static and its validity may not hold as a graph changes over time. Time and change are not built into first order logic, yet any practical knowledge base must demonstrate knowledge that may constantly be updated.

Having an actor in a CG does not necessarily require that when that actor is used that the entire sheet of assertion must be revalidated. When actors act, they can change referents on the sheet of assertion, and every time that they change a referent, the entire sheet is suspect, and must be revalidated to ensure that the sheet is still consistent.

At least two scenarios are possible. One scenario is that a single actor acts, either by changing or not changing the sheet of assertion. In the case that the sheet of assertion is not changed, then the sheet of assertion still maintains its previous consistency. In the case that the sheet of assertion does change, then all concepts associated with that changed referent must be re-evaluated. This offers just a subset of all of the concepts on the sheet of assertion. The subset's size is based on the connections of the changed referent. Thus, we argue that a benefit of finely-grained CGs is that if a single referent changes, then it affects a much smaller scope than if a referent changes on a monolithic graph.

A second scenario is that a single actor fires multiple times by design. Perhaps the actor is an iterator or a determiner of averages that must act several times until it has completed its job. Regardless, this class of actor must be noted as a class of actor that must complete multiple actions before the results are complete. In this situation, we wait for the actor to finish operating, and then follow the same rules as outlined above in the first scenario, treating a single actor that acts multiple times as a single actor that acts one time.

To constrain the scope of actors in the case of iteration, we require contexts so that the entire graph would not need to be re-validated between each step of the iteration. Contexts are necessary in the first scenario to enclose the area of the conceptual graph that, in the worst case, every concept within needs to be reevaluated. Contexts are also necessary in the second scenario to indicate which concepts must complete before the actor is marked for firing again, which are necessary to formulate loops. We demonstrate the use of contexts to control, in this case <foreach>, in Section 4.

4 Operational Semantics of a Graph with Actors

We can put sequence, alternative, and iteration together to get a programmatic CG. Fig 10 represents the CG that will be our case study. This graph can either sum or multiply the first three positive integers, and give us the result.

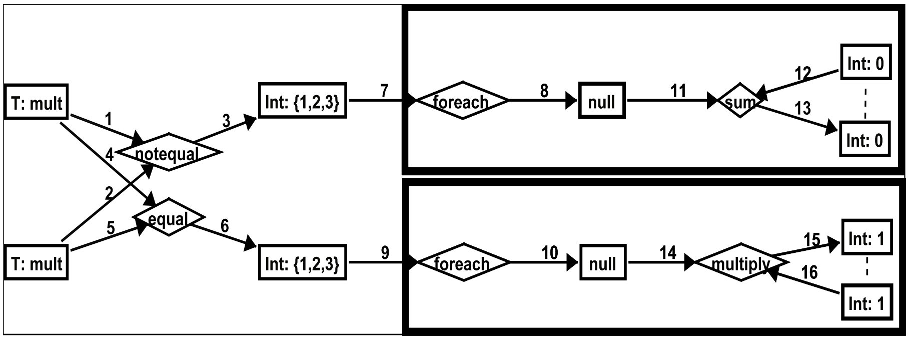


Fig. 10. Sequence, alternative, and iteration shown together initially

Table 1. Steps 1 through 3

Step	Precondition Link Active	Postcondition
1	1,2,4,5	<notequal>, <equal>
2	3,6	[Int: 1,2,3], [Int: 1,2,3]!
3	9	<foreach>

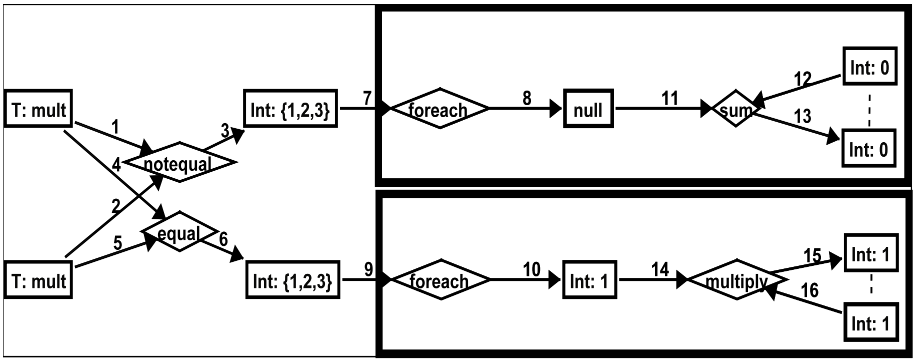


Fig. 11. At step 4

Table 2. Steps 4 through 6

Step	Precondition Link Active	Postcondition
4	10	[Int: 1]
5	14,15	<multiply>
6	16	[Int: 1]

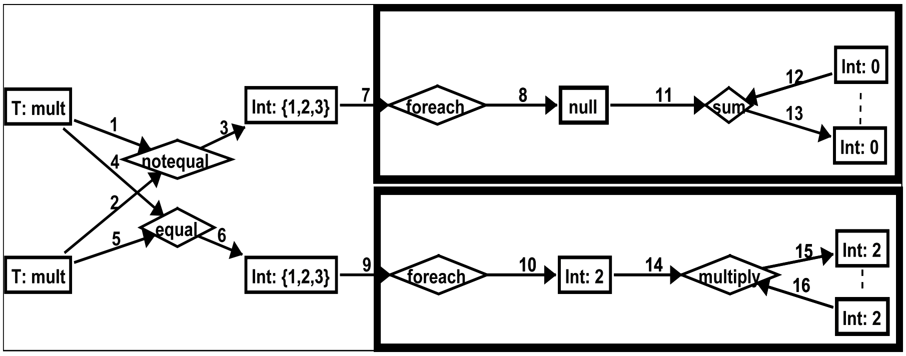


Fig. 12. At step 9

As described in Table 1, during the first step, the links labeled 1, 2, 4 and 5 are active, which gives all necessary inputs to <notequal> and <equal>. At step 2, <notequal>'s output's referent is not changed (this is the new functionality of <equal> and <notequal>), and the <equal>'s output's referent is asserted. Since <notequal>'s referent is not declared, the path consisting of 7, 8, 11, 12 and 13 is disabled, and not considered for the rest of this example. At step 3, the inputs of the <foreach> with the activated input is updated.

As described in Table 2, at step 4, the enabled <foreach> outputs its first input's referent, [Int: 1]. At step 5, <multiply> acts since its inputs are updated, resulting, at step 6, in an output of [Int: 1], since <multiply>'s inputs are [Int: 1] and [Int: 1].

Table 3. Steps 7 through 9

Step	Precondition Link Active	Postcondition
7	10	[Int: 2]
8	14,15	<multiply>
9	16	[Int: 2]

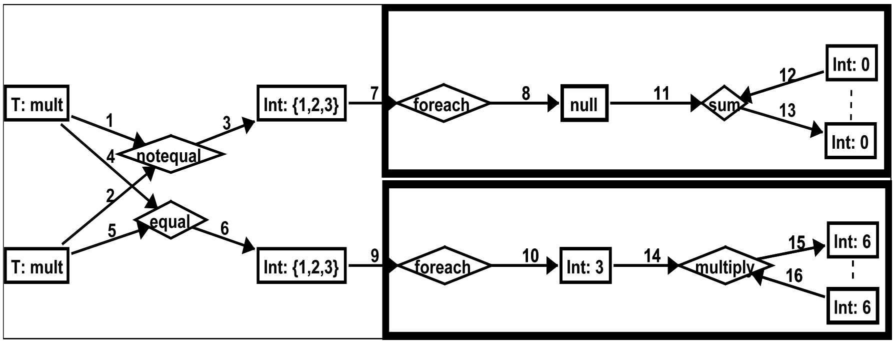


Fig. 13. At step 12

Table 4. Steps 10 through 12

Step	Precondition Link Active	Postcondition
10	10	[Int: 3]
11	14,15	<multiply>
12	16	[Int: 6]

As described in Table 3, during step 7, since the enabled thread had completed (all activity within the context had occurred), <foreach> outputs its next referent in its set, [Int: 2]. At step 8, <multiply> acts since its inputs are updated, resulting, at step 9, in an output of [Int: 2], since <multiply>'s inputs are [Int: 2] and [Int: 1].

