Klaus-Dieter Schewe
Neeraj Kumar Singh (Eds.)

LNCS 11815

Model and Data Engineering

9th International Conference, MEDI 2019
Toulouse, France, October 28–31, 2019
Proceedings

Springer

Klaus-Dieter Schewe · Neeraj Kumar Singh (Eds.)

Model and Data Engineering

9th International Conference, MEDI 2019
Toulouse, France, October 28–31, 2019
Proceedings

Springer

Editors
Klaus-Dieter Schewe
UIUC Institute
Zhejiang University
Zhejiang, China

Neeraj Kumar Singh
INPT-ENSEEIHT/IRIT
Toulouse, France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-32064-5 ISBN 978-3-030-32065-2 (eBook)
https://doi.org/10.1007/978-3-030-32065-2

LNCS Sublibrary: SL2 – Programming and Software Engineering

Preface

The International Conference on Model and Data Engineering (MEDI) is an international forum for the dissemination of research accomplishments on database modeling and data management. Specifically, MEDI provides a stimulating environment for the presentation of research on database models, database theory, data processing, database systems technology, and advanced database applications. This international scientific event, initiated by researchers from Euro-Mediterranean countries, also aims at promoting the creation of North–South scientific networks, projects as well as faculty/student exchanges.

In recent years, MEDI has taken place in Marrakesh, Morocco (2018), Barcelona, Spain (2017), Almería, Spain (2016), Rhodes, Greece (2015), Larnaca, Cyprus (2014), Armantea, Italy (2013), Poitiers, France (2012), and Óbidos, Portugal (2011). The ninth edition of MEDI took place in Toulouse during October 28–31, 2019. The Program Committee (PC) received 41 submissions from 30 countries around the world. The selection process was rigorous, where each paper received at least four reviews. The PC, after careful discussions, decided to accept 11 full papers, two application papers, one vision paper, and seven short papers. Accepted papers cover broad research areas on both theoretical, systems, and practical aspects. Some trends found in accepted papers include mining complex databases, concurrent systems, machine learning, swarm optimization, query processing, Semantic Web, graph databases, formal methods, model-driven engineering, blockchain, cyber physical systems, IoT applications, and smart systems.

Four keynotes were presented at MEDI 2019. César A. Muñoz from NASA gave a keynote on the "Formal Methods in the Development of Highly Assured Software for Unmanned Aircraft Systems." His talk first advocated the use of expressive logics to specify the operational and functional requirements of unmanned systems and to prove the correctness of these requirements, then gave an overview of the formal methods developed at NASA LaRC, and illustrated their use in the design, validation, and verification of highly assured autonomous unmanned aircraft systems. Jan Van den Bussche from Hasselt University presented "Fully Generic Queries: Open Problems and Some Partial Answers" dealing with the highly relevant and up-to-date problem of capturing manipulations on nested big data, where output can be generated without a need for looking in detail at the atomic data elements. The keynote by Bernhard Thalheim from Christian Albrechts University of Kiel was dedicated to "Semiotics in Databases" discussing foundations of syntax, semantics, and pragmatics in databases based on first-order predicate logic, highlighting the usefulness and pointing out problematic issues. He surveyed the beauty of classic database constraint theory and developed alternative approaches to some constraints including the handling of constraint sets, visual reasoning on constraints, and calculi for robust reasoning and object-relational reasoning. Finally, Rémi Delmas from ONERA presented a keynote on “Reinforcement Learning-Based Methods for Falsification: A New Trend in Critical Controllers Verification” emphasizing the use of reinforcement learning algorithms for property falsification together with an illustration of the novel approach on a significant Airbus case study.

MEDI 2019 was organized and supported by INP-ENSEEIHT. The conference would not have been possible without the contributions and support of the Institut de Recherche en Informatique de Toulouse (IRIT).

The main event was preceded by three workshops: the Workshop on Modeling, Verification and Testing of Dependable Critical Systems (DETECT), the Workshop on Data Science for Social Good in Africa (DSSGA), and the Workshop on Security Privacy in Models and Data (TRIDENT).

We would like to thank all the authors who submitted their work to MEDI 2019. In addition, we would like to thank the PC members and the external reviewers who carefully reviewed all submissions and selected the best contributions. Finally, we extend our thanks to the Organizing Committee members and local organizers who contributed to the success of MEDI 2019.

The EasyChair conference management system was set up for MEDI 2019. We thank EasyChair for providing a platform for managing the submissions, reviewing process, and proceedings production.

October 2019

Klaus-Dieter Schewe
Neeraj Kumar Singh

Organization

Program Committee

El Hassan Abdelwahed	University Cadi Ayyad Marrakech, Marocco
Alberto Abello	Universitat Politècnica de Catalunya, Spain
Yamine Aït Ameur	IRIT/INPT-ENSEEIHT, France
Idir Aït Sadoune	LRI, Centrale Supélec, France
Christian Attiogbe	University of Nantes, France
Ladjel Bellatreche	LIAS/ENSMA, France
Sidi Mohamed Benslimane	Ecole Superieure en Informatique, Algeria
Jorge Bernardino	ISEC, Polytechnic Institute of Coimbra, Portugal
Drazen Brdjanin	University of Banja Luka, Bosnia and Herzogovina
Francesco Buccafurri	UNIRC, Italy
Wellington Cabrera	University of Houston, USA
Antonio Corral	University of Almeria, Spain
Florian Daniel	Politecnico di Milano, Italy
Alex Delis	University of Athens, Greece
Georgios Evangelidis	University of Macedonia, Greece
Ylies Falcone	University of Grenoble Alpes, CNRS, Inria, France
Alfio Ferrara	University of Milan, Italy
Flavio Ferrarotti	Software Competence Centre Hagenberg, Austria
Mamoun Filali-Amine	IRIT, France
Enrico Gallinucci	University of Bologna, Italy
Matteo Golfarelli	University of Bologna, Italy
Raju Halder	Indian Institute of Technology Patna, India
Brahim Hamid	IRIT, University of Toulouse, France
Slimane Hammoudi	ESEO, France
Luis Iribarne	University of Almería, Spain
Mirjana Ivanovic	University of Novi Sad, Serbia
Nadjet Kamel	University Ferhat Abbes Setif, Tunisia
Selma Khouri	Ecole Supérieure d'Informatique (ESI), Algeria
Adamantios Koumpis	University of Passau, Germany
Eva Kühn	Vienna University of Technology, Austria
Regine Laleau	Paris Est Creteil University, France
Yves Ledru	Université Grenoble Alpes, France
Sebastian Link	The University of Auckland, New Zealand
Zhiming Liu	Southwest University, China
Ivan Luković	University of Novi Sad, Serbia
Hui Ma	Victoria University of Wellington, New Zealand
Sofian Maabout	LaBRI, University of Bordeaux, France
Yannis Manolopoulos	Open University of Cyprus, Cyprus

Dominique Mery	Université de Lorraine, LORIA, France
Mohamed Mosbah	LaBRI, University of Bordeaux, France
Chokri Mraidha	CEA LIST, France
Yassine Ouhammou	LIAS/ENSMA, France
Marc Pantel	IRIT/INPT, Université de Toulouse, France
Oscar Pastor Lopez	Universitat Politècnica de València, Spain
Jaroslav Pokorný	Charles University in Prague, Czech Republic
Giuseppe Polese	University of Salerno, Italy
S. Ramesh	General Motors R&D, USA
Elvinia Riccobene	University of Milan, Italy
Oscar Romero	Universitat Politècnica de Catalunya, Spain
Manoranjan Satpathy	IIT Bhubaneswar, India
Milos Savic	University of Novi Sad, Serbia
Klaus-Dieter Schewe	Zhejiang University, UIUC Institute, China
Timos Sellis	Swinburne University of Technology, Australia
Neeraj Singh	INPT-ENSEEIHT/IRIT, France
Bernhard Thalheim	Christian Albrechts University of Kiel, Germany
Riccardo Torlone	Roma Tre University, Italy
Ismail Toroslu	Middle East Technical University, Turkey
Goce Trajcevski	Iowa State University, USA
Javier Tuya	University of Oviedo, Spain
Michael Vassilakopoulos	University of Thessaly, Greece
Panos Vassiliadis	University of Ioannina, Greece
Hao Wang	University of Science and Technology, Norway
Qing Wang	The Australian National University, Australia
Alan Wassyng	McMaster University, Canada
Robert Wrembel	Poznan University of Technology, Poland

Additional Reviewers

Bilalli, Besim
Caruccio, Loredana
Crass, Stefan
Fernández-García, Antonio J.
Haq, Anam
Mallios, Nikolaos
Mammar, Amel

Local Organization

Yamine Aït Ameur
Sarah Benyagoub
Guillaume Dupont
Raphaele Esculier
Marc Pantel
Annabelle Sansus
Neeraj Singh

Abstracts of Keynote Presentations

Formal Methods in the Development of Highly Assured Software for Unmanned Aircraft Systems

César Muñoz

NASA Langley Research Center, Hampton, Virginia, USA
cesar.a.munoz@nasa.gov

Abstract. In traditional software development methodologies, operational and functional requirements of systems are often specified in structured natural language notations. These restricted notations provide good documentation support, but only provide limited support for semantic analysis. These notations are not rich enough to unambiguously specify the requirements of safety-critical systems that, for example, involve complex numerical computations or that interact with the physical environment. Examples of these safety-critical systems are autonomous vehicles such as unmanned aircraft systems. This talk advocates the use of expressive formal logics, such as higher-order logic, to specify the operational and functional requirement of unmanned systems and to prove the correctness of these requirements. Semantic analysis of requirements written in higher-order logic is supported through the use of interactive theorem provers. Formal models serve as ideal reference implementations of functional requirements. Hence, formal logics enable software validation techniques where software implementations can be checked against functional requirements in a mechanical way. The Formal Methods group in the Safety-Critical Avionics Systems Branch at NASA Langley Research Center (LaRC) has conducted research on the development and application of formal methods technology to safety-critical applications of interest to NASA for more than 30 years. This talk provides an overview of the formal methods technology developed at NASA LaRC and illustrates its use in the design, validation, and verification of highly-assured autonomous unmanned aircraft systems.

Reinforcement Learning-Based Methods for Falsification: A New Trend in Critical Controllers Verification

Rémi Delmas

ONERA, The French Aerospace Lab, Toulouse, France
`remi.delmas@onera.fr`

Abstract. The talk gives an overview of a relatively recent trend in critical embedded controller verification: the use of (possibly deep) reinforcement learning algorithms for property falsification. The central idea is to use temporal logics with real-valued robust semantics to formulate safety objectives, and to formulate the property falsification problem as reward optimization problem, which can be solved using reinforcement learning algorithms for optimal planning or optimal policy synthesis. After introducing basic definitions and concepts, we review a collection of landmark papers, then we illustrate the approach with results obtained on an significant Airbus case study. Last, we outline current challenges and future research directions.

Contents

Keynote Articles

Semiotics in Databases

Bernhard Thalheim(✉)

Department of Computer Science, Christian-Albrechts University at Kiel, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de
http://www.is.informatik.uni-kiel.de/~thalheim

Abstract. In database research and practice, syntax is commonly considered a "firstness" property, while semantics is a "secondness" property (in the sense of Peirce); pragmatics is largely neglected. This paper discusses foundations in first-order predicate logic, highlights its usefulness, but also point out its problematic issues. These cover in particular safe expressions in the relational tuple calculus, rigid normalisation emphasising atomicity of attributes, and a large body of knowledge on database constraints. Database theory is still oriented on flat relational structures although systems became object-relational. We first survey the goodliness of classical database constraint theory and then develop an alternative approach to some constraints including (α) the handling of constraint sets instead of homogeneous classes of constraints, (β) visual reasoning on constraints and structures, and (γ) calculi for robust reasoning, in particular for "exceptions" and object-relational reasoning.

Keywords: Constraints · Dependencies · Database theory · Semiotics

1 Introduction

1.1 Semiotics: Syntax $\bowtie$ Semantics $\bowtie$ Pragmatics

Semiotics is the study of signs and sign-using behaviour. According to Peirce, signs are icons, indexes, or symbols. The structuralist approach to semiotics distinguishes syntax as the arrangement of words and the study of formation of sentences, semantics as the study of meaning in natural and artificial languages, and pragmatics as the study of relations between languages and their users, e.g. the use of languages in communication. Morphology studies the internal construction of words. Natural languages use typically semantically fixed base elements from a vocabulary and have general purpose means for expressions. They use a manifoldness of expression forms and speech acts. They have a high error tolerance, use a metalanguage on its own, don't distinguish abstraction levels, interpret connectives and quantifies in a variable way, widely use ambiguities and ellipses, apply indefiniteness as a concept for later focusing, and use tempus and modality as well as metaphoric elements with an embedment of context.

Therefore, semiotics in linguistics is far more broad than in computer science and database research. Main elements of semiotics are, however, inherited

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 3–19, 2019.
https://doi.org/10.1007/978-3-030-32065-2_1

for computer science research. For instance, database semantics considers how meaning is constructed, interpreted, clarified, obscured, illustrated, simplified, negotiated, contradicted and paraphrased. Formal semantics is typically given in the Fregean and Quine sense by (1) by an interpreter that maps syntactic types to semantic types, (2) by a context abstraction that is based on an aggregation of values which remain fixed in certain temporal and spatial intervals, (3) by states that provide a means of representing changes over time and space, (4) by an configuration that is based on an association of contexts and states, (5) by an interpretation function that yields state results based on certain computation, (6) by an evaluation function that yield some value results for the syntactic types, and (7) by an elaboration function that yield both state and some other value results. Computer sciences uses a small set of syntax constructors instead of using a larger number of constructors. English language has, for instance, 25 sentence construction pattern with the SPO pattern as the hyper-simplistic one.