As described in Table 4, at step 10, since the enabled thread had completed (all activity within the context had occurred), <foreach> outputs its next referent in its set, [Int: 3]. At step 11, <multiply> acts since its inputs are updated, resulting, at step 12, in an output of [Int: 6], since <multiply>'s inputs are [Int: 3] and [Int: 2].

It is important to note that when a graph with actors is viewed, it is a snapshot of first-order logic, even though the graph with the actors represents second-order logic. This is extended to software that saves a graph with actors. When that graph is reopened, the graph should not necessarily be considered active; otherwise, it would be very difficult to be read by a human. Instead, some sort of activation should have to be initiated before the graph activates.

5 Discussion and Issues of Sequence

We initially wanted to not enforce unique labeling of each link, but instead wanted to label links that could be active during the same step, as shown in Fig 14.

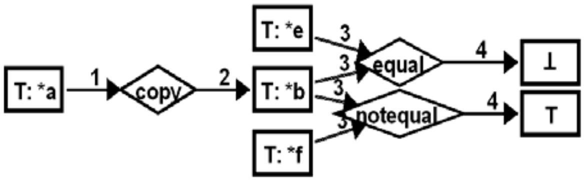


Fig. 14. An example of sequence that violates an actor's definition by not being able to decompose back to all of the actors' definition

This gives us the benefit of being able to see the most concepts that can change in a given step; however, we are not able to make the transformation from a connected graph containing actors back to the set of primitive definition actors.

We also would like to consider a pathological case. What if the outputs of two different actors happen to be the same concept? (From [1], this is a *conflicting concept.*) Fig 15 would be machine dependent and nondeterministic and therefore not encouraged. But it does represent an example of a valid execution of Dijkstra's guarded command [6].

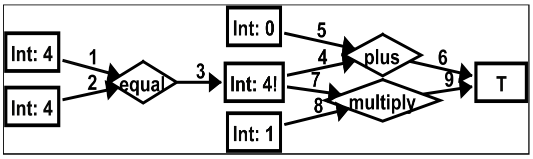


Fig. 15. A pathological case, the guarded command

We consider two scenarios where this type of graph would be useful. In the first, shown above, there are two different ways to arrive at the same value, and, depending on hardware implementation, one way is faster than the other. In the second scenario, we have a case where only one of many possible criteria must be met. For example, an individual wonders if they are eligible for a promotion and in order to be considered, they must satisfy one of two criteria. The potential candidate must have:

A. three years relevant full-time experience after completion of a Master's degree or
B. a Master's degree and five years relevant full-time experience after completion of a Bachelor's degree.

The following graph, Fig 16 describes this situation. Here, [Database: Smith.txt] contains several columns with data that tells us relevant work and education data about an employee. In this scenario, the columns are

1. BeginWork: the year of graduation with a Bachelor's Degree and the year that the employee began working
2. MastersDegree: true if the employee has a Master's degree, false otherwise
3. MSGrad: the year of graduation with a Master's degree

In addition to what is found in the external repository, this graph also contains

1. [Year: 2008]: the referent is populated by the actor <currentyear>
2. [Experience: 2]: the referent is populated by <minus> which is a calculation of the number of years of experience from the year of graduation with a Master's degree and the current year
3. [YearsNeeded: 3]: this referent came from the requirement that if you are going to be promoted based on Rule A, then you must have three years relevant full-time experience
4. [Experience: 5]: this referent is populated by <minus> which is a calculation of the number of years of experience from the first year of work and the current year
5. [YearsNeeded: 5]: this referent came from the requirement that if you are going to be promoted based on Rule B, then you must have five years relevant full-time experience
6. [Promoted: true]: this referent is the result of the employee being promoted because he meets the criteria based on Rule B, but could also apply to the employee if he meets the criteria based on Rule A.

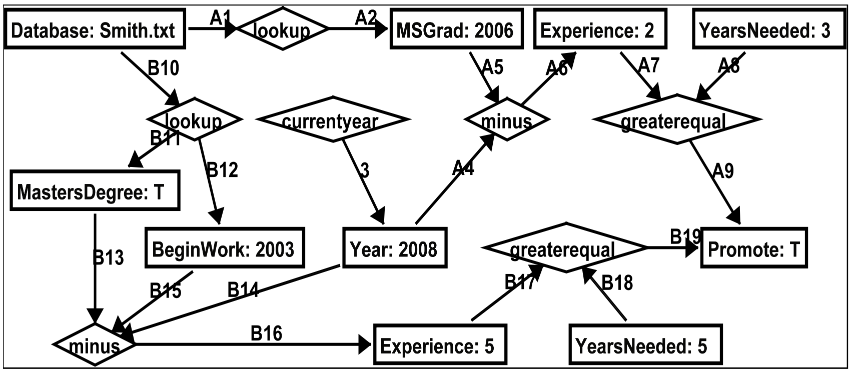


Fig. 16. Another pathological case, only one path needs to be true. One path has arcs that begin with "A," while the other path has arcs that begin with "B".

In order for this to work as shown, there must exist some definition that a result of *true*, here [Promote: true], stops the action of the graph for which there is no simple indicator. Alternatively, there must be a way to define that a result of *false* stops the reasoner's evaluation of the graph. For instance, if an employer was checking the background of a potential employee, then that employer would probably stop the background check as soon as the employer found out that the candidate had a felony conviction. In addition to these features, the automatic parallelization of "A" arcs and "B" arcs would benefit us greatly. As of now, the knowledge engineer must define these arcs himself.

6 Background (or Previous Work)

It is hard to argue against the merits of computers, and programming languages are a powerful force that makes computers general purpose. However, little work has been done to further CGs as a programming language. In [8], a review of the state of dynamic conceptual graphs was written, which focused on MODEL-ECS, while mentioning other executable systems based on conceptual graphs. These include Sowa's Actors and Dataflow Diagrams [1], Nagle's CONGENT [9], Hartley's Actor Graphs [10], Delugach's Dynamic Conceptual Graph [11], Mineau's Conceptual Programming Environment [12], Kabbaj and Frasson's Actors [13], Bos, et al.'s Executable Conceptual Graphs implemented on CoGITo platform [14], and Raban and Delugach's Animating Conceptual Graphs [15].

MODEL-ECS introduced a new relationship to indicate sequence. This conceptual construct, Finish Before Starting (FBS), is very interesting, and easily captures the desire of finishing many activities before beginning a new activity (or activities). However, for many sequences of individual activities, we feel that this would clutter the sheet of assertion. In [11] and [15], the idea of actors asserting referent values is discussed. We choose, in this paper, to really limit the changes and assertions of referents to a sequential and ordered fashion, but also without disagreeing with the idea addressed there. However, the idea of an initiator can describe the first input concept to the first actor in the sequence. The Conceptual Programming Environment [12] has been suggested where imperative programming and logical programming both coexist and complement each other. After [13], there has been additional work creating an object-oriented programming language based on CGs, named Synergy [16]. We particularly liked how Synergy breaks up actors into different slices of activity and how Synergy differentiates coupled and non-coupled actors.