1.2 Database Semiotics

Database semiotics and especially database semantics uses a strict form with exact matching of syntax and semantics, without context consideration, without state restrictions beside finiteness, with exact interpretation, with full evaluation of variables, and full elaboration. Database pragmatics is the study how languages are used for intended deployment functions in dependence on the purposes and goals within a community of practice. It is of interest for conceptualisation and conceptual modelling but not for database semiotics.

The variety of competing viewpoints within a community of practice might lead to a conglomerate of meanings. Database research has been considering semantic specification languages in the past. These semantic database modelling languages have, however, not found their way to implementations. A typical example is the simplification of the meaning of Is-A constructions.

Database semiotics does not consider the variety of abstraction level, i.e. user worlds and agreements among users in a community. Instead, the solution world of DBMS and the performance criterion have taken over. Database semiotics has been toggling between a rich structuring with less expressive power of semantics (e.g. object-relational languages such as the higher-order entity-relationship modelling language [20]) and poor structuring with the need for a rich semantical system (e.g. relational languages). Each of these languages have their own 'music'.

The main – sometimes the only one – quality characteristics for DBMS is storage and computation performance. For this reason, database semiotics is based on the harmonising container (or class) typing instead of object (or data) semiotics. Database semiotics has to compromise with (I) performance or richness, (II) full semantics or sufficient semantics, (III) stable semantics or robust semantics, (IV) static semantics or dynamic and evolving semantics, and (V) full service or fast and sufficient service.

1.3 The Special Case of Database Constraints

The theory of database dependencies and constraints is fairly rich (e.g. [7,17,19]). We know almost hundred classes of static integrity constraints. They can be categorised into constraints for partial identification of values, constraints declaring relative independence of objects, existence constraints, and redundancy constraints. Typically constraints are defined in languages at the conceptual or implementation layer. We might also consider user constraints.

	Identification	Independence	Existence	Redundancy
Conceptual layer	Functional, equality-generating	Multivalued, hierarchical, join, exclusion, Tuple-generating constraint, horizontal decomposition	NULL-marker-free, union constraint, numerical, cardinality	Inclusion, exclusion constraint
Implementation layer	Key, uniqueness, trigger, check	Decomposition, stored procedures, trigger	no NULL, stored procedures, trigger	Referential integrity, surrogate, capsules
User layer	Identification	Structuring!	no NULL	Elementary facts

The theory of database constraints and dependencies has mainly been developed in the 70ies, 80ies, and early 90ies. Later it has been considered to be a "dying swan" (DOOD91 conference talk). Many of the results from this time are forgotten. Some theories have later been re-developed using different notations and names, e.g. conditional functional dependencies [12]. The list of open problems [18,20,21] is fairly long and – surprisingly – relatively stable.

Static integrity constraints are typically realised as dynamic transition constraints, i.e. they support transition from one valid state to another valid database state. Beside transition constraints we might consider also temporal and general dynamic constraints.

1.4 The Storyline and Database Semiotics Research

We start first with a discussion of the state-of-art for database semiotics. Next we reconsider the body of knowledge for a small number of problems. Finally, we draw lessons. We discuss some of the results and refer to database research for a deeper discussion of the issues.

This paper is a keynote paper and thus presents our experience we gained from research that started already 40 years ago, reflects our thoughts and insights into the topic, and should stimulate future research. We select a narrative style for explanation of suggestions and recommendations.

We restrict the paper to database constraints. Further, we cannot survey the entire body of knowledge on database constraints. A detailed introduction and consideration of constraint classes would result in a full textbook (e.g. a revision and extension of [19]). We refer to [1,12,19,20] or the survey [7] for the definition of constraints. We consider in this paper only six main classes of constraints that are used in almost all textbooks: keys, functional dependencies, multivalued dependencies, inclusion constraints, exclusion constraints, and cardinality constraints.

2 Goodliness of Mathematical Logic

2.1 Components of (Meta-semiotic) Logics

The classical mould to specification follows the meta-semiotic approach

– The syntactic constituent defines which symbols are going to be used in the language, which combinations (word) of these symbols are allowed, and which constructors can be (partially) used. Syntax is typically constructed inductively
– The semantic constituent defines what is the purpose of the language, which structures are of interest, and what is going to be expressed.
– The two constituents are interrelated syntax and semantics. Mathematical logics uses syntax as firstness and semantics as secondness, i.e. truth values (or appropriateness) allow to define which structures or expressions are true or meaningful or potentially meaningful.
– The pragmatic constituent can define which meaning can be canonically assigned to words, which restrictions must be considered, and which closure operators are applicable.

This approach defines the playground of Mathematics Logics by starting with a signature of the language and structure and then by deploying construction rules for the construction of words (or expressions) of the language. Semantics in the Fregean and Quine sense canonically uses the same construction rules. Beside canonical semantics in mathematical logic we might also consider ideational, behaviourist, referential, possible-world, verificationist, truth-conditional, conceptual-role, and Gricean belief semantics. Computer science uses a number semantics beside canonical semantics [14]: lexical (e.g. Web 3.0), statistical, prototype, program, dynamic, operational, denotational, axiomatic, transformational, algebraic, and macro-expansion (e.g. λ calculus) semantics. These frames also follow the meta-semiotic approach. We might also use second-order construction of languages [23]. Logics can also be defined based on other meta-approaches, e.g. [8,11,22].

2.2 Forgotten and Supplanted Properties

Static database constraints are classically defined by expression that can be translated to expressions in first-order predicate logics.

Typical forgotten properties of constraints are the following ones: Hierarchical dependencies can be based on reasoning through the tableau calculus. They are not invariant for most operations. Similar observations can be made for universally quantified formulas. For instance, functional dependencies are preserved for join operations, are subset-invariant, are intersection-invariant, are restricted union-invariant, and are not complement-invariant. There is a semantical gap between functional and multivalued dependencies. Both can be considered to be structuring constraints. All specification languages use implicit language assumptions (e.g. component inclusion condition, identifiability for set semantics, cycles and finiteness assumptions). Normalization is considered to be structurally instead of behavioral optimization.

Supplanted properties in constraint specification are finiteness properties (finite implication, finite calculi, finite representation of potentially infinite sets, finite computation ('safety')), parallel execution (with transaction based concurrent semantics, causal semantics of processes, predicative semantics for engines) uncertainty of data (represented by NULL markers (overloaded, logics, computation, identification), fuzzy data with error models, aggregated macro-data instead of micro-data), completeness of specification, and weak constraints (e.g. deontic maintenance, temporal maintenance, default values).

Alternative definitions provide an insight to constraints. For instance, functional dependencies can be differently treated:

Quantity matrices: The functional dependency $X \longrightarrow Y$ can also be defined as an algebraic expression $|\pi_X(R^C)| = |\pi_{XY}(R^C)|$ for a relational class.

Equality sets: For a given relational class we consider the set of value-equivalence classes of object $E_{R^C}(A) = \{\{t' \in R^C | t =_A t'\} \,|\, t \in R^C\}$ for an attribute $A \in U_R$ and $E_{R^C}(X) = \{\{t' \in R^C | t =_X t'\} \,|\, t \in R^C\}$ for a subset of attributes $X \subseteq U_R$.

Lattices of closed sets: Instead of reasoning on functional dependencies we can construct the set of all closures X^+ on subsets X of attributes as a lattice $\mathbb{L}(R^C)$ of equality sets of a relation class R^C. It forms a *sub-cylindric* lattice $\mathbb{L}(\mathfrak{M}, \cap, \cup, \top, \bot, \leq)$ and thus can be considered as a lattice.

Pair algebra reasoning: Functional dependencies $X \longrightarrow Y$ can be represented by a pair (X, Y) with a specific order $(X, Y) \preceq (X', Y')$ if $X \subseteq X'$ and $Y \supseteq Y'$. The left and right sides of pairs form two lattices $(\mathcal{L}_1, +, \cdot, \leq)$ and $(\mathcal{L}_2, +, \cdot, \leq)$ which are interrelated by a Galois correspondence.

3 Revising Approaches to Constraints

3.1 Constraints Brilliance and Syntax Glory

Constraint set axiomatisation can be developed for each variant of syntax. For instance, there are Armstrong style axiomatisations for the basic relational modelling language and for its extensions by lists, by sets, by multi-sets, by multi-lists, and by powersets. Advanced modelling languages such as the higher-order entity-relationship modelling language have also an axiomatisation. All these axiomatisations are similar.

Based on pair algebra reasoning it is easy to prove the following theorem: General *deduction theory* for closed Horn formulas: A sound and complete axiomatisation for closed Horn formulas $\forall ...(\alpha \rightarrow \beta)$ consists of

axioms $\frac{}{\alpha \rightarrow \beta}$ for all facets of substructures $\beta \preceq \alpha$
augmentation rules for super/sub-structures $\frac{\alpha \rightarrow \beta}{\alpha' \rightarrow \beta'}$
for either $\beta' \preceq \beta$ and $\alpha \preceq \alpha'$ or as well as $\alpha' = \alpha \sqcup \gamma$ and $\beta' = \beta \sqcup \gamma$, and transitivity rules for all connecting pairs $(\alpha \rightarrow \beta,\ \beta \rightarrow \gamma)$.

The order $\preceq$ and the union $\sqcup$ are defined for the lattices of the pair algebra.

The simplest case for this theorem is the classical Armstrong axiomatisation for functional dependencies in the basic relational modelling language:

$$\text{Axiom} \quad X \cup Y \rightarrow Y$$

$$\text{Rules} \quad (1)\ \frac{X \longrightarrow Y}{X \cup V \longrightarrow Y \cup V} \qquad (2)\ \frac{X \longrightarrow Y\ ,\quad Y \longrightarrow Z}{X \longrightarrow Z}$$

The advantage of such Horn formula systems is the inversion of constraints to their excluded variants. For instance, an excluded functional dependency $X \longrightarrow / \ Y$ is valid for a collection of classes if the functional dependency $X \longrightarrow Y$ is not valid in one of the classes of the collection.

For instance, the transitivity rule may be rephrased for negations of constraints:

$$\frac{X \longrightarrow Y\ ,\quad Y \longrightarrow Z}{X \longrightarrow Z} \qquad\qquad \frac{X \longrightarrow Y\ ,\quad X \not\longrightarrow Z}{Y \not\longrightarrow Z}$$

The following Hilbert-type deductive system is sound and complete for functional dependencies and excluded functional dependencies.

$$\text{Axiom} \qquad X \cup Y \rightarrow Y$$

$$\text{Rules} \qquad (1)\ \frac{X \longrightarrow Y}{X \cup V \longrightarrow Y \cup V} \qquad (2)\ \frac{X \longrightarrow Y\ ,\quad Y \longrightarrow Z}{X \longrightarrow Z}$$

$$(3)\ \frac{X \longrightarrow Y\ ,\quad X \not\longrightarrow Z}{Y \not\longrightarrow Z} \qquad (4)\ \frac{X \not\longrightarrow Y}{X \not\longrightarrow Y \cup Z} \qquad (5)\ \frac{X \cup Z \not\longrightarrow Y \cup Z}{X \cup Z \not\longrightarrow Y}$$

$$(6)\ \frac{X \longrightarrow Z\ ,\quad X \not\longrightarrow Y \cup Z}{X \not\longrightarrow Y} \qquad (7)\ \frac{Y \longrightarrow Z\ ,\quad X \not\longrightarrow Z}{X \not\longrightarrow Y}$$

The deductive system is not minimal due to the union rule (6), the advancement rule (5), and the subset rule (4). It is, however, more convenient.

3.2 The Boon of the Propositional Logic

Propositional logic is far simpler than predicate logic. The reasoning procedures use a Boolean calculus. Propositional logics is best fitted to cases when the implication is based on a two-object property, i.e. two-object classes can be used as a counterexample for validity of a constraint. Propositional representation also allows to derive simple counterexamples for a set of constraints.

Functional dependencies $X \longrightarrow Y$ for $X = \{A_1, ..., A_k\}$, $Y = \{A_{k+1}, ...A_l\}$ can be represented by a Boolean implication $p_1 \wedge ... \wedge p_k \rightarrow p_{k+1} \wedge ... \wedge p_l$ or by a set of implications with a singleton right side $p_X \rightarrow p_j$ for $p_X := p_1 \wedge ... \wedge p_k$ and $k+1 \leq j \leq l$. A functional dependency follows from a set of functional dependencies if and only if this is valid for the Boolean implications.

Another propositional approach can be used for full multivalued dependencies $X \twoheadrightarrow Y|Z$ by a Boolean implication $p_X \rightarrow p_Y \vee p_Z$. W.l.o.g. we may assume $Y \cap Z = \emptyset$. Functional and multivalued dependencies can be represented together using these forms. The implication problems for sets of these constraints and for sets of Boolean implications are equivalent.

This propositional representation is also valid for generalised types of functional dependencies like the Hungarian functional dependencies. This approach can also be used for handling of functional dependencies with one of their negated forms (i.e. either afunctional dependencies or excluded functional dependencies). This representation is however an either-or form since we change semantical meaning of the constraints. Hungarian functional dependencies that are not systems of functional dependencies and multivalued dependencies use also different meaning and cannot be handled this way.

This Boolean representation of constraints allows to derive complexity results. For instance, the maximal size of a system of minimal keys in relational types with n attributes or components is $\binom{n}{\lfloor \frac{n}{2} \rfloor}$. A similar exponential boundary can be derived for the maximal size of independent functional dependencies: $\lceil \frac{n}{2} \rceil \binom{n}{\lfloor \frac{n}{2} \rfloor}$ [5].