Visual programming languages exist on a large field with many players. The Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) uses the dataflow programming language G which connects different function-nodes by drawing wires. The nodes can execute as soon as all input data is available, allowing inherent support for parallel execution [17], the same way CG actors can support parallel execution. However, LabVIEW is proprietary and is not managed by a third party standards committee. AgentSheets was developed to bridge the gap between what a customer wanted and what a programmer delivers [18]. Storytelling Alice is developed to teach programming, requires memorization of no syntax, and is designed to appeal to those who would not normally be exposed to programming [19]. Alice 2.0 extends Alice towards high school and college students.

All of these systems are building blocks that could lead in the future to bridging the gap between static, first order CGs and dynamic knowledge based systems.

7 Conclusion and Summary

We suggest the incorporation of actors into more uses of CGs. Actors need further definition, perhaps by using primitives, or more formally, using Herbrand's Universe. We also believe that CGs can be considered a functional language. Also, we would like to know where (if at all), CGs fit in with visual programming languages. It would be interesting to compare our CG research with OWL and RDF research. Furthermore, we would like to compare our research with the research done in [20] to ensure consistency. Finally, we would like to incorporate the work done in [21] regarding, among other topics, their use of sequences and their proofs regarding while loops.

References

1. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
2. Turing, A.M.: Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE). Teddington, UK (1945)
3. Smith, B.J., Delugach, H.S.: A Framework for Analyzing and Testing Requirements with Actors in Conceptual Graphs. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS, vol. 4068, pp. 401–412. Springer, Heidelberg (2006)
4. Tucker, A., Noonan, R.: Programming Languages, 2nd edn. McGraw-Hill, New York (2007)
5. Delugach, H.S.: CharGer Conceptual Graph Editor. version 3.6, `http://sourceforge.net/projects/charger/` (accessed, August 2008)
6. Dijkstra, E.W.: A Discipline of Programming. Prentice Hall, Englewood Cliffs (1976)
7. Mineau, G.W.: From Actors to Processes: The Representation of Dynamic Knowledge Using Conceptual Graphs. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS, vol. 1453, pp. 65–79. Springer, Heidelberg (1998)
8. Lukose, D., Mineau, G.W.: A Comparative Study of Dynamic Conceptual Graphs. In: 11th Workshop on Knowledge Acquisition, Modeling and Management, Albert, Canada (1998)
9. Nagle, T.E.: Conceptual Graphs in an Active Agent Paradigm. In: Proceedings of the IBM Workshop on Conceptual Graphs, Thornwood, USA (1986)
10. Hartley, R.H.: The Foundation of Conceptual Programming. In: Proceedings of the First Annual Rockey Mountain Conference on Artificial Intelligence, Boulder, Colorado, USA (1986)
11. Delugach, H.S.: Specifying Multiple-Viewed Software Requirements With Conceptual Graphs. Journal Systems and Software 19, 207–224 (1992)
12. Mineau, G.W.: Constraints on Processes: Essential Elements for the Validation and Execution of Processes. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS, vol. 1453, pp. 66–82. Springer, Heidelberg (1998)
13. Kabbaj, A., Frasson, C.: Dynamic CG: toward a general model of computation. In: Ellis, G., Rich, W., Levinson, R., Sowa, J.F. (eds.) ICCS 1995. LNCS, vol. 954, pp. 46–60. Springer, Heidelberg (1995)

14. Bos, C., Botella, B., Vanheeghe, P.: Modeling and Simulating Human Behaviours with Conceptual Graphs. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 275–299. Springer, Heidelberg (1997)
15. Raban, R., Delugach, H.S.: Animating Conceptual Graphs. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 431–445. Springer, Heidelberg (1997)
16. Kabbaj, A.: Synergy as an Hybrid Object-Oriented Conceptual Graph Language. In: Tepfenhart, W.M. (ed.) ICCS 1999. LNCS, vol. 1640, pp. 198–213. Springer, Heidelberg (1999)
17. Optimizing your LabVIEW FPGA VIs: Parallel Execution and Pipelining, `http://zone.ni.com/devzone/cda/tut/p/id/3749#toc1` (accessed March, 2009)
18. Rausch, M.: AgentSheets – Programming above C-Level. Computer Graphik Topics 10, 10–12 (1998)
19. Cooper, S., Dann, W., Pausch, R.: ALICE: A 3-D Tool for Introductory Programming Concepts. In: CCSCNE, Ramapo, NJ (April 2000)
20. Chein, M., Mugnier, M.-L.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer, Heidelberg (2008)
21. Huth, M., Ryan, M.: Logic in Computer Science: Modeling and Reasoning about Systems, 2nd edn. Cambridge University Press, Cambridge (2004)

Relational Scaling in Relational Semantic Systems[*]

Karl Erich Wolff

Mathematics and Science Faculty
Darmstadt University of Applied Sciences
Schoefferstr. 3, D-64295 Darmstadt, Germany
karl.erich.wolff@t-online.de
http://www.fbmn.fh-darmstadt.de/home/wolff

Abstract. In this paper two developments in Conceptual Knowledge Processing are combined, namely Contextual Logic introduced by Rudolf Wille and Temporal Concept Analysis introduced by the author. The basic structures connecting both theories are Relational Semantic Systems (RSS), each consisting of conceptual scales and a Relational Data Systems (RDS) for the representation of relational knowledge. We introduce the notion of a concept graph of a RSS. As opposed to the definition of a concept graph of a power context family the concepts used in the concept graph of a RSS are taken from the conceptual scales and not from the concept lattices of the contexts of k-ary relations. For the graphical representation of relational knowledge in information maps we modify tools from Temporal Concept Analysis and develop *relational trace diagrams*. Its usefulness is shown in a small example of a Relational Semantic System.

1 Introduction

In this paper relational conceptual structures are investigated with the purpose to develop practically successful methods for the representation, evaluation and visualization of relational structures. These investigations are based on the Theory of Conceptual Graphs as developed by J. Sowa [So84, So00] and the mathematization of concepts in Formal Concept Analysis [Wi82, GW99a]. For the purpose of representing relational knowledge R. Wille [Wi97] has introduced power context families. To represent judgments he has introduced the notion of a concept graph of a power context family which is now the main tool in Contextual Judgment Logic and Contextual Conclusion Logic where inferences between formal judgments are studied [Pr98, PW99]. To combine the knowledge representation in power context families with the advantages of many-valued contexts and the successful tools for conceptual scaling, as for example the program TOSCANAJ (see [BH05]), the notion of Relational Scaling has been introduced in [PW99] and connected to database theory in [He02]. These ideas had been

[*] This work is supported by DFG-RFBR project COMO, GZ: 436 RUS 113/829/0-1.

continued in the paper *A Contextual-Logic Extension of TOSCANA* [EGSW00]. From its abstract we cite:

As graphical representations we recommend, besides labelled line diagrams of concept lattices and Sowa's diagrams of conceptual graphs, particular information maps for utilizing background knowledge as much as possible.

In the following we contribute to these investigations by combining power context families and concept graphs with Conceptual Semantic Systems as introduced by the author in [Wo05b, Wo06, Wo07a]. A first step into that direction was the introduction of the notion of a *Relational Data System* (RDS) which, combined with a family of conceptual scales, yields the notion of a *Relational Semantic System* [Wo09]. These structures help to clarify the discussion about relational and conceptual scaling in power context families; they also improve the practical representation of relational knowledge, including the development of useful information maps.

1.1 Relational Trace Diagrams as Information Maps

An example of a nice information map about flights in Austria is shown in [EGSW00], Fig. 8, where surrounded by the border-line of Austria the Austrian towns with airport are represented as small circles which are connected by arrows indicating flight connections. Each arrowhead is connected by a dashed line with a small data table indicating further information about this flight connection, as for example the corresponding flight numbers, its departure and arrival times and its days.

The construction of a good information map needs a clever combination of several graphical tools, in this case the border-line of Austria, the correctly embedded points for the towns, the arrows and the small data tables. It is obvious that the geographical information about the border-line and the towns are *not* given in the data table on the Austrian flights as represented in Fig. 1 in [EGSW00].

It was shown by the author [Wo07a] how such geographical information can be used to construct an information map which may also include life tracks of moving objects (see Fig. 3 in [Wo07a]). The corresponding conceptual theory is based on the notion of *Temporal Conceptual Semantic Systems*. The ideas developed there seem to be relevant also for the discussion of relational conceptual structures in general. A first example has been shown by the author in [Wo09] where an animation of the Austrian flights [EGSW00] is explained using life tracks of flights.

In this paper we develop *relational trace diagrams* which are based on a new representation of objects which generalizes the broadly employed *objects as formal objects strategy* (OFOS) to represent objects from application domains as formal objects in many-valued contexts or formal contexts.

1.2 Representation of Objects: Not Necessarily as Formal Objects

Conceptual Semantic Systems have been introduced by the author [Wo04] with the purpose to understand the physical notions of "particles" and "waves" from a conceptual point of view. These notions are now well understood in that framework. The main idea was to study the ternary relation that "an object is at some time at some place" with respect to different granularities. That led to the notion of a *distributed object* as for example a wave or a wave packet. For that reason it was necessary to think about the role of formal objects and the consequences of representing objects (like particles) as formal objects. The main consequence of this representation is that each formal object has a unique object concept in each part of the derived context of the scaled many-valued context. That is nice if we represent a particle as a formal object and represent its place as an object concept in the concept lattice representing the space. Such an object "occupies a single point" in that space. But sometimes we wish to represent a particle, for example a ball or an electron, in such a way that it occupies a certain volume in space. For that purpose it is necessary to represent the objects not as formal objects; the mostly used alternative is to represent objects as values in the data table. That is for example the case when we represent several kinds of objects, like persons, days, places and some judgments of the kind that a person visited a place at some day. In the practice of many applications of FCA it is tried to choose a suitable kind of objects, for example the persons, to be represented as formal objects. Usually, the other objects are then represented as values in a many-valued context. It is well-known for all experts in FCA that the obligation to choose in some given application domain a suitable kind of objects to be represented as formal objects often yields no problems, but in some applications it is quite problematic. A typical example was our search for a good representation of temporal data where for example several persons at several points of time had to be represented conceptually. Should we take the persons as formal objects or the points of time? My actual solution to this problem, as given in the notion of a Conceptual Semantic System, is to take none of the many kinds of objects as formal objects. Instead, the formal objects g are interpreted as *basic judgments*, each representing the information given in the row of g in the data table of the given Conceptual Semantic System.

1.3 Formal Concepts as Values in a Many-Valued Context

At that point of the discussion we should notice that the usage of the notion of a "value" in a many-valued context does not fit really well with the accepted and basic doctrines of concepts, judgments, and conclusions [Wi00]. It seems natural to understand a "value" as a concept, and to represent it as a formal concept in some formal context which explains the meaning of the given "value".

Therefore, in the definition of a conceptual semantic system (CSS) [Wo04] we start with a family $(\mathbb{S}_m)_{m\in M}$ of formal contexts; their formal concepts are used as values of a many-valued context, described by a mapping $\lambda : G \times M \to W$ which satisfies $\lambda(g,m) \in \mathfrak{B}(\mathbb{S}_m)$ for all $g \in G$ and all $m \in M$.

Each element of $g \in G$ represents the information $(\lambda(g, m)_{m \in M})$ given in its row of the data table. For the theory and applications of CSS and temporal CSS the reader is referred to [Wo04, Wo05b, Wo06, Wo07a, Wo07b]. Useful construction methods for information maps based on (temporal) CSS have been demonstrated in these papers, for example in [Wo07a], Fig. 3 where a weather map with a moving high pressure zone is shown. It is well-known that such maps are valuable tools for the representation of relational knowledge in a suitably chosen granularity. The formal connection between these two important fields of research, namely the relational structures on one side and the granularity structures on the other side are not well understood until now. In this paper we do some steps to improve our understanding of the connections among these fields of research.

1.4 Relations and Granularity

For the purpose of combining relations and granularity in a mathematical theory I like to use well-established approaches, namely to ground on the philosophical doctrines of concepts, judgments, and conclusions as emphasized by R. Wille [Wi00]. I also agree with Wille's idea to represent relations as formal concepts, such that the extent of such a relation concept is a subset of a direct product of sets; hence the usual mathematical notion of a relation corresponds to the extent of a relation concept.

For the formal representation of statements like
"ALICE works as a TEACHER in BERLIN" we introduce the relation " . works as a . in ." and formalize that statement as an *infon* in the sense of Devlin [De91] by (. works as a . in .; ALICE, TEACHER, BERLIN). In general, an infon has the form $(R; x_1, ..., x_k)$ where R denotes a k-ary relation and $(x_1, ..., x_k) \in R$.

For the contextual representation of a set of infons with possibly different arities of their relations we represent each infon in a row of a data table. The formal definition of such conceptual structures has been introduced by the author in [Wo09] under the name *Relational Semantic Systems* (RSS) since they are Conceptual Semantic Systems with an additional structure for the formal representation of the relational information. This additional relational structure is defined in the notion of a *Relational Data Systems* (RDS). The precise definitions will be recalled from [Wo09] in the following.

2 Relational Data Systems

The main idea for the introduction of a Relational Data System can be explained easily using the example in Table 1 where five infons labeled from 1,...,5 are represented.

To represent statements like "BOB lives in ENGLAND" or "ALICE works as a TEACHER in BERLIN" in a data table we have to represent some linear ordering on the set of words of the statement. One possibility is to use the spoken sequence of the words of the given statement, but we restrict ourselves

Table 1. A data table for relational information

infon	r*	$PERSON_1$	$PERSON_2$	PROFESSION	LOCATION
1	.lives in.	BOB			ENGLAND
2	.lives in.	BOB			LONDON
3	.works as a.in.		ALICE	TEACHER	BERLIN
4	.meets.in.	BOB	ALICE		PARIS
5	.is the native town of.	ALICE			PARIS

Table 2. The arity-position table

c	$\alpha(c)$	$PERSON_1$	$PERSON_2$	PROFESSION	LOCATION
.lives in.	2	1			2
.works as a.in.	3		1	2	3
.meets.in.	3	1	2		3
.is the native town of.	2	2			1

to statements of the form of an infon $(R; x_1, \ldots, x_n)$. The example "PARIS is the native town of ALICE" shows that we do not insist that the statements have to be read from left to right in the table; clearly, the set M of attributes of a RDS is not assumed to be ordered.

In general, we map each value c of the many-valued attribute r* (called the *relational attribute*) first to a non-negative integer $\alpha(c)$ which is interpreted as its arity, second to a subset of the set of many-valued attributes $\beta(c)$ which is called the *region* of c, and third, for $\beta(c) \neq \emptyset$, to a bijection π_c which assigns to each integer $i \in [1, \alpha(c)]$ a many-valued attribute $\pi_c(i) \in \beta(c)$, called the "$i-$th position of c". In our example the mappings α, β and π_c are given in Table 2.

For example, for the relation ".lives in." the arity is 2, the region β(.lives in.) = {$PERSON_1$, LOCATION} and the first position of ".lives in." is $PERSON_1$, its second position is LOCATION. From Table 1 we see that there are two rows in which the relation ".lives in." occurs, giving the statements "BOB lives in ENGLAND" and "BOB lives in LONDON".