Minimal keys form a Sperner system on $attr(R)$, i.e. $X \not\subseteq Y$ for all minimal keys X and Y. The set of antikeys is the set of maximal subsets of $attr(R)$ which are not keys. Antikeys and minimal keys characterise each other. The antikey characterisation can be used for construction of Armstrong relations which obey all minimal keys and only those. The set of all minimal keys is represented by a monotone Boolean formula.

Formulas of first-order predicate logics that are generalisations can be represented by an open first-order predicate logics without quantifiers. For this reason, these formulas may also be represented by Boolean formulas.

3.3 The Boon and Bane of First-Order Predicate Logic

Almost all classes of static database constraints can be expressed as tuple-generating or equality-generating formulas in first-order predicate logics. For instance, equality generating dependencies are constraints of the following form:

$$\forall(x_{1,1},...,x_{m,n})\ ((P_R(x_{1,1},...,x_{1,n}) \wedge ... \wedge P_R(x_{m,1},...,x_{m,n}) \wedge F(x_{1,1},...,x_{m,n})) \longrightarrow G(x_{1,1},...,x_{m,n}))$$

where $F(x_{1,1},...,x_{m,n}), G(x_{1,1},...,x_{m,n})$ are conjunctions of equalities of the form $x_{i,j} = x_{i',j'}$ and P_R is the predicate symbol associated with R.

Full tuple-generating dependencies are constraints

$$\forall(x_{1,1},...,x_{m,n})((P_R(x_{1,1},...,x_{1,n}) \wedge ... \wedge P_R(x_{m,1},...,x_{m,n}) \wedge F(x_{1,1},...,x_{m,n})) \longrightarrow (P_R(x'_{1,1},...,x'_{1,n}) \wedge ... \wedge P_R(x'_{k,1},...,x'_{k,n})))$$

where $x'_{i,j} \in \{x_{1,j},...,x_{m,j}\}$ for $1 \leq j \leq n$, $1 \leq i \leq k$.

Temporal and dynamic constraints can be specified on extensions of these logics. Therefore, this logic seems to be sufficient for reasoning on constraints. This kind of reasoning is supported by the CHASE procedure for full tuple-generating and equality-generating formulas [1]. It seems thus that deduction is thus well-founded.

One of the main negative results in database theory is the theorem on non-axiomatisability of functional dependencies and inclusion dependencies:

"For each k there is a database schema $\mathcal{DB}$ for which there is no k-ary[1] *sound and complete axiomatization for finite implication of functional dependencies and inclusion constraints over $\mathcal{DB}$."* (e.g. [1], page 205)

This theorem has however a definitional flaw. In reality the statement must be read as follows:

There is no finite $\underline{\text{Hilbert-type}}$ axiomatisation in $\underline{\text{first-order predicate logics}}$ for functional dependencies and inclusion constraints.

We could ask whether there exist a Gentzen-type axiomatisation or whether we have to stick to first-order predicate logics. [9] could prove that there is a rather simple axiomatisation in a weak second-order logics for these two classes of constraints. Join dependencies are a typical example of non-Hilbert-style axiomatisability but Gentzen-style axiomatisability.

First-order predicate logic has been used for a brute-force definition of a relational query language. A formula with free variables might be considered as a relational tuple calculus query. The result can be a finite but also an infinite set of tuples. Databases and queries must, however, be finite. This finiteness restriction results in a different semantics. The tuple calculus is considered to be a genuine piece of junk: safe formulas which safety property cannot be recursively decided. If we use a strongly typed first-order logic [2] then unsafe formulas cannot be declared.

[1] The number k is the maximal number of premises in a Hilbert-type deduction rule.

A third flaw is the difference between set semantics that is used for the relational database model and the multi-set semantics that is used in DBMS. The identification property and the uniqueness property are different for multi-set semantics. A fourth crap is the identification requirement for object-oriented databases [3].

3.4 Constraint Classes or Real-Application Constraints

The constraint research has been concentrating on a separation of concern by classification into constraint classes. The deduction of constraints is handled according to the constraint class. In order to provide a homogenised handling for all objects in a class, we target on classes which have a small and easy to manage class of constraints. This classical approach has been used for most DBMS which supported keys, specific inclusion constraints (referential constraints), and domain constraints. The normalisation of structures lead to a Salami-slice approach in which the semantics for all classes uses only these constraints.

Real-application constraints typically co-occur in a structure, i.e. we should consider the constraints for a given class instead of constraint classes in general. We might use a unit-oriented approach. A semiotic and pragmatistic unit is defined by its structure and its constraints.

The component-oriented approach uses such semiotic components with specific specialisations and generalisations form a component network. This separation of semantics to components results in a 'Venetian glassware' representation instead of Salami slice representation. A similar uni-oriented approach is based on pivoting of constructs in extended ER approaches (HERM)[20]. Sets of functional dependencies can be handled by an constraint equivalence reasoning. For instance, we define an abstraction $[A, B, C]$ for $\{A \to B, B \to C, C \to A\}$ and represent the attributes A, B, C by their abstraction. The set of functional dependencies becomes simpler. Additionally we avoid the normalisation trap by equivalence set reasoning.

Reasoning by abstraction on attributes is far simpler than reasoning on basic attributes. For instance, the set of functional dependencies $\{AB \to C, C \to [A, D, E]\}$ with $\{A \leftrightarrow D \leftrightarrow E\}$ has 45 canonical minimal covers $\{[A, D, E]B \to C \to [A, D, E]\}$. So, selecting a good normalisation variant becomes an art.

Another reasoning system can be developed for an integrated development of sets of constraints. For instance, types with a smaller component set allow a holistic treatment of the entire constraint set instead of reasoning constraint by constraint [6]. This reasoning system allows to reason on the impact a constraint has. For instance, the rule

$$\frac{C \to A \quad \text{supported by} \quad A \cup C \to B}{C \to B}$$

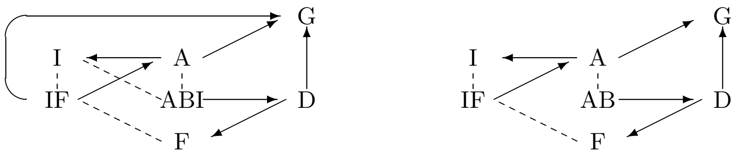
states that one constraint which is supported by another constraint allows to conclude a third constraint. This rule explains transitivity of constraints.

4 Calculi Beyond Classical Mathematical Logics

4.1 Visual Reasoning

Calculi must not necessarily be logical ones. They can also be graphical [16]. A typical example is the graphical representation of functional dependencies by edges from subsets of the set of attributes to singleton attributes at the right side. These graphs may be enhanced by subset edges (dotted) for a simpler detection which subsets are contained in another one.

Let us consider a simple example: given attributes $U_R = \{A, B, D, F, G, I\}$ and a set of functional dependencies $\Sigma_R = \{A \longrightarrow IG, D \longrightarrow FG, IAB \longrightarrow D, IF \longrightarrow AG\}$. This set can be represented by the graph on the left side of the following picture. This set can be reduced by deleting $IF \longrightarrow G$ from the graph since it is derivable through the edges representing $IF \longrightarrow A$ and $A \longrightarrow G$. Furthermore, the set ABI can be reduced since the edge representing $A \longrightarrow I$supports subset reduction. No other reduction can be applied to the graph. The calculus for graphical reasoning [6] is complete.



We may directly derive a normalisation according to this graph reduction. Each constraint must be covered. We arrive with the synthesis algorithm to:

$R_1 = (\{A, G\}\ ,\ \{A \longrightarrow G,\ R_1[A]\ \subseteq\supseteq\ R_2[A],\})\ ,$

$R_2 = (\{A, F, I\}\ ,\ \{A \longrightarrow I, FI \longrightarrow A,\ R_2[F]\ \subseteq\supseteq\ R_4[F]\})\ ,$

$R_3 = (\{A, B, D\}\ ,\ \{AB \longrightarrow D,\ R_3[D]\ \subseteq\supseteq\ R_4[D],\ R_1[A]\ \subseteq\supseteq\ R_3[A]\})\ ,$

$R_4 = (\{D, F, G\}\ ,\ \{D \longrightarrow FG,\ R_1[G]\ \subseteq\supseteq\ R_4[G]\})\ .$

Classical normalisation would result in another normalised relation type:

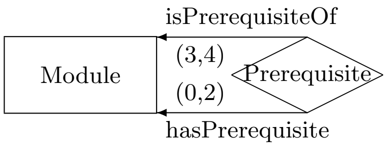
$R_1' = (\{A, G, I\}\ ,\ \{A \longrightarrow GI,\ R_1'[AI]\ \subseteq\supseteq\ R_2[AI]\}).$

4.2 Numerical Calculi

Cardinality constraints of the form (0,1), (1,1), (1,n) essentially express functional dependencies (through upper bound restrictions (0,1), (1,1)) and inclusion constraints (through lower bound restrictions (1,1), (1,n)). For this reason, the axiomatisation result discussed above applies also to cardinality constraints. Therefore, logical calculi have either to use weak second-order calculi or to turn to other kinds of calculi. The simplest calculus is in this case a numerical one.

Consider, for instance the following simple schema with the cardinality constraints: card(Prerequisite, hasPrerequisite) = (0,2)

and card(Prerequisite, isPrequisiteOf) = (3,4).



This schema is inconsistent for non-empty finite object sets. Its inconsistency can directly be concluded with the numerical calculus [20]:

$$\frac{3 * |M| \leq |R| \, , \; \frac{1}{2} * |R| \leq |M|}{\frac{3}{2}|M| \leq 1|M| \quad \lightning}$$

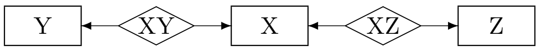
This calculus allows to derive two repairs: (3,4) $\curvearrowright$ (2,y) (for $y \geq 2$) or
(0,2) $\curvearrowright$ (x,3) (for $x \geq 0$).

It also allows to derive weaker systems as long as the conclusion is weaker than the premise.

4.3 Structural Reasoning

Multivalued dependencies $X \twoheadrightarrow Y|Z$ on a relational type R and the corresponding R-classes R^C are defined as generalisations of functional dependencies by a condition: for all $t, t' \in R^C$ with $t =_X t'$ exists an object $t'' \in R^C$ with $t'' =_{X \cup Y} t$ and $t'' =_{X \cup Y} t'$ where $X \cup Y \cup Z = attr(R)$.

This mathematical definition is difficult to handle. Four other equivalent definitions are based on a decomposition property, on independence of Y-values from Z-values for each X-value, on product constructor, and on non-first-order structuring of R. The simplest equivalent definition is based on a separation of concern into an XY and an XZ concern in the entity-relationship modelling language.



4.4 Divide and Conquer

Constraints have also a pragmatical meaning that should be considered. For instance, functional dependencies $X \longrightarrow Y$ have their own structural and pragmatical meaning:

Explicit declaration of partial identification: The functional dependency explicitly declares a identification of Y-values through X-values.

Tight functional coupling: There exists potentially a function from X-values to Y-values. It is a specific cardinality constraint.

Semantic constraint specific for the given application: The functional dependency is specific for the given application and thus stronger than in other applications.

Semantical or structural unit with functional coupling: The X–Y association forms a semantical units and can thus be represented in a separate form.

The axiomatisation for functional dependencies can be refined according to these meanings. For instance, tight functional coupling hints on pivoting. Semantical and structural units can be considered on their own.

Inclusion, exclusion, and domain constraints have also a pragmatical meaning. Classical normalisation theory does not consider this meaning. The result of a normalisation process might result in inappropriate solutions that destroy the associations with more important pragmatical meaning.

4.5 Exception Handling

Exception handling is one of the great lacunas in computer science research. We may approach exception problems by a separation of concern approach [4,10]. There are five types of exceptions for database management: *errors*, *incompleteness*, *insufficiency*, *deviation from normality* (e.g. *dynamic changes* or *hidden cases*), and *operational nondeterminism*. The management of exceptions can be based on (i) conceptualisation of solutions to exceptions, (ii) on enhancement of conceptual schemata by exception templates, and (iii) on development of control and measurement practices. A number of exception handling mechanisms on the basis of constraints and macro-constraints have been realised: (a) multi-layering of applications separates objects in a good state, bad state, and exceptional state; (b) state-based separation applies special macro-states (normal state, intermediate states, final states, etc.) (c) business rule injection into exception handling is based on extended transaction or trigger models; (d) automatic exception containers use orthogonal 'ghost' schemata based on horizontal decomposition; (e) multi-shell exception handling is based on explicit shell modelling; (f) tolerance/toleratability with provenance allow temporary constraint set violations.

NULL markers (originally called NULL values) are one kind of exceptions [15]. NULL is a special marker used in SQL to indicate that a value for an attribute of an object does not exist in the database. There are many reasons for this non-existence: the value is currently unknown but exists; the attribute is not applicable for the object; the value does not exist at all for the given object; the value never exists for this object; ... The SQL standard assigns for a comparison of an existing value to a NULL marker the value `unknown` what is completely counterintuitive. For instance, if a NULL marker has a meaning that a value never exists then a comparison does not make sense. In this case, the logical value is neither `false` nor `true`. It is worse than `false`, i.e. it is `nonsense`. The corresponding logics must be revised by redefining logical connectives (negation, conjunction, disjunction, etc.) and quantifiers. The logics used becomes then a paraconsistent. The notions of keys, functional dependencies, etc. must be redefined too.