In the following mathematical definition of a Relational Data System we do not represent any specific interpretation of the values as for example the interpretation that a value "is a relation". Therefore we do not use here the dots occuring in the relation names.

2.1 Definition of a Relational Data System

The following definition of a Relational Data System has been introduced by the author in [Wo09] with the purpose to represent any given power context family, and any given concept graph, and to combine them with the possibility for conceptual scaling which will be done in Relational Semantic Systems later.

Definition 1. *"Relational Data System"*
Let G, M, W be sets, $\lambda : G \times M \to W$. Then $\mathfrak{R} := (G, M, W, \lambda, r^*, A, \alpha, \beta, \pi)$ is called a Relational Data System (RDS) if

- $r^* \in M$
- $A \subseteq W_{r^*} := \{\lambda(g, r^*) \mid g \in G\}$
- $\alpha : W_{r^*} \to \mathbb{N} := \{1, 2, \ldots\}$
- $\beta : W_{r^*} \to \mathfrak{P}(M \setminus \{r^*\}) := \{X \mid X \subseteq M \setminus \{r^*\}\}$
- π *is a mapping which maps each* $c \in W_{r^*}$ *with* $\beta(c) \neq \emptyset$ *to a bijection* $\pi_c : [1, \alpha(c)] \to \beta(c)$.

For $\beta(c) \neq \emptyset$ and $1 \leq i \leq \alpha(c)$ the many-valued attribute $\pi_c(i)$ is called the $i-$th position of c. For $c \in W_{r^*}$ the integer $\alpha(c)$ is called the arity of c. For $c \in W_{r^*}$ let $D_c := \{g \in G \mid \lambda(g, r^*) = c\}$, and for $\beta(c) \neq \emptyset$ and $g \in D_c$ let $\vec{c}(g) := (\lambda(g, \pi_c(i)))_{1 \leq i \leq \alpha(c)}$; $\vec{c}(g)$ is called the tuple of c at g. The set $A \subseteq W_{r^*}$ is called the set of artificial relations. The set $A \subseteq W_{r^*}$ will be used for the representation of a normed power context family $(\mathbb{K}_k)_{k \in S}$ by an RDS; then the *isolated* formal objects of $\mathbb{K}_k$, namely the elements of $\{g \in G_k \mid g^{\uparrow} = \emptyset\}$ will be collected in an artificial relation k^*. Then $\beta(c) \neq \emptyset$ for all $c \in A$ and the set $\tau(A) := \{\vec{c}(g) \mid c \in A, \, g \in D_c\}$ is disjoint from $\tau(W_{r^*} \setminus A)$.

3 Relational Data Systems and Power Context Families

Power context families have been introduced by Wille [Wi97] using the following definition.

> A *power context family* is a sequence $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ of formal contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ with $G_k \subseteq (G_0)^k$ for $k = 1, 2, \ldots$. The formal concepts of $\mathbb{K}_k$ with $k = 1, 2, \ldots$ are called *relation concepts*, because they represent k-ary relations on the object set G_0 by their extents.

For our purposes it is not necessary to distinguish between the formal contexts $\mathbb{K}_0$ and $\mathbb{K}_1$. Furthermore, we would like to make explicit the dots "..." in the previous definition by introducing a set S of indices.

Definition 2. *"Normed Power Context Family of Type S"*
$\vec{\mathbb{K}} := (\mathbb{K}_k)_{k \in S}$ *is called a normed power context family of type* S *if*

- $S \subseteq \mathbb{N} := \{1, 2, \ldots\}$, $1 \in S$, *and*
- $\mathbb{K}_k = (G_k, M_k, I_k)$ *is a formal context for all* $k \in S$ *such that* $G_k \subseteq (G_1)^k$ *for* $k > 1$.

$\vec{\mathbb{K}}$ *is called a normed power context family (NPCF) if* $\vec{\mathbb{K}}$ *is a normed power context family of type* S *for some set* S.

Clearly each power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ in the sense of the definition in [Wi97] can be represented by a normed power context family, simply by deleting $\mathbb{K}_0$ and replacing $\mathbb{K}_1$ by $(G_0, M_0 \cup M_1, I_0 \cup I_1)$ assuming M_0 and M_1 disjoint.

It has been proven by the author in [Wo09] that each normed power context family can be faithfully represented by a RDS in the following sense.

In [Wo09] two operators $\mathbf{R}$ and $\vec{\mathbf{K}}$ have been introduced; $\mathbf{R}$ maps each normed power context family $\vec{\mathbb{K}}$ to a Relational Data System $\mathbf{R}(\vec{\mathbb{K}})$, while $\vec{\mathbf{K}}$ maps each Relational Data System $\mathfrak{R}$ to a normed power context family $\vec{\mathbf{K}}(\mathfrak{R})$; in [Wo09], Proposition 1 says that for any normed power context family $\vec{\mathbb{K}}$ (with non-empty contexts) we get $\vec{\mathbf{K}}(\mathbf{R}(\vec{\mathbb{K}})) = \vec{\mathbb{K}}$; and Proposition 2 shows that for any RDS $\mathfrak{R}$ (with non-empty contexts in its power context family) we get $\vec{\mathbf{K}}(\mathbf{R}(\vec{\mathbf{K}}(\mathfrak{R}))) = \vec{\mathbf{K}}(\mathfrak{R})$, but $\mathbf{R}(\vec{\mathbf{K}}(\mathfrak{R}))$ is not necessarily equal to $\mathfrak{R}$.

4 Relational Semantic Systems

4.1 Definition of a Relational Semantic Systems

In this section we recall from [Wo09] the notion of a Relational Semantic System (RSS) which covers the notions of a Conceptual Semantic System (CSS) [Wo05b, Wo06] and the notion of a Relational Data System. That makes explicit the idea that we list in each row of a data table an infon $(R; c_1, \ldots, c_k)$. According to traditional philosophical logic with its doctrines of concepts, judgements, and conclusions (cf. [Wi00]) we start with concepts $c_1, \ldots, c_k$ which are combined with a relational concept R to build an infon $(R; c_1, \ldots, c_k)$. Therefore, we represent the concepts $R, c_1, \ldots, c_k$ as formal concepts of formal contexts $\mathbb{S}_m$ $(m \in M)$. These formal contexts will play the role of conceptual scales of the many-valued context described in the following definition of a Relational Semantic System.

Definition 3. *"Relational Semantic System"*
Let $\mathfrak{R} := (G, M, W, \lambda, r^*, A, \alpha, \beta, \pi)$ be a Relational Data System and for each $m \in M$ let $\mathbb{S}_m := (G_m, N_m, I_m)$ be a formal context and $\underline{\mathfrak{B}}(\mathbb{S}_m)$ its concept lattice. If $\lambda : G \times M \to W$ satisfies $\lambda(g, m) \in \mathfrak{B}(\mathbb{S}_m)$ for all $g \in G$ and all $m \in M$, then the pair $(\mathfrak{R}, (\mathbb{S}_m)_{m \in M})$ is called a Relational Semantic System (RSS).

Remark: If $(\mathfrak{R}, (\mathbb{S}_m)_{m \in M})$ is a RSS, then $(G, M, (\underline{\mathfrak{B}}(\mathbb{S}_m))_{m \in M}, \lambda)$ is a CSS.