4.6 Viewpoint-Oriented Treatment of Semiotics

Classical DBMS handle constraint satisfaction on the basis of the global (or central) data. Views are derived by queries applied to these data. Views might however also be interrelated. Therefore, view might form their own viewpoint schema. These viewpoint schemata are often the basis for database interfacing. Instead of handling constraints at the basic data level, we can separate constraint sets into those that can be handled and supported at the viewpoint level and those that must be enforced in the global database. This approach results in a better computational behaviour of the system since constraint enforcement is often a performance trap. This separation of enforcement can be refined by a separation into declarative constraints that are supported by DBMS structuring and into complexes of procedural constraints that are supported by triggers or stored procedures.

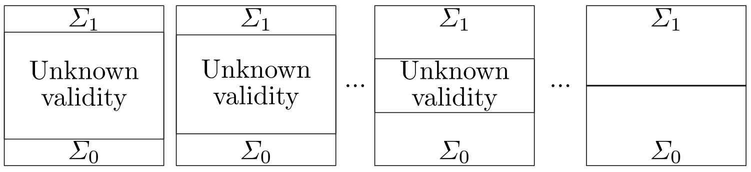
4.7 Constraint Acquisition and Negated Constraints

Most optimisation algorithms and especially normalisation assume fully defined semantics. All potentially valid constraints of a given kind must either be derivable from a given constraint set or explicitly marked as not valid for the given application case. This approach is neither feasible nor realistic for real applications since the size of a set of logically independent constraints can be exponential for the number of type components (attributes). Types with 20 attributes require therefore a procedure for constraint check that is far beyond time constraints.

One approach might be the development of a robust constraint specification that does not consider all potentially valid constraints but only essential ones. The separation by pragmatical meaning and the orientation on those constraints which validity must be guaranteed is one option.

This option is supported by results on the average complexity of constraint sets [20]. For instance, the length of a minimal key in an average class is bounded by a multiple of the logarithm on the number n of attributes where the basis of the logarithm depends on the size of the domains for the attributes. This size k is less then $\lfloor \frac{n}{2} \rfloor$ for larger n. The number of minimal keys is then about $\binom{n}{k}$.

Constraint acquisition is heuristically organised by considering constraints which validity and invalidity has the most influence of potentially derivable or rejectable constraint. We can simultaneously consider the set of valid constraints Σ_1 and of invalid constraints Σ_0.



Initial step Intermediate steps Final step

For instance, for functional dependencies the procedure would be then the following one:

1. Basic step: Design obvious constraints.
2. Recursion step: Repeat until the constraint sets Σ_0 and Σ_1 do not change.
   - Find a functional dependency α that is neither in Σ_1 nor in Σ_0.
     - If α is valid then add α to Σ_1.
     - If α is invalid then add α to Σ_0.
   - Generate the logical closures of Σ_0 and Σ_1.

5 Lessons to Learn and to Consider

5.1 Holistic Treatment of Syntax and Semantics

Natural languages use a holistic approach to semiotics. Some languages (e.g. German) allow word-based interpretation. Others (e.g. English) also use a context-biased interpretation. Some languages follow a rule-based approach (e.g. Indo-European with verb-based sentence structure) while others (e.g. Semitic, Japanese and other Asian languages) use a network approach.

Computer Science uses formal languages which definitional frame reuses the one from Mathematical Logics, i.e. syntax is defined first and then semantics is defined based on the syntax structure. Peirce considers syntax as some kind of "firstness" property and semantics as some kind of "secondness". Pragmatics and pragmatism are neglected. This approach simplifies compiler and interpreter construction.

We observed however that structuring and interpretation cannot be separated for applications. They are interleaved. Due to the classical approach, database constraints are often difficult to handle, e.g. multivalued dependencies (wrong specification language; sixth (and fifth or fourth) definition are appropriate). Constraint maintenance might become a nightmare and becomes an incubus with big data. Normalisation theory is based on doubtful assumptions and has become obsolesce for object-relational structures. Optionality (e.g. NULL markers) is not properly understood and supported. Moreover, semantics is also dynamic.

5.2 Maintenance: From Ugly Duckling to Beautiful Swan

Constraint maintenance is based on answering the following question: Provided the database $\mathcal{DB}$ is in a consistent state specified by the integrity constraints Σ before running the transition and given a transition TA, what is the best solution to the requirement that the application of the transition t to the database $\mathcal{D}$ does not invalidate the constraints in Σ?

SQL and DBMS systems provide a number of mechanisms for constraint maintenance: (A) refusal of modification by the transition TA (e.g. no action or forbidden action); (B) transformation of the transition TA to a correct one (by repair actions, by transformation to the greatest consistent specialisation [13], or by effect preserving specialisations); (C) reduction of the transition; (D) repairing the database (by terminating, confluent, and effect preserving triggers); (E) normalisation approaches; (E) modification of interfaces (e.g. on the basis of stored procedures); (F) maintenance layering inside the database; and (G) hybrid approaches (e.g. restricting the application domain of operations and of modifications).

A general constraint handling framework supports validity of constraints at different levels:

1. At the specification level: the description of the constraints properties is enhanced by validation, policies for evaluation, specific policies, and transformations of constraint properties to others constraints.
2. At the control or technical level: The application of the constraint, of the constraint property portfolio, of techniques and of methods is added.
3. At the application or technology level: The management of constraint handling is embedded into the DBMS management.
4. The establishment or organisational level is based on a methodology and constraint maintenance system.

Level 5 adds facilities for handling satisfaction of constraints and for predicting changes of satisfaction. Level 6 optimises the constraint management. Level 7 uses experiences for modernisation, modification and adaptation of constraint handling.

5.3 Lessons to Consider

Finally, we conclude:

+ Hierarchical structuring or inductive composition of types leads to a generalized first-order predicate logics. It is simple but might be too simple for applications.

+− First-order axiomatisability: A deductive Hilbert-type system exists if and only if the implication operator is reflexive, monotone, closed, and compact.

+− Functional, key, inclusion and exclusion dependencies are constraints that are natural in the relational model but not natural for object-relational database models. Key-based inclusion dependencies are handicapped for more complex structuring. Multi-valued dependencies are far better expressed in the ER modelling language. Cardinality constraints are overloaded. We should treat maximal and minimal cardinalities in separate systems. Join dependencies are mainly representational constraints.

−− Combinatorial problems might limit the use and the usability of constraints. Constraint specification is thus often incomplete. Since normalisation theory requires completed constraint specification it is useful only for simple cases.

−+ Logical reasoning should be combined with other calculi such as graphical and numerical ones.

We also arrive with a number of lessons:

Rigidity of validity: Some integrity constraints are very important in an application area. Others are less important. Users can often "live" with temporary violations of the latter. Soft constraints are constraints whose satisfaction is desirable, but not essential.

Behaviour in presence of NULL markers: NULL markers carry a variety of different semantics. Most constraints are not defined on null values. The behaviour of some types of constraints such as functional dependencies becomes cumbersome if null values are permitted.

Weakening validity temporarily: In the daily operation of a database exceptions may arise due to various reasons. In some cases a constraint may be allowed to be invalid within a time interval. Weakening validity may be supported by special extensions transaction and temporary classes.

Enforcement time: Validity of constraints may be enforced at different points of time. This situation has been taken into account to some extent. For instance, SQL:1999, allows one to specify that constraints are to be enforced whenever a tuple that might violate the constraint is modified, or at the end of the transaction, or based on the occurrence of some events. But the consistent management of constraint enforcement time is still an open problem.

Partial satisfaction of constraints: Constraints may be partially or totally satisfied. We may collect all those objects for which a constraint is not satisfied into a separate database unit.

Execution time deadlines: Constraints may be violated due to the late arrival of data or events. A contingency plan or contingency transactions may be invoked with guaranteed execution time characteristics.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. AlBdaiwi, B., Thalheim, B.: Revisiting the definition of the relational tuple calculus. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. CCIS, vol. 539, pp. 3–11. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23201-0_1
3. Beeri, C., Thalheim, B.: Identification as a primitive of database models. In: Proceedings FoMLaDO 1998, pp. 19–36. Kluwer, London (1999)
4. Berztiss, A., Thalheim, B.: Exceptions in information systems. In: Digital Libaries: Advanced Methods and Technologies, RCDL 2007, pp. 284–295 (2007)
5. Demetrovics, J., Katona, G.O.H., Miklós, D., Thalheim, B.: On the number of independent functional dependencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 83–91. Springer, Heidelberg (2006). https://doi.org/10.1007/11663881_6
6. Demetrovics, J., Molnár, A., Thalheim, B.: Graphical reasoning for sets of functional dependencies. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 166–179. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30464-7_14

7. Godfrey, P., Grant, J., Gryz, J., Minker, J.: Integrity constraints: semantics and applications. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, pp. 265–306. Springer, Boston (1998). https://doi.org/10.1007/978-1-4615-5643-5_9
8. Gurevich. Y.: On Kolmogorov machines and related issues. In: Current Trends In Theoretical Computer Science: Essays and Tutorials, pp. 225–234. World Scientific (1993)
9. Hannula, M., Kontinen, J.: A finite axiomatization of conditional independence and inclusion dependencies. In: Beierle, C., Meghini, C. (eds.) FoIKS 2014. LNCS, vol. 8367, pp. 211–229. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04939-7_10
10. Jaakkola, H., Thalheim, B.: Exception-aware (information) systems. In: Information Modelling and Knowledge Bases, vol. XXIV, pp. 300–313. IOS Press (2013)
11. Kolmogorov, A.N., Dragalin, A.G.: Mathematical Logics. KomKniga, Moscov (2006). (in Russian)
12. Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model. Springer, Heidelberg (1989)
13. Schewe, K.-D.: Fundamentals of consistency enforcement. In: Information Modelling and Knowledge Bases X, vol. 51. Frontiers in Artificial Intelligence and Applications, pp. 275–291. IOS Press (1998)
14. Schewe, K.-D., Thalheim, B.: Semantics in data and knowledge bases. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 1–25. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88594-8_1
15. Schewe, K.-D., Thalheim, B.: NULL value algebras and logics. In: Information Modelling and Knowledge Bases, vol. XXII, pp. 354–367. IOS Press (2011)
16. Sörensen, O., Thalheim, B.: Semantics and pragmatics of integrity constraints. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2011. LNCS, vol. 7693, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36008-4_1
17. Thalheim, B.: Bibliographie zur Theorie der Abhängigkeiten in relationalen Datenbanken, 1970–1984. Technical Report 566/85, TU Dresden (1985)
18. Thalheim, B.: Open problems in relational database theory. Bull. EATCS **32**, 336–337 (1987)
19. Thalheim, B.: Dependencies in Relational Databases. Teubner, Leipzig (1991)
20. Thalheim, B.: Entity-Relationship Modeling - Foundations of Database Technology. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-662-04058-4
21. Thalheim, B.: Open problems of information systems research and technology. In: Kobyliński, A., Sobczak, A. (eds.) BIR 2013. LNBIP, vol. 158, pp. 10–18. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40823-6_2
22. Uspensky, V.A.: Kolmogorov and mathematical logic. J. Symbolic Logic **57**(2), 385–412 (1992)
23. Van Wijngaarden, A.: Orthogonal Design and Description of a Formal Language. MR 76, October 1965

Fully Generic Queries: Open Problems and Some Partial Answers

Dimitri Surinx, Jan Van den Bussche(✉), and Jonni Virtema

Hasselt University, Hasselt, Belgium
jan.vandenbussche@uhasselt.be

Abstract. The class of fully generic queries on complex objects was introduced by Beeri, Milo and Ta-Shma in 1997. Such queries are still relevant as they capture the class of manipulations on nested big data, where output can be generated without a need for looking in detail at, or comparing, the atomic data elements. Unfortunately, the class of fully generic queries is rather poorly understood. We review the big open questions and formulate some partial answers.

1 Introduction

For the task of querying a database, database systems offer a database query language. Unlike a general-purpose programming language, a database query language allows us to formulate queries on the logical level of the data model on which the database system is based. Working on that higher, logical level has many advantages. First, programs can be correct independently of how the data is physically stored. Moreover, it makes it easier for the database query processor to recognize "tractable" parts of queries that can be processed more efficiently. Tractability here can mean many things: a selection on an attribute for which an index is available; a join operation for which a specific algorithm can be used; and so on [17].

Historically, the logical nature of database queries was explicated independently by several researchers [5,8,23]. In the relational data model, a database instance I is viewed as a relational structure. Moreover, a query Q may involve additional predicates and functions on the atomic values that can appear in I. For example, consider the following SQL query over relations $R(A, B)$ and $S(C)$:

```
select A from R, S where B <= C
```

This query can be applied to instances that are logical structures involving, in addition to the database relations R and S, a less-than relation $\leq$.

The logical nature of queries then amounts to the general principle that a database query commutes with permutations of the atomic values that preserve all the relations over which the query is formulated. Such permutations are nothing else than isomorphisms of the relational structures. Thus, concisely, the principle can be stated as follows: *if Q is a query, I is an instance, and f is*

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 20–31, 2019.
https://doi.org/10.1007/978-3-030-32065-2_2

an isomorphism, then $Q(f(I)) = f(Q(I))$. Chandra and Harel [16] later called this principle "the consistency criterion" for database queries. The principle became finally known under the name *genericity* [2,20]. Genericity can be easily adapted to data models other than the relational model, simply by adopting the appropriate notion of an isomorphism. Interestingly, genericity coincides with the definition Tarski proposed in 1966 of "logical notions" [25].

Commuting with isomorphisms is thus a property expected from all database queries, even the most complex ones. Simpler queries, however, may have stronger commutation properties. Well known, for example, is the class of queries that can be formulated without invoking the equality predicate; these queries can be characterized as those commuting not only with all isomorphisms, but more generally with all strong surjective homomorphisms [15].[1]

One the most stringent commutation property of queries one can consider was proposed by Beeri, Milo and Ta-Shma under the name of *full genericity* [9]. Recall that a query Q is generic in the classical sense if, for every instance I and permutation f of atomic values, we have $Q(f(I)) = f(Q(I))$. Now a query Q is called fully generic if the same holds for all functions f from atomic values to atomic values; so f does not need to be a permutation.

In order to get a feeling for full genericity, let us consider the operations of the relational algebra. Union, projection and cartesian product are fully generic, but selection, intersection and difference are not. More generally, one can develop the intuition that the fully generic database queries are those that combine or restructure the data without really having to look at the concrete data values. In particular, a fully generic query is not sensitive to the presence of duplicates in the input. Fully generic queries may produce a lot of output, but the amount of processing relative to the output size is typically minimal. For example, consider a Big Data setting, where data is distributed over different compute nodes. When performing a join, we need to ensure that joinable tuples reside on a common node [4]. For fully generic queries, however, no such requirement seems to be necessary.

A form of full genericity is also found in provenance queries, which track or propagate provenance annotations from the input to the output. Semantic characterizations of provenance queries [12] involve the property that annotations can be copied (or omitted), but are not to be compared with each other. Such a property is similar to full genericity, but applied only to the annotations.

In the relational data model, full genericity is a rather poor notion. Indeed, if we fix the input database schema and the output relation schema, there are only finitely many different fully generic queries. All we can do is form cartesian products of projections of database relations that produce results of the right output width, and take unions of these. When moving to the complex-object data model, however, the situation changes.

[1] Madelaine [22] has given a complete overview of the classes of morphisms corresponding to classes of queries expressible in different fragments of first-order logic, formed by the possible combinations of allowed features among existential quantification, universal quantification, conjunction, disjunction, negation, and equality.

The complex-object data model is a generalization of the relational data model. A relational database instance is essentially a tuple of relations, where each relation is a set of tuples of atomic values. The width of the database instance, i.e., the number of relations, and the widths of these relations, are given by the database schema. Note that the way tuple and set formations can be nested is completely fixed in the relational model: we have a tuple of sets of tuples of atomic values, nothing more, nothing less. Now moving to the complex-object model [2], we are allowed arbitrary combinations of tuple and set formation. Complex objects have been around since the early 1980s, and remain relevant for modern database systems. For example, the data model underlying Apache Spark [7,26] is essentially that of complex objects, and also JSON databases such as MongoDB essentially store complex objects [11].

When thus considering queries over complex objects, where the types of inputs and outputs can be nested more and more deeply, the class of fully generic queries grows substantially. For a simple example, let k be a natural number. Given as input a set of atomic values, we may output the set of all subsets of the input having at most k elements. Each value of k gives rise to a different query, and all of these queries are fully generic. Note that all these queries have the same signature $\{d\} \rightarrow \{\{d\}\}$, i.e., they take as input a set of atomic values and they output a set of sets of atomic values (the symbol d stands for the atomic value type).

The goal of this paper is to draw attention to the fascinating class of fully generic queries. We find this class indeed fascinating because, while full genericity is a very stringent requirement, we still do not understand it well, especially when input and output types are deeply nested. What exactly are the fully generic queries? Some very basic questions about them remain unanswered:

1. Is every fully generic query effectively computable?
2. Can we effectively decide, given an input–output pair (A, B) of complex objects, whether there exists a fully generic query that maps A to B?
3. Is there a query language that captures the fully generic queries? (I.e., a language in which only fully generic queries can be expressed, but that is also complete, in that every computable fully generic query can be expressed.)

In contrast, the corresponding questions for classical generic queries have readily available answers: obviously negative for the first; affirmative for the second (just check if every automorphism of A is also an automorphism of B [8,23]); and again affirmative for the third [16].

Question 3 originates from Beeri, Milo and Ta-Shma [9], who proposed as a candidate language the classical powerset algebra for complex objects [1,21], from which we remove equality testing, and to which we add the intriguing operator *one-each*. Ta-Shma shows in her PhD thesis [24] that the resulting language, called $\mathcal{L}$, indeed captures the fully generic queries in two special but interesting cases: the case where the output is a flat relation, and the case where the signature is $\{\{d\}\} \rightarrow \{\{d\}\}$. Unfortunately, the given arguments are hard to verify. We will be able to offer a more transparent proof of the first result, and also of the special case $\{d\} \rightarrow \{\{d\}\}$. The language $\mathcal{L}$ also prompts various further interesting questions, some already posed by Ta-Shma, which will be expounded below.

2 Complex Objects, Queries, and Genericity

In order to discuss the issues presented in the Introduction more formally, we start by defining the complex-object data model, the notion of a query, and the notions of classical and full genericity.

A *type* is an expression τ conforming to the following grammar:

$$\tau ::= d \mid [\tau, \dots, \tau] \mid \{\tau\},$$

where d is a fixed symbol denoting the atomic value type. So, a type is either d, a tuple of types, or of the form $\{\tau\}$ with τ a type.

We assume, as given, a countably infinite domain **dom** of atomic values, or *atoms* for short. In examples, we will often use natural numbers for atoms. The set of *objects* is the smallest set such that

- every atom is an object;
- every tuple $[o_1, \dots, o_k]$ of objects is an object; and
- every finite set $\{o_1, \dots, o_n\}$ of objects is an object.

We will work only with well-typed objects. Informally, these are objects where in every set, all its elements are of the same kind. For example, the set $\{1, [2], [1, 2], \{1, 2, 3\}\}$ is very badly typed: its four elements are all of different kinds (an atom, a one-tuple, a two-tuple, and a set of atoms, respectively). Formally, an object o is said to be *of type* τ if one of the following holds:

- τ is d and o is an atom;
- τ is a tuple type $[\tau_1, \dots, \tau_k]$ and o is a k-tuple $[o_1, \dots, o_k]$ with o_i of type τ_i for $i = 1, \dots, k$;
- τ is a set type $\{\tau'\}$ and o is a finite set of objects of type τ'.

We will denote the set of objects of type τ by $[\![\tau]\!]$.

For types τ_{in} and τ_{out}, we now define a *query* of signature $\tau_{\text{in}} \to \tau_{\text{out}}$, quite simply, to be a total function q from $[\![\tau_{\text{in}}]\!]$ to $[\![\tau_{\text{out}}]\!]$. We will denote this by $q : \tau_{\text{in}} \to \tau_{\text{out}}$. We say that q is

- *generic* if we have $q(f(D)) = f(q(D))$, for every object D of type τ_{in} and every permutation f of **dom**.
- *fully generic* if we have $q(f(D)) = f(q(D))$, for every object D of type τ_{in} and every function $f : \textbf{dom} \to \textbf{dom}$.

Here, by $f(D)$, we naturally mean the object obtained from D by replacing each atom x by the atom $f(x)$.

The notion of genericity is classical and, for extensive discussion and motivation, we refer to the literature cited in the Introduction, and also to the work by Abiteboul and Vianu on generic computation [3] and by Blass, Gurevich and Shelah on Choiceless Polynomial Time [10], on which quite a bit of recent follow-up work has been performed [18].

In order to get a feeling for full genericity, let us look at two simple examples. First, consider the query $q : \{[d, d]\} \to \{[d, d, d]\}$ that takes as an input a binary relation of atoms. It outputs the ternary relation obtained from the input by swapping the two columns, and duplicating the second column in a third column. So, formally, $q(D) = \{[y, x, y] \mid [x, y] \in D\}$. We may view q as the projection operator $\pi_{2,1,2}$ from relational algebra. This query is readily verified to be fully generic:

$$\begin{aligned} q(f(D)) &= \{[y, x, y] \mid [x, y] \in f(D)\} \\ &= \{[f(v), f(u), f(v)] \mid [u, v] \in D\} \\ &= f(q(D)). \end{aligned}$$

In contrast, the query $q : [\{d\}, \{d\}] \to \{d\}$ that takes as an input two sets of atoms, and outputs their intersection, is not fully generic. Indeed, just consider the input $D = [\{1\}, \{2\}]$. Then $q(D)$ is empty, so also $f(q(D))$ is empty for any f. However, for f that maps both 1 and 2 to 1, we obtain $q(f(D)) = q([\{1\}, \{1\}]) = \{1\}$, which is nonempty.

3 Computability

The standard notion of computability, through Turing machines, is only defined as such for functions from Σ^* to Σ^*, for some finite alphabet Σ. Consequently, computability of queries needs a proper definition [2], which we recall next.

We first need to fix some encoding of atoms as strings; as usual, binary strings will suffice. So, assume some bijection $enc : \mathbf{dom} \to \{0, 1\}^*$. We can now consider the finite alphabet obtained by extending $\{0, 1\}$ with the punctuation symbols needed to write down objects: the comma, the square brackets, and the curly brackets. So, $\Sigma = \{0, 1\} \cup \{\texttt{,}, \texttt{[}, \texttt{]}, \texttt{\{}, \texttt{\}}\}$. We can similarly consider the *infinite* alphabet Λ obtained by adding these punctuation symbols to **dom** directly. For any object o, an *enumeration* of o is a string over Λ that describes o when interpreted in the obvious manner. For example,

- The only enumeration of the object $[9, 7]$ is the string `[9,7]`.
- The three strings `{1,2,3}`, `{3,2,1}`, and `{2,1,2,1,3,3}` all enumerate the same object $\{1, 2, 3\}$.

When we apply *enc* to an enumeration of an object o, we obtain what we call an *encoding* of o. For example, we can take again the first example above, and assume *enc* is the standard binary representation of natural numbers. Then the string `[1001,111]` encodes the object $[9, 7]$.

We now agree that a query $q : \tau_{\text{in}} \to \tau_{\text{out}}$ is *computable under enc* if there exists a Turing machine M that, whenever started on an input that is an encoding of some object o of type τ_{in}, will eventually halt and produce an encoding of $q(o)$ as output. In this case we also say that *M computes q under enc*.

The nice thing about generic queries is that the choice of encoding does not matter:

Proposition 1. *Let q be a generic query and let enc : $\mathbf{dom} \to \{0,1\}^*$ be a bijection. Let M be a Turing machine. If M computes q under enc, then M also computes q under any other bijection enc$'$: $\mathbf{dom} \to \{0,1\}^*$.*

In line with the above proposition, Hull and Su [19] proposed the notion of a *domain Turing machine*, which can work directly over the alphabet Λ. Thereto, the machine is equipped with a register that can hold an arbitrary atom. The machine can copy the atom from the current tape cell into the register, and conversely can copy the atom from the register into the current tape cell. Furthermore, the machine can test for equality between the atom in the register and the atom in the current tape call. Since domain Turing machines can directly take enumerations of objects as inputs, and can produce such enumerations as outputs, we no longer need encodings. Now Hull and Su showed that if a domain Turing machine computes a query q, then q must be generic, and conversely, every computable generic query can be computed by some domain Turing machine.

For fully generic queries, we can strengthen the Hull-Su result as follows. A *domain-oblivious* Turing machine is a restricted domain Turing machine that lacks the facility to test for equality in the sense described in the previous paragraph.

Theorem 1. *If query q is computed by a domain-oblivious Turing machine, then q is fully generic. Conversely, every fully generic computable query can be computed by some domain-oblivious Turing machine.*

The above result provides some insight into the notion of a fully generic query. It confirms the intuition that to process a fully generic query, we never need to inspect atoms in detail. We merely copy them or omit them altogether. Good examples are the relational algebra operations union, projection, and cartesian product. Furthermore, a fully generic query should not be sensitive to duplicates in the input, as these are allowed in enumerations as defined above.

Example 1 (Duplicates) . To illustrate the sensitivity to duplicates, consider the query $q_1 : \{\{d\}\} \to \{\{d\}\}$ defined as follows. Let D be an input object, $D = \{s_1, \dots, s_n\}$, where all the sets s_i are distinct. Then $q_1(D)$ consists of all the sets that can be written as $\{o_1, \dots, o_n\}$ such that $o_i \in s_i$ for $i = 1, \dots, n$. Note that the o_is picked need not be all distinct; for example, assuming $D = \{\{1,2\},\{1,3\}\}$, we can pick $o_1 = 1 \in \{1,2\}$ and $o_2 = 1 \in \{1,3\}$. Thus, the set $\{1,1\} = \{1\}$ belongs to $q_1(\{\{1,2\},\{1,3\}\})$.

This query is not fully generic, intuitively, because the given prescription for computing q_1 assumes that all the s_i are distinct. For example, suppose $D = \{\{1,2\}\}$. Then $q_1(D) = \{\{1\},\{2\}\}$. However, if we had presented D as an input with duplicates, say $\{\{1,2\},\{1,2\}\}$, the above prescription could generate $\{1,2\}$ as a possible element of the result, which is wrong.

We can formalize the above observation as follows. Let $D' = \{\{1,2\},\{3,4\}\}$ and take some $f : \mathbf{dom} \to \mathbf{dom}$ such that $f(1) = 1$, $f(2) = 2$, $f(3) = 1$, and $f(4) = 2$. Since $\{1,4\} \in q_1(D')$, we have $f(\{1,4\}) = \{1,2\} \in f(q_1(D'))$. However, $f(D') = \{\{1,2\}\}$ and $\{1,2\} \notin q_1(f(D'))$. Hence, q_1 is not fully generic.

Theorem 1 begs the following question, which is embarrassingly open:

Question 1. Do there exist fully generic queries that are not computable?

The conjecture is that the answer is negative. We even dare to conjecture that every fully generic query is computable in time linear in the output size. (Here, to get a useful notion of linear time, we would need to move from a Turing machine model to a RAM model of computation.) This conjecture is in line with the intuition that the processing is done in a manner that is oblivious to the actual identities of the atoms. Thus, all the processing time can be devoted to producing the output.