4.2 Example of a Relational Semantic Systems

We construct a RSS $(\mathfrak{R}_1, (\mathbb{S}_m)_{m \in M})$ from the
RDS $\mathfrak{R}_1 := (G, M, W, \lambda, r^*, A, \alpha, \beta, \pi)$ given in Table 1 and Table 2, where $G := \{1, 2, 3, 4, 5\}$, $M := \{r^*, PERSON_1, PERSON_2, PROFESSION, LOCATION\}$, $W := \bigcup_{m \in M} \mathfrak{B}(\mathbb{S}_m)$; to define the scales we denote for any

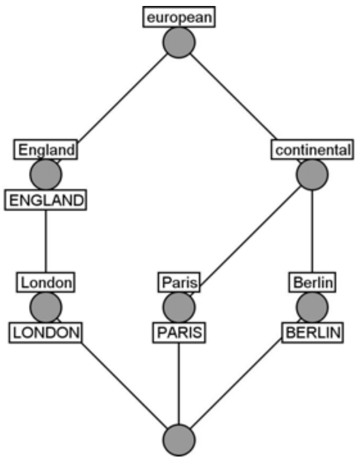


Fig. 1. The scale for LOCATION

set X by $N(X) := (X, X, =)$ the nominal scale on X.
$\mathbb{S}_{r^*} := N(\{$.lives in., .works as a.in, .meets.in., .is the native town of.$\})$,
$\mathbb{S}_{PERSON_1} := \mathbb{S}_{PERSON_2} := N(\{\text{BOB, ALICE}\})$,
$\mathbb{S}_{PROFESSION} := N(\{\text{TEACHER}\})$,
$\mathbb{S}_{LOCATION}$ is given by the line diagram of its concept lattice in Fig. 1.

For $g \in G$ and $m \in M$ let $\lambda(g, m) := \gamma_m(h)$ be the object concept in $\mathbb{S}_m$ of the formal object h which occurs in row g and column m in Table 1. For example, the fact that the name "BOB" occurs in Table 1 in row 1 and column "PERSON$_1$" is understood as an abbreviation for $\lambda(1, \text{PERSON}_1) = \gamma_{PERSON_1}(BOB)$. For short, each value in Table 1 is a formal object in the scale of its column and represents its object concept in that scale.

In other examples the tabular description of the mapping λ may not use the object names for the representation of object concepts, for example if some λ-values are not object concepts. To continue the definition of the RDS $\mathfrak{R}_1$ we mention that r^* is the relational attribute, that $A := \emptyset$, and that α, β, π can be seen from the arity-position table in Table 2.

5 Concept Graphs of a Relational Semantic System

5.1 Definition of a Concept Graph of a Relational Semantic System

To define the notion of a *concept graph of a Relational Data System* we recall the definition of a relational graph ([Wi04]).

A *relational graph* is a structure (V, E, ν) consisting of two disjoint sets V and E together with a map $\nu : E \to \bigcup_{k=1,2,...} V^k$; the elements of V and E are called *vertices* and *edges*, respectively, and $\nu(e) = (v_1, \ldots, v_k)$ is read: $v_1, \ldots, v_k$ are the *adjacent vertices* of the *k-ary edge* e ($|e| := k$ is the *arity* of e; the arity of a vertex is defined to be 0). Let $E^{(k)}$ be the set of all elements of $V \cup E$ of arity k ($k = 0, 1, 2, \ldots$).

In the following definition of a concept graph of a RSS we modify the definition of a concept graph of a power context family.

Definition 4. *"Concept Graph of a Relational Semantic System"*
Let $(\mathfrak{R}, (\mathbb{S}_m)_{m\in M})$ *be a RSS and* $\mathfrak{R} := (G, M, W, \lambda, r^*, A, \alpha, \beta, \pi)$*. A concept graph of* $(\mathfrak{R}, (\mathbb{S}_m)_{m\in M})$ *is a structure* $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ *for which*

- (V, E, ν) *is a relational graph, where the arity of an edge* $e \in E$ *is* $|e| := k$*, if* $\nu(e) \in V^k$
- $\kappa: V \cup E \to \bigcup_{m\in M} \mathfrak{B}(\mathbb{S}_m)$
- $\rho: V \to \mathfrak{P}(\bigcup_{m\in M\setminus\{r^*\}} \mathfrak{B}(\mathbb{S}_m))\setminus\{\emptyset\}$

such that for $v \in V$ *and* $e \in E$

1. $\kappa(v) \in \bigcup_{m\in M\setminus\{r^*\}} \mathfrak{B}(\mathbb{S}_m)$
2. $\kappa(e) \in \mathfrak{B}(\mathbb{S}_{r^*})$
3. *if* $\nu(e) = (v_1, \ldots, v_k)$*, then* $\alpha(\kappa(e)) = k = |e|$ *and for* $1 \leq i \leq k$
4. $\kappa(v_i) \in \mathfrak{B}(\mathbb{S}_{p(e,i)})$*, where* $p(e, i) := \pi_{\kappa(e)}(i)$ *is the i-th position of* $\kappa(e)$ *and*
5. $\rho(v_i) \subseteq \mathfrak{B}(\mathbb{S}_{p(e,i)})$ *such that*
6. $d \leq \kappa(v_i)$ *for all concepts* $d \in \rho(v_i)$ *and*
7. *for all* $(d_1, \ldots, d_k) \in \rho(v_1) \times \ldots \times \rho(v_k)$ *there is a* $g \in G$ *such that* $\lambda(g, r^*) = \kappa(e) =: c$ *and* $\vec{c}(g) = (d_1, \ldots, d_k)$*.*

This definition of a concept graph of a RSS differs from the definition of a concept graph of a power context family in a remarkable point, namely in the choice of the employed concept lattices: for a concept graph of a RSS the concept lattices $\mathfrak{B}(\mathbb{S}_m)$ are employed, while for a concept graph of a power context family $(\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ the concept lattices $\mathfrak{B}(\mathbb{K}_k)$ are used. Either of the two series of concept lattices are needed: the concept lattices $\mathfrak{B}(\mathbb{S}_m)$ represent the meaning of the formal concepts which occur as values in a given RSS $(\mathfrak{R}, (\mathbb{S}_m)_{m\in M})$; the concept lattice $\mathfrak{B}(\mathbb{K}_k)$ of the power context family $\vec{\mathbf{K}}(\mathfrak{R})$ represents the intersections of all chosen $k-$ary relations.

An example of a concept graph of a Relational Semantic System will be shown in the next subsection.

5.2 Example of a Concept Graph of a Relational Semantic System

Fig. 2 shows a graphic representing a concept graph of the Relational Semantic System $(\mathfrak{R}_1, (\mathbb{S}_m)_{m\in M})$. This graphic is drawn according to J. Sowa's convention for drawing conceptual graphs [So84]. This concept graph represents a judgment consisting of the first four infons in Table 1.

The graphic in Fig. 2 visualizes the relational graph with three edges e_1, e_2, e_3, drawn as ellipses, and six vertices $v_1, \ldots, v_6$, drawn as boxes. The ellipse of an edge e is connected to the boxes of its adjacent vertices by straight lines which are labeled by the integers $1, \ldots, |e|$ where $|e|$ is the arity of e. For example, $\nu(e_2) = (v_1, v_4, v_3)$ is the tuple of adjacent vertices of e_2. Now it is easily seen that Fig. 2 represents a relational graph.