Boolean Queries and Canonical Forms. Restricted to Boolean queries, the above question has quite readily a negatie answer. A Boolean query has just a yes/no answer and can be modeled as a query $q : \tau \to \{[\,]\}$. Indeed, there are only two objects of type $\{[\,]\}$: the empty set and the singleton set containing the empty tuple. The empty set can be taken to represent 'no' and the other set 'yes'.

To see that any fully generic $q : \tau \to \{[\,]\}$ is computable, let D be an object of type τ and let $\mathbf{1}$ be the mapping that maps every atom to 1. We have $q(\mathbf{1}(D)) = \mathbf{1}(q(D)) = q(D)$. Consequently, we fully know the behavior of q once we know the behavior of q on objects in which only the atom 1 occurs. Such objects are called *canonical forms* [24]. For any given type τ, there are only finitely many canonical forms of type τ. Thus, every fully generic Boolean query can be summarized in a finite table and hence is always computable.

As an example, consider the type $\{\{d\}\}$. There are only four canonical forms: $\emptyset$; $\{\emptyset\}$; $\{\{1\}\}$; and $\{\emptyset,\{1\}\}$. For example, the canonical form of the object $\{\{1,2\},\{2,3,4\},\emptyset\}$ is $\{\emptyset,\{1\}\}$; the canonical form of $\{\{1,2\},\{2,3,4\},\{5\}\}$ is $\{\{1\}\}$. As a consequence, there are exactly $2^4 = 16$ fully generic Boolean queries with input type $\{\{d\}\}$.

We can characterize when two objects have the same canonical form in terms of a pre-order $A \leq B$ on objects of the same type. We define $A \leq B$ to hold when there exists a function $f : \mathbf{dom} \to \mathbf{dom}$ such that $A = f(B)$. We can now show the following.

Proposition 2. *Let A and B be objects of the same type. Then $\mathbf{1}(A) = \mathbf{1}(B)$ if and only if A and B have a common upper bound w.r.t. $\leq$, i.e., if there exists an object C such that $A \leq C$ and $B \leq C$.*

The 'if' implication is immediate; if $A = f(C)$ then $\mathbf{1}(A) = \mathbf{1}(f(C)) = \mathbf{1}(C)$, and similarly for B. The 'only if' implication is less trivial.[2]

[2] A comparable result was shown by Ta-Shma [9, Claim 3.4], [24, Proposition 4.2.4].

The Definability Question. Another computability question concerns definability by a fully generic query. This is the following problem:

Problem: Fully generic definability
Input: Two objects A and B
Decide: Does there exist a fully generic query q such that $q(A) = B$?

Question 2. Is the fully generic definability problem decidable?

In contrast, the corresponding classically generic definability problem is well understood. For simplicity, assume B is of some set type. Then A and B qualify if and only if B has only atoms from A, and every automorphism of A is also an automorphism of B.[3] In that case, the generic query mapping A to B can even taken to be expressible in first-order logic [6,8,23].

4 Query Language

More concrete insight in the fully generic queries can be gained by studying the language $\mathcal{L}$ already mentioned in the Introduction [9]. This language is an algebra of queries, similar to the powerset algebra of Abiteboul and Beeri [1], presented in monad style [13]. The algebra is built up from the following list of primitive queries, for all types τ, σ, τ_1, …, τ_k:

- The identity query $id : \tau \to \tau : o \mapsto o$.
- The unit query $[\,] : \tau \to [\,] : o \mapsto [\,]$ which always outputs the empty tuple on every input.
- For each $i \in \{1, \dots, k\}$, the projection $\pi_i : [\tau_1, \dots, \tau_k] \to \tau_i : [o_1, \dots, o_k] \mapsto o_i$.
- The empty-set query $\emptyset : \tau \to \{\sigma\} : o \mapsto \emptyset$, which always outputs the empty set of type σ.
- The singleton query $\{\cdot\} : \tau \to \{\tau\} : o \mapsto \{o\}$.
- The flatten query $\bigcup : \{\{\tau\}\} \to \{\tau\} : o \mapsto \bigcup o$.
- The union query $\cup : [\{\tau\}, \{\tau\}] \to \{\tau\} : [o_1, o_2] \mapsto o_1 \cup o_2$.
- The cartesian product $\times : [\{\tau\}, \{\sigma\}] \to \{[\tau, \sigma]\} : [o_1, o_2] \mapsto o_1 \times o_2$.
- The emptiness test

$$\mathit{ifempty} : [\{\sigma\}, \tau, \tau] \to \tau : [s, o_1, o_2] \mapsto \begin{cases} o_1 & \text{if } s \text{ is empty} \\ o_2 & \text{if } s \text{ is not empty.} \end{cases}$$

Moreover, we close the algebra under composition, tuple construction, and *map*, as follows:

- If $q_1 : \tau_1 \to \tau_2$ and $q_2 : \tau_2 \to \tau_3$ belong to $\mathcal{L}$, then so does the composition $q_2 \circ q_1 : \tau_1 \to \tau_3$.

[3] Here, an automorphism of A is a permutation f of the atoms occurring in A such that $f(A) = A$.

– If $q : \tau \to \sigma$ belongs to $\mathcal{L}$, then so does

$$map(q) : \{\tau\} \to \{\sigma\} : o \mapsto \{q(o') \mid o' \in o\}.$$

– If, for $i = 1, \dots, k$, we have $q_i : \sigma \to \tau_i$ in $\mathcal{L}$, then also $[q_1, \dots, q_k] : \sigma \to [\tau_1, \dots, \tau_k] : o \mapsto [q_1(o), \dots, q_k(o)]$ belongs to $\mathcal{L}$.

So far, we have done nothing else than described the standard nested relational algebra [13], without equality test. However, the definition of the language $\mathcal{L}$ must be completed by adding to the above list of primitive queries, for every type τ, the query *one-each* : $\{\{\tau\}\} \to \{\{\tau\}\}$ defined by

$$\{s_1, \dots, s_n\} \mapsto \{s'_1 \cup \dots \cup s'_n \mid \emptyset \neq s'_i \subseteq s_i \text{ for } i = 1, \dots, n\}.$$

The query *one-each* is quite intriguing and, compared to the powerset algebra mentioned above, the only novel aspect of the language $\mathcal{L}$. Actually, a more correct name would be *at-least-one-each*, but we stick to the original name. The query *one-each* is certainly primitive in $\mathcal{L}$, as it is the only query from $\mathcal{L}$ that can produce exponential-sized outputs. Indeed, when applied to just a singleton $\{s\}$, the output is already the powerset of s, except for the empty set. Conversely, however, it is conjectured that *one-each* can not be replaced by the powerset query, but a proof is lacking so far:

Question 3. Let $\mathcal{L}^{pow}$ denote the variant of $\mathcal{L}$ where we replace *one-each* by the powerset query *pow* : $\{\tau\} \to \{\{\tau\}\} : s \mapsto 2^s$. Does *one-each* belong to $\mathcal{L}^{pow}$?

Of course, the million-dollar question is the following:

Question 4. Does $\mathcal{L}$ contain all the fully generic queries?

An affirmative answer to this question would immediately yield a negative answer to Question 1 on the existence of noncomputable fully generic queries.

The only investigation on Question 4 so far was done in the PhD thesis by Paula Ta-Shma [24]. Her main result is that $\mathcal{L}$ does contain all the fully generic queries of signature $\{\{d\}\} \to \{\{d\}\}$. While her arguments are rich in valuable ideas, the arguments are also very intricate, and at some point no longer fully rigorous. We have failed to verify the arguments in detail; nevertheless, the thesis is a must-read for anyone interested in solving the above open question.

In our attempt to find more transparent arguments, we could prove the following result. For any natural number k, define the *k-powerset* of a set s as the set of all subsets of s of cardinality at most k.

Theorem 2. *The only fully generic queries of signature* $\{d\} \to \{\{d\}\}$ *are:*

– *the powerset query;*
– *for any* k, *the* k*-powerset query;*
– *for any of the above queries* q, *also the queries* $q^{(0)} : s \mapsto q(s) - \{\emptyset\}$; $q^{(1)} : s \mapsto q(s) \cup \{s\}$; *and* $q^{(2)} : s \mapsto (q(s) - \{\emptyset\}) \cup \{s\}$.

Note that all queries mentioned in the above theorem belong to $\mathcal{L}$, so this answers Question 4 for the special case of the signature $\{d\} \rightarrow \{\{d\}\}$.

As already mentioned, the language $\mathcal{L}$ without *one-each* is the standard nested relational algebra without equality test. Equivalence of nested relational algebra expressions is well known to be undecidable. When emptiness test is removed, and equality test is restricted to atoms, equivalence becomes decidable [14]. However, the equivalence problem when keeping emptiness test, but removing equality test altogether, seems to have escaped attention so far. We have the following interesting questions:

Question 5. Is the equivalence problem for expressions in $\mathcal{L}$ without *one-each* decidable? What about $\mathcal{L}$ including *one-each*? And what about $\mathcal{L}^{pow}$?

5 Technical Observations

A possible approach to comprehending the fully generic queries better is to investigate how large inputs we must consider to show the difference between two fully generic queries. For example, we cannot see the difference between the 5-powerset and the 6-powerset by considering only input sets with at most five elements. We can also investigate how many different atoms must come into play.

Consider, for example, the behavior of *one-each* : $\{\{d\}\} \rightarrow \{\{d\}\}$ on inputs in which at most two distinct atoms can occur. We can show:

Proposition 3. *Let $q' : \{\{d\}\} \rightarrow \{\{d\}\}$ be fully generic, such that $q'(D) =$ one-each(D) for every D in which at most two distinct atoms appear. Then q' equals one-each.*

We summarize the above result by saying that *one-each* is *2-determined*. In general, we say that a query q is *k-determined* if no other fully generic query agrees with q on all inputs involving at most k distinct atoms. If q is k-determined for some k, we also say that q is *finitely determined*.

For example, the k-powerset query of signature $\{d\} \rightarrow \{\{d\}\}$ is $(k+1)$-determined. In contrast, the powerset query is not finitely determined, because for any k, it agrees up to k atoms with the k-powerset query.

A special class of queries, also considered by Ta-Shma, are the *flat-output* queries, defined as those of signature of the form $\tau \rightarrow \{[d, \ldots, d]\}$ (the output is a flat relation). We can show:

Theorem 3. *Let q be a fully generic flat-output query and let k be the width of its output tuple type. Then q is $(k+1)$-determined.*

As a corollary, we obtain a more transparent proof of the result by Ta-Shma to the effect that every fully generic flat-output query belongs to $\mathcal{L}$. We only sketch the argument in this conference paper. Fix some ℓ atoms and, for a signature $\tau_{\text{in}} \rightarrow \tau_{\text{out}}$, define $[\![\tau_{\text{in}}]\!]^{(\ell)}$ and $[\![\tau_{\text{out}}]\!]^{(\ell)}$ as the (finite) subsets of $[\![\tau_{\text{in}}]\!]$ and $[\![\tau_{\text{out}}]\!]$ consisting of the objects involving only the ℓ given atoms. There are only finitely many fully generic functions $q : [\![\tau_{\text{in}}]\!]^{(\ell)} \to [\![\tau_{\text{out}}]\!]^{(\ell)}$, and these can be represented in $\mathcal{L}$.

In general, the usefulness of finite determinacy remains unclear, as it is not preserved by composition. For example, both $\{\cdot\}$ and *one-each* are finitely determined, but their composition, the powerset query, is not.

6 Conclusion

This paper is an invitation, a "call to arms", for a renewed investigation of the forgotten, but very natural and fascinating class of fully generic queries. Indeed, we have characterized these queries as those that can be processed in a domain-oblivious manner, and are insensitive to duplicates in the input. Various data transformations or restructurings have this property. We have posed the main open questions and have proposed some possible approaches. We are looking forward to answers appearing in the near future.

It should also be investigated how full genericity behaves in a setting where collections are bags instead of sets. For example, the bag version of the query q_1 from Example 1, where we do not assume that all the s_i are distinct, is fully generic in the bag setting. Some of the questions we have posed may become easier in the bag setting, but others (e.g., Question 5) may become more difficult.

References

1. Abiteboul, S., Beeri, C.: On the power of languages for the manipulation of complex objects. VLDB J. **4**(4), 727–794 (1995)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995)
3. Abiteboul, S., Vianu, V.: Computing with first-order logic. J. Comput. Syst. Sci. **50**(2), 309–335 (1995)
4. Afrati, F., Ullman, J.: Optimizing multiway joins in a map-reduce environment. IEEE Trans. Knowl. Data Eng. **23**(9), 1282–1298 (2011)
5. Aho, A., Ullman, J.: Universality of data retrieval languages. In: Conference Record, 6th ACM Symposium on Principles of Programming Languages, pp. 110–120 (1979)
6. Arenas, M., Diaz, G.: The exact complexity of the first-order logic definability problem. ACM Trans. Database Syst. **41**(2), 13:1–13:14 (2016)
7. Armbrust, M., Xin, R., et al.: Spark SQL: relational data processing in Spark. In: Proceedings 2015 International Conference on Management of Data, pp. 1383–1394. ACM (2015)
8. Bancilhon, F.: On the completeness of query languages for relational data bases. In: Winkowski, J. (ed.) MFCS 1978. LNCS, vol. 64, pp. 112–123. Springer, Heidelberg (1978). https://doi.org/10.1007/3-540-08921-7_60
9. Beeri, C., Milo, T., Ta-Shma, P.: Towards a language for the fully generic queries. In: Cluet, S., Hull, R. (eds.) DBPL 1997. LNCS, vol. 1369, pp. 239–259. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-64823-2_14
10. Blass, A., Gurevich, Y., Shelah, S.: Choiceless polynomial time. Ann. Pure Appl. Logic **100**, 141–187 (1999)

11. Botoeva, E., Calvanese, D., Cogres, B., Xiao, G.: Expressivity and complexity of MongoDB queries. In: Kimelfeld, B., Amsterdamer, Y. (eds.) Proceedings 21st International Conference on Database Theory. LIPIcs, vol. 98, pp. 9:1–9:23. Schloss Dagstuhl-Leibniz Center for Informatics (2018)
12. Buneman, P., Cheney, J., Vansummeren, S.: On the expressiveness of implicit provenance in query and update languages. ACM Trans. Database Syst. **33**(4), 28:1–28:47 (2008)
13. Buneman, P., Naqvi, S., Tannen, V., Wong, L.: Principles of programming with complex objects and collection types. Theor. Comput. Sci. **149**(1), 3–48 (1995)
14. Van den Bussche, J., Van Gucht, D., Vansummeren, S.: Well-definedness and semantic type checking for the nested relational calculus. Theor. Comput. Sci. **371**(3), 183–199 (2007)
15. Chandra, A.: Programming primitives for database languages. In: Conference Record, 8th ACM Symposium on Principles of Programming Languages, pp. 50–62 (1981)
16. Chandra, A., Harel, D.: Computable queries for relational data bases. J. Comput. Syst. Sci. **21**(2), 156–178 (1980)
17. Garcia-Molina, H., Ullman, J., Widom, J.: Database Systems: The Complete Book. Prentice Hall, Upper Saddle River (2009)
18. Grädel, E., Grohe, M.: Is polynomial time choiceless? In: Beklemishev, L.D., Blass, A., Dershowitz, N., Finkbeiner, B., Schulte, W. (eds.) Fields of Logic and Computation II. LNCS, vol. 9300, pp. 193–209. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23534-9_11
19. Hull, R., Su, J.: Algebraic and calculus query languages for recursively typed complex objects. J. Comput. Syst. Sci. **47**(1), 121–156 (1993)
20. Hull, R., Yap, C.: The format model, a theory of database organization. J. ACM **31**(3), 518–537 (1984)
21. Kuper, G., Vardi, M.: The logical data model. ACM Trans. Database Syst. **18**(3), 379–413 (1993)
22. Madelaine, F.: Mémoire d'habilitation à diriger des recherches, Université Blaise Pascal, Clermond-Ferrand (2012). https://tel.archives-ouvertes.fr/tel-01096078
23. Paredaens, J.: On the expressive power of the relational algebra. Inf. Process. Lett. **7**(2), 107–111 (1978)
24. Ta-Shma, P.: Genericity in Database Query Languages. Ph.D. thesis, Hebrew University (1997)
25. Tarski, A.: What are logical notions? History and philosophy of logic **7**, 143–154 (1986). Edited by J. Corcoran
26. Zaharia, M., et al.: Spark: cluster computing with working sets. In: Proceedings 2nd USENIX Workshop on Hot Topics in Cloud Computing (2010)

Data Analysis

Keeping the Data Lake in Form: DS-kNN Datasets Categorization Using Proximity Mining

Ayman Alserafi[1,2](✉), Alberto Abelló[1], Oscar Romero[1], and Toon Calders[3]

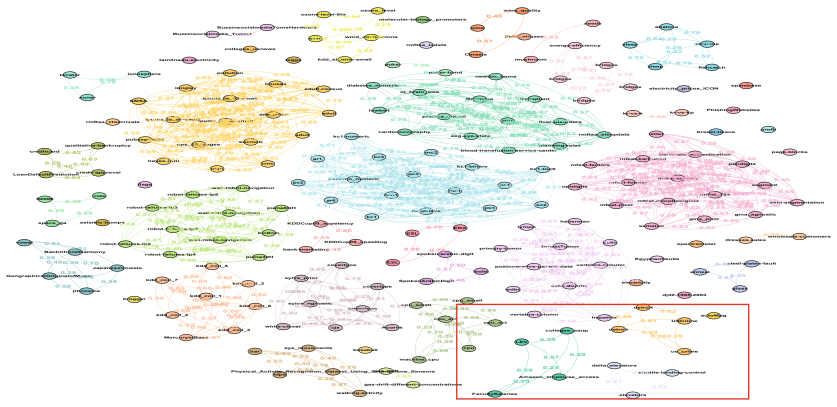
[1] Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Catalunya, Spain
{alserafi,aabello,oromero}@essi.upc.edu
[2] Université Libre de Bruxelles (ULB), Brussels, Belgium
[3] Universiteit Antwerpen (UAntwerp), Antwerp, Belgium
toon.calders@uantwerp.be

Abstract. With the growth of the number of datasets stored in data repositories, there has been a trend of using Data Lakes (DLs) to store such data. DLs store datasets in their raw formats without any transformations or preprocessing, with accessibility available using schema-on-read. This makes it difficult for analysts to find datasets that can be crossed and that belong to the same topic. To support them in this DL governance challenge, we propose in this paper an algorithm for categorizing datasets in the DL into pre-defined topic-wise categories of interest. We utilise a k-NN approach for this task which uses a proximity score for computing similarities of datasets based on metadata. We test our algorithm on a real-life DL with a known ground-truth categorization. Our approach is successful in detecting the correct categories for datasets and outliers with a precision of more than 90% and recall rates exceeding 75% in specific settings.

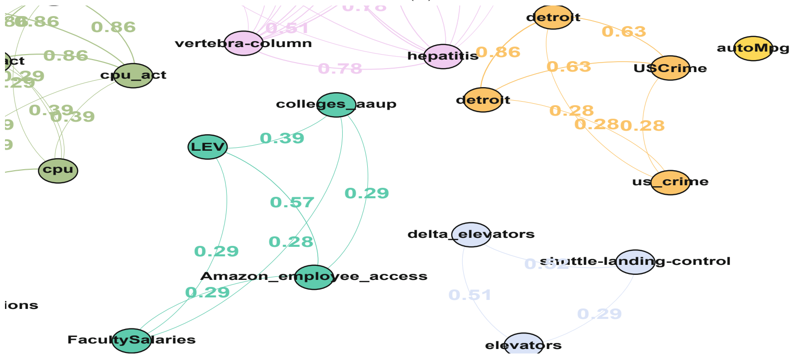
Keywords: Data lake categorization · k-Nearest-Neighbour · Metadata management · Proximity mining

1 Introduction

Today, a lot of data is generated covering different heterogeneous topics and domains. Those data are frequently stor ed as tabular datasets which describe different entities (in the rows) with information about them stored as attributes (in the columns). A collection of such raw datasets which are stored in their original schema without preprocessing or transformations is called a Data Lake (DL) [3,16]. Over its lifetime, a DL becomes very diverse and can cover different topics, making it difficult to find and retrieve relevant datasets for analysis. Therefore, it is a challenge for the users to govern the DL by detecting the groupings and underlying structures of similar datasets covering relevant topics for analytics [3,4,11]. To tackle this challenge, we propose an automated approach called DS-kNN to detect such groupings using k-nearest-neighbour (k-NN). The approach relies on collecting relevant metadata about the datasets when they are ingested, then we compute proximity models of dataset similarities based on supervised machine learning, and apply those models on new datasets to compute their similarity scores with datasets stored in the DL. Once we computed the similarities, we apply a k-NN algorithm to categorize the ingested datasets into the groupings already present in the DL or to classify them as outliers. An example of the expected results can be seen in Fig. 1. Here, we visualise the DL as a *proximity graph* having datasets as nodes and edges connecting the nodes showing the similarity scores ($\mathbb{R} \in [0, 1]$) computed using the proximity model. Datasets in the same category (cluster) are shown in the same colour. We only show edges between datasets in the same category. In Fig. 1(a) we show an example of a complete DL proximity graph and in (b) we zoom-in on the specific part highlighted with a box for showing more details.



(a)



(b)

Fig. 1. A visualisation of the output from DS-kNN data lake (DL) categorization. A proximity graph shows the datasets as nodes and the proximity scores as edges between nodes. (a) complete DL and (b) a zoomed-in view highlighted by the red box in (a) (Color figure online)

The main contributions of this paper are: (1) We propose a kNN-based proximity mining algorithm for finding the correct categories for datasets based on existing categories in the DL, (2) we evaluate the algorithm in a real-world setting to prove its effectiveness in assigning correct categories to new datasets ingested in the DL, (3) we experimentally test the effect of different DL settings on the performance of our approach.

In the rest of this paper, we define the DL and the scenario we consider in Sect. 2, we present the DS-kNN algorithm in Sect. 3, then we test the algorithm on a real-life DL and we experiment with our algorithm in Sect. 4, we present related work in Sect. 5, and we conclude in Sect. 6.

2 Preliminaries

We consider a DL consisting of tabular datasets. Those are large heterogeneous repositories of *flat structured data* (i.e., CSV, web tables, spreadsheets, etc.). Such datasets are structured as groups of *instances* describing real-world entities, where each instance is expressed as a set of *attributes* describing the properties of the entity. We formally define a dataset D as a set of instances $D = \{I_1, I_2, ...I_n\}$. The dataset has a set of attributes $S = \{A_1, A_2, ...A_m\}$, where each attribute A_i has a fixed type, and every instance has a value of the right type for each attribute. We focus on two types of attributes: continuous ***numeric attributes*** with real numbers and categorical ***nominal attributes*** with discrete values.

For each dataset, we collect different statistics about their content which we call *content meta-features*:

- **Nominal attributes:** their data profile mainly involves frequency distributions of their distinct values.
- **Numeric attributes:** their data profile mainly involves aggregated statistics like mean, min, max, and standard deviations.

We compute similarity scores between pairs of datasets $[D_a, D_b]$, as follows:

- $Sim(D_a, D_b)$: an estimation ($\mathbb{R} \in [0, 1]$) of the similarity based on the comparison of the *content meta-features* we collect about the datasets and their attributes. Typically, the information contained in highly similar datasets would overlap. An example would be a pair of datasets having similar numeric values and distribution of values, or nominal attributes having the same number of values. Alternatively, it could be based on *name string-similarity* between datasets and their attributes.

Scenario. We aim at governing the DL by incrementally maintaining the clusters of datasets defined for them. We consider the scenario where we initially have an existing DL for which we know all clusters of datasets based on their categories. However, given the dynamic nature of DLs, new datasets are frequently ingested. Thus, we need to compare these new datasets against the datasets already in the DL to find their similarity with them, and then to find their most appropriate

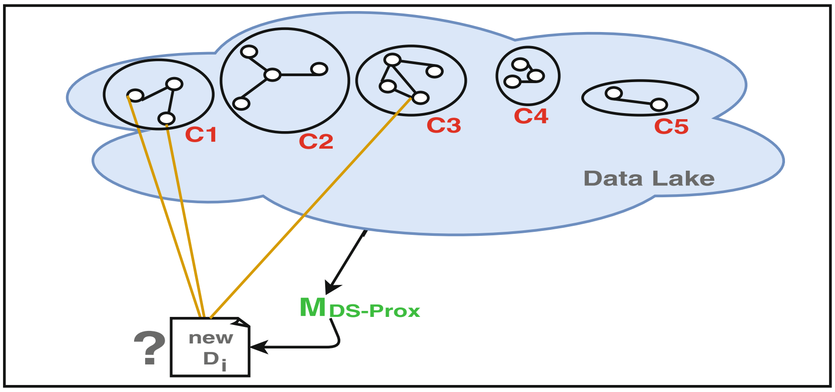


Fig. 2. The data lake categorization scenario using k-NN proximity mining

category based on the similar datasets found in the DL, or to assign them to a separate category as an outlier.

This shapes the main problem for this research paper: given a collection of datasets in a DL and a newly ingested dataset, find all pairs of highly similar datasets, and based on their categories, assign a new category for the new dataset, or if no highly similar datasets are found then indicate that the dataset is an outlier. To compute the similarity between the datasets, we use a proximity model, which we call $M_{DS-Prox}$. We discuss how we create this model in Subsect. 2.1.

The scenario discussed is visualized in Fig. 2. Consider that there is a DL having a group of datasets (white circles) which have annotations of all their $Sim(D_a, D_b)$ relationships between pairs (as seen by the lines linking the datasets). Groups of datasets with linkages are segmented into categories (seen by the encompassing black circles). Those categories are the groupings of the subject-areas or domains-of-knowledge we have in the DL. We need to automatically use this DL and its known annotations to create a model $M_{DS-Prox}$ which can automatically annotate relationships of a new dataset D_i with the $Sim(D_i, D_b)$ similarity scores. Therefore, we need to learn a model from the DL and apply it to estimate the similarity between a new dataset and all other datasets already in the DL, in order to find the top-k neighbours.

Based on the similarity scores we assign a category to D_i. The highlighted edges between the new dataset D_i and some nodes in the DL are those having the highest similarity scores computed by the model (in this case, we give an arbitrary example where we use top-3 nearest-neighbours). In our proposed approach, each of those top neighbours proposes its category as the correct one for D_i, and the most proposed category should be assigned, or if no such similar datasets are found then D_i is marked as an outlier without any relevant category found. In the case of tied categories among the proposed ones from the top-k similar datasets, then all of them are assigned to D_i. In the example in Fig. 2, category 'C1' would be assigned as the final category as it has 2 *votes*, compared to only 1 *vote* by category 'C3'.

To learn the $M_{DS-Prox}$ model we use supervised machine learning as described in Subsect. 2.1.