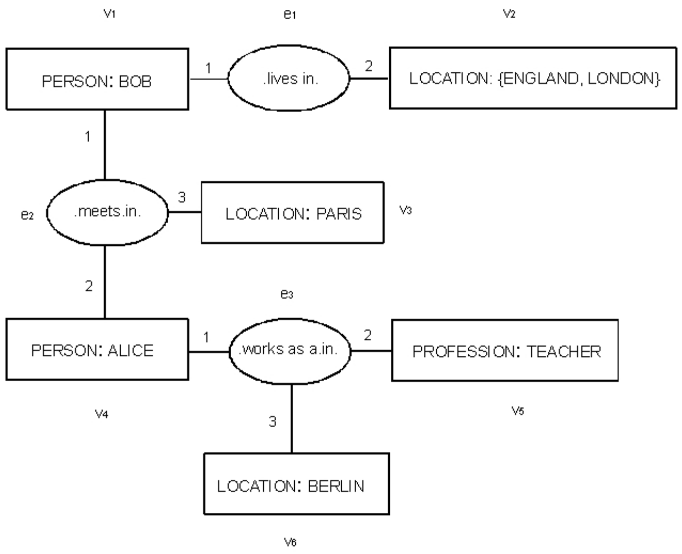


Fig. 2. A concept graph of a Relational Semantic System

To define the mappings κ and ρ using Fig. 2 we mention that the name in the ellipse of an edge e denotes the formal concept $\kappa(e)$; for example, $\kappa(e_1)$ is the object concept of the formal object ".lives in." in the scale $\mathbb{S}_{r^*}$. The first name in the box of a vertex v denotes the formal concept $\kappa(v)$, the second name denotes its reference set $\rho(v)$. In Fig. 2 the first name in a box denotes the top concept of the corresponding scale; for example, $\kappa(v_2) = LOCATION$ is understood here as the top concept in the concept lattice of $\mathbb{S}_{LOCATION}$. Therefore, one can easily check that κ and ρ are mappings as demanded.

Conditions (1.) to (3.) in Def. 4 are obviously satisfied, for example, the arity $\alpha(\kappa(e_1)) = \alpha(\text{.lives in.}) = 2 = |e_1|$. Condition (4.) is satisfied; for example, in the tuple $\nu(e_2) = (v_1, v_4, v_3)$ the vertex v_4 as the *second* entry in that tuple satisfies that $\kappa(v_4)$ is the top concept of the scale $\mathbb{S}_{PERSON_1} = \mathbb{S}_{PERSON_2}$, hence it is an element of $\mathfrak{B}(\mathbb{S}_{PERSON_2})$ and $PERSON_2 = p(\kappa(e_2), 2)$ is the *second* position of $\kappa(e_2)$, the object concept of ".meets.in.". Condition (5.) is also satisfied; as an example we choose edge e_1 and its *second* vertex v_2. Its reference set is $\rho(v_2) = \{\gamma_{LOCATION}(ENGLAND), \gamma_{LOCATION}(LONDON)\}$ which is a subset of $\mathfrak{B}(\mathbb{S}_{p(e_1,2)}) = \mathfrak{B}(\mathbb{S}_{LOCATION})$. Obviously, condition (6.) is satisfied; we discuss condition (7.) for the edge e_1; for each of the two 2-tuples in the set $\rho(v_1) \times \rho(v_2)$ there is an infon satisfying the condition, namely infon 1 and infon 2, see Table 1. Condition (7.) is satisfied obviously also for the other edges.

6 Relational Trace Diagrams

Trace diagrams have been introduced by the author [Wo07a]. The main idea for the construction of trace diagrams stems from usual weather maps where

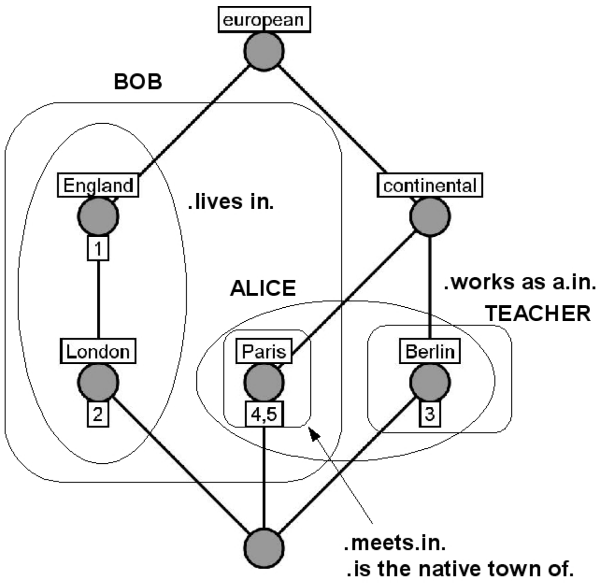


Fig. 3. An information map of the location view

information about many entities like pressure or temperature is embedded into a single map. In this section we introduce *relational trace diagrams* of Relational Semantic Systems using a small example.

6.1 Example of a Relational Trace Diagram

Fig. 3 shows a relational trace diagram of the Relational Semantic System of the previous example.

It is constructed as follows: starting from the RSS $(\mathfrak{R}_1, (\mathbb{S}_m)_{m\in M})$ in the previous example we take its underlying CSS $(G, M, (\underline{\mathfrak{B}}(\mathbb{S}_m))_{m\in M}, \lambda)$ where $\mathbb{S}_m = (G_m, N_m, I_m)$ $(m \in M)$ and construct its (semantically) derived context [Wo07a]

$$\mathbb{K} := (G, N, J) \text{ where } N := \{(m, n) \mid m \in M, n \in N_m\} \text{ and}$$
$$gJ(m, n) :\Longleftrightarrow n \in int(\lambda(g, m)).$$

Remark: Roughly speaking, each formal concept $\lambda(g, m) \in \mathfrak{B}(\mathbb{S}_m)$ is represented in the derived context by its intent.

Then we select a *view* Q which is defined as a subset $Q \subseteq N$ and construct its corresponding subcontext $\mathbb{K}_Q := (G, Q, J \cap (G \times Q))$, called the $Q-$*part* of $\mathbb{K}$. For the concept lattice in Fig. 3 the view is chosen as the set of the LOCATION-attributes in N. The concept lattice of the $Q-$part is shown in Fig. 3. To understand the role of the infons 1,...,5 in this $Q-$part we mention, that for each many-valued attribute $m \in M$ and each formal object (infon) $g \in G$ in the $m-$part $\mathbb{K}_m := (G, \{m\} \times N_m, J \cap (G \times (\{m\} \times N_m)))$ of the (semantically) derived context $\mathbb{K} := (G, N, J)$ we have

$$g^J = \{(m,n)|n \in B\} \text{ in } \mathbb{K}_m \iff B \text{ is the intent of } \lambda(g,m) \text{ in } \mathbb{S}_m$$

by definition of the semantically derived context.

For example, for the many-valued attribute LOCATION and the infon 2 the set $B = \{\text{London, England, european}\}$ is the intent of $\lambda(2, LOCATION) = \mu_{LOCATION}(London)$, the attribute concept of *London* in the scale $\mathbb{S}_{LOCATION}$.

To explain the representation of relational knowledge in Fig. 3 we use the object representation and the traces of objects in CSS as introduced in [Wo07a, Wo07b].

6.2 Object Representation and Traces of Objects

For the formal definition of the notion of an *object* in Conceptual Semantic Systems and the definition of a trace of an object in some view Q the reader is referred to [Wo07a, Wo07b]. To present the main ideas quickly we use our example of the RSS $(\mathfrak{R}_1, (\mathbb{S}_m)_{m \in M})$ where we represent the object "BOB" not as a formal object, but as the formal concept $\mu_{PERSON_1}(BOB)$ of the scale $\mathbb{S}_{PERSON_1}$. Then we select the set $S := \{g \in G | \lambda(g, PERSON_1) = \mu_{PERSON_1}(BOB)\}$ and construct the set of object concepts $\gamma_{LOCATION}(S)$ which is a special trace of the formal concept $\mu_{PERSON_1}(BOB)$ in the LOCATION-view. This set of three object concepts is visualized in Fig. 3 by the rectangle labeled "BOB".

Clearly, in the same way we can represent traces of relation concepts in $\mathfrak{B}(\mathbb{S}_{r^*})$. They are shown in Fig. 3 for all four relation concepts. The same can be done for subtuples of an infon; for example, we might be interested in which locations Alice works as a teacher, for short denoted by the *question* "ALICE works as a TEACHER in ?"; then we would find only the object concept of 3 in the LOCATION-view, and that is the attribute concept of BERLIN.

The construction of such traces can be supported in the computer program TOSCANAJ choosing the scale of the view Q as the last in the priority list for the nested line diagrams. For example, for the question "ALICE works as a TEACHER in ?" one should choose the priority list (PERSON$_2$, r^*, PROFESSION, LOCATION). Clicking on ALICE, .works as a.in., TEACHER in their scales yields only those object concepts which satisfy all the conditions. Hence the trace of the object tuple (ALICE, .works as a.in., TEACHER) consists only of the object concept of 3, hence we get as answer the location "BERLIN".

7 Conclusion and Future Work

In this paper two developments in Conceptual Knowledge Processing are combined, namely Contextual Logic introduced by Rudolf Wille and Temporal Concept Analysis introduced by the author. The basic structures which can now serve for both theories are Relational Semantic Systems. They are defined using the notion of a Relational Data System which has been introduced recently by the author for the representation of relational knowledge [Wo09].

A Relational Semantic System $(\mathfrak{R}, (\mathbb{S}_m)_{m \in M})$ contains the conceptual scales $\mathbb{S}_m$ as well as the relational scales, namely the formal contexts of its power context family $\vec{\mathbf{K}}(\mathfrak{R})$. The concept graphs of a Relational Semantic System represent formal concepts of the conceptual scales, while the concept graphs of a power context family represent the formal concepts of the relational scales.

For the graphical representation of relational structures we have employed the object representation by tuples and the visualization of traces of objects as developed by the author in Temporal Concept Analysis. That yields trace diagrams for the representation of relational knowledge. Its usefulness is shown in a small example of a Relational Semantic System.

Future work has to combine the rich theory in Contextual Logic with Relational Semantic Systems. One of the most challenging problems is the development of a Conclusion Logic in Relational Semantic Systems. In many problems in practice the relational structures change with time, therefore temporal Relational Semantic Systems should be developed.

References

[BH05] Becker, P., Hereth Correia, J.: The ToscanaJ Suite for Implementing Conceptual Information Systems. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 324–348. Springer, Heidelberg (2005)

[De91] Devlin, K.: Logic and Information. Cambridge University Press, Cambridge (1991)

[EGSW00] Eklund, P., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of TOSCANA. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867. Springer, Heidelberg (2000)

[GW99a] Ganter, B., Wille, R.: Formal Concept Analysis: mathematical foundations. Springer, Heidelberg (1999); German version: Springer, Heidelberg (1996)

[He02] Hereth, J.: Relational Scaling and Databases. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS (LNAI), vol. 2393, pp. 62–76. Springer, Heidelberg (2002)

[Pr98] Prediger, S.: Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik. Dissertation, TU Darmstadt 1998. Shaker, Aachen (1998)

[PW99] Prediger, S., Wille, R.: The lattice of concept graphs of a relationally scaled context. In: Tepfenhart, W.M. (ed.) ICCS 1999. LNCS (LNAI), vol. 1640, pp. 401–414. Springer, Heidelberg (1999)

[So84] Sowa, J.F.: Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading (1984)

[So00] Sowa, J.F.: Knowledge representation: logical, philosophical, and computational foundations. Brooks Cole Publ. Comp., Pacific Grove (2000)

[Wi82] Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered sets, pp. 445–470. Reidel, Dordrecht (1982)

[Wi97] Wille, R.: Conceptual Graphs and Formal Concept Analysis. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS (LNAI), vol. 1257, pp. 290–303. Springer, Heidelberg (1997)

[Wi00] Wille, R.: Contextual Logic summary. In: Stumme, G. (ed.) Working with conceptual structures: Contributions to ICCS 2000, pp. 265–276. Shaker-Verlag, Aachen (2000)

[Wi04] Wille, R.: Implicational Concept Graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 52–61. Springer, Heidelberg (2004)

[Wo01] Wolff, K.E.: Temporal Concept Analysis. In: Mephu Nguifo, E., et al. (eds.) ICCS 2001 International Workshop on Concept Lattices-Based Theory, Methods and Tools for Knowledge Discovery in Databases, Stanford University, Palo Alto (CA), pp. 91–107 (2001)

[Wo02a] Wolff, K.E.: Transitions in Conceptual Time Systems. In: Dubois, D.M. (ed.) International Journal of Computing Anticipatory Systems, CHAOS 2002, vol. 11, pp. 398–412 (2002)

[Wo04] Wolff, K.E.: 'Particles' and 'Waves' as Understood by Temporal Concept Analysis. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 126–141. Springer, Heidelberg (2004)

[Wo05a] Wolff, K.E.: States, Transitions, and Life Tracks in Temporal Concept Analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS, vol. 3626, pp. 127–148. Springer, Heidelberg (2005)

[Wo05b] Wolff, K.E.: States of Distributed Objects in Conceptual Semantic Systems. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS (LNAI), vol. 3596, pp. 250–266. Springer, Heidelberg (2005)

[Wo06] Wolff, K.E.: Conceptual Semantic Systems - Theory and Applications. In: Goncharov, S.S., Downey, R., Ono, H. (eds.) Mathematical Logic in Asia. Proceedings of the 9th Asian Logic Conference, pp. 288–301. World Scientific, New Jersey (2006)

[Wo07a] Wolff, K.E.: Basic Notions in Temporal Conceptual Semantic Systems. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS, vol. 4390, pp. 97–120. Springer, Heidelberg (2007)

[Wo07b] Wolff, K.E.: Applications of Temporal Conceptual Semantic Systems. In: Zagoruiko, N.G., Palchunov, D.E. (eds.) Knowledge - Ontology - Theory. Russian Academy of Sciences. Sobolev Institute for Mathematics. Novosibirsk, vol. 2, pp. 3–16 (2007)

[Wo09] Wolff, K.E.: Relational Semantic Systems, Power Context Families, and Concept Graphs. In: The Proceedings of the International Conference on Formal Concept Analysis 2009, Darmstadt (accepted for publication, 2009)

Author Index

Angelova, Galia 100

Baget, Jean-François 86
Boytcheva, Svetla 100
Burrow, Andrew 114

Cooke, James 247
Croitoru, Madalina 86, 128, 233

Dau, Frithjof 141
Delugach, Harry 293

Eremin, Evgeny 155

Ferré, Sébastien 159
Fortin, Jérôme 86

Ganascia, Jean-Gabriel 219
Genest, David 270
Giesbrecht, Eugenie 173
Greaves, Mark 1

Haase, Peter 48
Habib, Bassel 219

Ignatov, Dmitry I. 185

Kent, Robert E. 201
Knechtel, Martin 141
Kuznetsov, Sergei O. 185

Laudy, Claire 219
Leclère, Michel 233
Loiseau, Stéphane 270
Loke, Jeremy 247
Loukachevitch, Natalia 2

Majumdar, Arun K. 32
Moreau, Nicolas 233
Mugnier, Marie-Laure 23

Polovina, Simon 247
Priss, Uta 257

Raimbault, Thomas 270

Sertkaya, Barış 284
Smith, B.J. 293
Sowa, John F. 32
Studer, Rudi 48

Thomopoulos, Rallou 86, 128
Tran, Thanh 48

Wille, Rudolf 66
Wolff, Karl Erich 307