2.1 Proximity Mining: Meta-features Metrics and Models

For all the datasets in the DL, we collect two metadata types: **A. Content-based** and **B. Name-based** meta-features. The name-based techniques are the most commonly used metadata in previous research [7,11,13,14]. In our DS-kNN approach, we propose content-based meta-features as an alternative to name-based metadata when computing similarity scores. Such content meta-features include data profiling statistics about the content of the datasets. Thus, we use two types of metadata for similarity computations:

- *Name-based metadata:* the naming of datasets and their attributes.
- *Content-based metadata:* profiling statistics about the data stored in the datasets. The collected meta-features (described in Table 1) include statistics concerning all attributes collectively, the attribute types found and the overall number of instances. Those form a concise list of meta-features that have been proved in our previous work [4] to be effective in predicting related datasets with similar schemata and stored information. Our purpose for those meta-features is to describe the general structure and content of the datasets for an approximate comparison using our proximity mining classification models.

To compute similarity scores $Sim(D_a, D_b)$ from name-based metadata, we use the Levenshtein distance as a standard string comparison metric [12]. The output from this comparison is considered as the similarity score from $M_{DS-Prox}$. For content meta-features, we construct the $M_{DS-Prox}$ model using the proximity mining approach from our previous work [4]. First, we compute distances for each meta-feature m_i from Table 1 between each pair of datasets $[D_a, D_b]$ using Eq. 1 which gives the relative difference as a number between 0 and 1. We compute this for all dataset pairs $[D_a, D_b]$ in the training sample.

$$dist_{m_i}(D_a, D_b) = \frac{\max\{m_i(D_a), m_i(D_b)\} - \min\{m_i(D_a), m_i(D_b)\}}{\max\{m_i(D_a), m_i(D_b)\}} \tag{1}$$

Once we have the metadata collected and their distances computed, we feed them to a supervised machine learning algorithm to produce a classification model which identifies those dataset pairs in the same assigned category. This creates the proximity mining model to compute similarity scores. For this initial training sample of datasets we have in the DL, a data analyst should have incrementally assigned a category cluster to each dataset based on their topics. The target variable for those classifiers is a binary value whether the datasets in the pair belong to the same category or not.

We use the two top performing ensemble learning algorithms from [4] to learn the model, which are the boosting machine learning algorithms AdaBoost [15]

Table 1. DS-Prox meta-features

Type	Meta-feature	Description
General	Number of instances	The number of instances in the dataset
	Number of attributes	The number of attributes in the dataset
	Dimensionality	The ratio of number of attributes to number of instances
Attributes by type	Number per type	The number of attributes per type (nominal or numerical)
	Percentage per type	The percentage of attributes per type (Nominal or Numerical)
Nominal attributes	Average number of values	The average number of distinct values per nominal attribute
	Standard deviation of number of values	The standard deviation in the number of distinct values per nominal attribute
	Minimum/Maximum number of values	The minimum and maximum number of distinct values per nominal attribute
Numeric attributes	Average Numeric Mean	The average of the means of all numeric attributes
	Standard Deviation of the Numeric Mean	The standard deviation of the means of the numeric attributes
	Minimum/Maximum numeric mean	The minimum and maximum mean of numeric attributes
Missing values	Missing attribute count	The number of attributes with missing values
	Missing attribute percentage	The percentage of attributes with missing values
	Minimum/Maximum number of missing values	The minimum and maximum number of instances with missing values per attribute
	Minimum/Maximum missing values percentage	The minimum and maximum percentage of instances with missing values per attribute
	Mean number of missing values	The mean number of missing values from each attribute
	Mean Percentage of Missing Values	The mean percentage of missing values from each attribute

and LogitBoost [6]. Those algorithms were compared in our previous work to other algorithms and were found to be the best in finding related schemata. The positive-class distribution produced by the ensemble model is used as the similarity score $Sim(D_a, D_b)$ [15]. Finally, we apply the learnt $M_{DS-Prox}$ model on pairs of one new ingested dataset and each existing datasets in the DL to generate the similarity scores. We compare the score against a minimum threshold like in Eq. 2. Only pairs passing the threshold are considered as candidate top-k nearest neighbours to a dataset in our DS-kNN algorithm. We discuss this in detail in Sect. 3. Different similarity thresholds lead to a different performance of the algorithm, so we test multiple threshold values in our experiments to discover the best one to use.

$$Top(D_a, D_b) = \begin{cases} 1, & Sim(D_a, D_b) > c_{rel} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

3 DS-kNN: A Proximity Mining Based k-Nearest-Neighbour Algorithm for Categorizing Datasets

Algorithm 1: DS-kNN Categorization of a dataset ingested in a Data Lake

```
Input: A new ingested dataset $D_a$, each existing dataset $D_b$ in the data lake
       $DL$, Dataset-level meta-features distance metrics $MF$ for each pair of
       datasets $\{D_a, D_b\}$, the category $Cat_{D_b}$ for each existing dataset in the
       $DL$, the classification model $M_{ds-prox}$, algorithmic parameters: the
       number $k$ of nearest neighbours, and the similarity score threshold $c_{rel}$
Output: The set $SP$ of the ingested dataset and its similarity scores
        $Sim(D_a, D_b)$ and category $Cat_{D_b}$ for each pair $\{D_a, D_b\}$ passing $c_{rel}$,
        the set $SP\text{-}Top$ of top matching $k$ datasets and their categories, the
        assigned category for the new dataset $Cat_{D_a}$
$SP \leftarrow \emptyset$;
$SP\text{-}Top \leftarrow \emptyset$;
foreach $\{D_a, D_b\} \subset DL$ and $a \neq b$ do
    $[D_a, D_b, Sim(D_a, D_b)] = M_{ds-prox}(MF_{\{D_a,D_b\}})$;
    if $Sim(D_a, D_b) > c_{rel}$ then
        $SP \leftarrow SP \cup \{[D_a, D_b, Sim(D_a, D_b), Cat_{D_b}]\}$;
    end
end
$SP\text{-}Top$ = Top-k_Nearest_Neighbours($SP$, k); \\Retrieve the subset of the
highest ranking k-pairs by similarity score
$Cat_{D_a} = Top\text{-}category(SP\text{-}Top)$; \\Get category with majority vote from Top-k
if $(Cat_{D_a} = NULL)$ then
    $Cat_{D_a} =' Outlier'$;
end
```

We propose an algorithm for computing the categories of an ingested dataset as described in the scenario in Sect. 2. After learning the classification model $M_{DS-Prox}$, we apply the classifier to each new pair $[D_a, D_b]$ where D_a is any new ingested dataset and D_b is each of the existing datasets in the DL, in order to achieve the similarity score $Sim(D_a, D_b)$ with all datasets in the DL. Then, we apply k-NN in our proposed DS-kNN Algorithm 1 to compute the category for a new dataset. k-NN was also successful in similar categorization problems, like in free-text document categorization [8].

First, our algorithm applies the $M_{DS-Prox}$ model on all the dataset pairs for the new dataset D_a to compute their similarity scores, and those passing the minimum threshold are stored in the set SP. To improve efficiency, a heap data structure could be used to store the datasets with their similarity scores for quick search and retrieval of top-k nearest-neighbours. The next step involves finding those top-k nearest-neighbours which are existing datasets in the DL with the highest similarity scores to D_a. Finally, we assign the category with the most number of pairs in the top-k nearest neighbours as the assigned category based on simple majority voting by top-k nearest neighbours. If no top-k nearest neighbours are found then the dataset is marked as an 'outlier' with no proposed category.

The algorithm has the following parameters as input:

- **The number of neighbours (k):** the top-k number of nearest neighbours which our algorithm uses to predict the new category for an ingested dataset.
- **The proximity model** ($M_{ds-prox}$): this is the proximity mining model created using our approach described in Sect. 2. We use different models depending on the metadata, i.e. content-based, dataset-name based or attribute-name based.
- **The similarity threshold** (c_{rel}): the minimum allowed similarity score to consider a dataset pair as candidate nearest neighbour.

4 Experimental Evaluation

We test our proposed categorization algorithm on a real-life DL. We describe the dataset used, the experimental setup and our results with a detailed discussion of the performance of DS-kNN.

4.1 Dataset: OpenML DL Ground-Truth

We created a ground-truth based on manual annotations of 203 datasets from a real-life DL called OpenML[1]. It consists of different datasets covering heterogeneous topics, each having a name and a description. The categories found in the ground-truth are visualised in Fig. 1.

The sample of datasets collected from OpenML is scraped to extract datasets having a description of more than 500 characters. The descriptions helped the manual annotators deciding on the assigned topic for each dataset. Out of the 514 datasets retrieved, we selected 203 with meaningful descriptions (i.e., excluding datasets whose descriptions do not allow to interpret its content and to assign a topic). A domain expert and one of the authors collaborated to manually label the datasets with their topic. The datasets were labelled by both their *broad* **subject** (e.g., '*social demographics*') and their more *specific* **entity** they describe (e.g., *'citizens census data'*). The interested reader can download the two annotated datasets from GitHub[2].

Table 2 shows the number of datasets per category assigned based on topic grouping type. We only show the top 10 categories found by size for each grouping. The total number of categories is also given and the number of categories bigger than a specific size (i.e., with at least this number of members), and the number of outliers (datasets with their own specific category without any other members). As can be seen in the table, the datasets in the DL we use in the experiments cover heterogeneous topics and different category sizes.

[1] http://www.openml.org.
[2] https://github.com/AymanUPC/ds-knn.

Table 2. A description of the 203 OpenML categorized datasets collected. Datasets are categorized by subject and by entity.

Category type	No. of categories	Categories by type	Categories by size	Outliers
Subject	53	Computer Software (17), Social Demographics (17), Image Recognition (16), Health (14), Robot (11), Disease (11), Natural Water (8), Ecology (8), Computer Hardware (6), Motion Sensing (5)	8+ members (8), 5+ members (14), 3+ members (25)	21
Entity	77	Computer Software defects (16), Citizens Census Data (12), Digit Handwriting Recognition (12), Diseases (11), Robot Motion Measurements (11), Health Measurements (10), Chemical Contamination (8), Plantation Measurements (8), CPU Performance Data (6), Animal Profile (5)	8+ members (8), 5+ members (12), 3+ members (21)	47

4.2 Experimental Setup

Our goal is to test the performance of the DS-kNN algorithm in correctly assigning the right category to datasets. We compare the performance of the DS-Prox content-based models when applied in DS-kNN against the baseline models of dataset-names and attribute-names, which are the commonly used metadata in previous work [7,11,13,14] (see Sect. 5). We implement DS-kNN based on those different models in Java using a Postgres SQL database as its backend for storing the metadata, and we feed it with the datasets from the OpenML DL. We initially tested the algorithm on a random sample of datasets using different values for $k \in \{3, 5, 7, 9, 11, 13\}$ and found the best performing value under the same settings was $k = 3$, so we conduct all our trials in the experiments using $k = 3$. To test the generalizability and adaptability of DS-kNN under different DL settings, we also conduct trials with the algorithm under the following different settings which affect the ground-truth used in the training and testing of $M_{ds-prox}$:

– **Different category sizes:** we test DS-kNN with all the datasets (including outliers where category size is just 1 dataset) and with categories that at least contain the following number of members (3,5,8). We test different sizes of categories to check if the algorithm is affected by category sizes.

- **Different ground-truth types:** We test the algorithm with the broad (1) subject-based categories and the more detailed (2) entity-based categories.

For each DL setting, we compare the performance of the DS-kNN algorithm using the following input parameters:

- **Different models** ($M_{ds-prox}$): we test the different models generated by different metadata, which are (1) Dataset Name, (2) Attribute Name and (3) DS-Prox Content.
- **Different similarity thresholds:** we use different thresholds for $Sim(D_a, D_b)$ including $(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)$

We test the different combinations of the above parameters and settings which resulted in the execution of a total of 240 independent trials. We utilise a leave-one-out experimental setup to test our categorization algorithm in each trial, as seen in Fig. 2, so for each experimental trial we train the model under the same settings and with the same parameters 203 times, where for each run we keep a single dataset out from the training of the model and treat it as the new test dataset. We train the proximity model with all the datasets in the DL except the test dataset, we apply the proximity model on the test dataset with all dataset pairs found in the DL, and we run our algorithm to compute its allocated category or to mark it as an outlier. We apply Algorithm 1 on the test dataset and we find the top categories it should be allocated to. The goal is to maximise the number of correctly assigned categories based on top-k nearest neighbours.

To evaluate the effectiveness, we consider our algorithm as an example of a multi-class classification problem. We evaluate whether each dataset gets assigned the correct category based on *top-k nearest neighbours*. We compute the number of correctly annotated categories and outliers by measuring recall, precision and F1-scores which are commonly used for evaluation in similar settings [1,2,11]. We compute the F1 score as the harmonic mean of the recall and the precision [12]. The evaluation metrics are described in Eqs. (3), (4) and (5) respectively. Here, TP means true-positives which are the datasets correctly classified to their category. FN are false negatives, and FP are false positives. We compute the evaluation metrics per category and average the final scores from all categories to achieve macro-averaging scores [12]. For example, consider we have in the ground-truth two categories $C1$ and $C2$ consisting of 10 datasets each. $C1$ had 9 TPs and 1 FP (i.e. a dataset from a different category incorrectly assigned to it by DS-kNN) while $C2$ had 8 TPs and 2 FPs, therefore they will have a precision of 0.9 and 0.8 respectively. Therefore, the macro-precision will be $\frac{0.8+0.9}{2} = 0.85$.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1{-}score = 2 \times \frac{(Recall \times Precision)}{Recall + Precision} \tag{5}$$

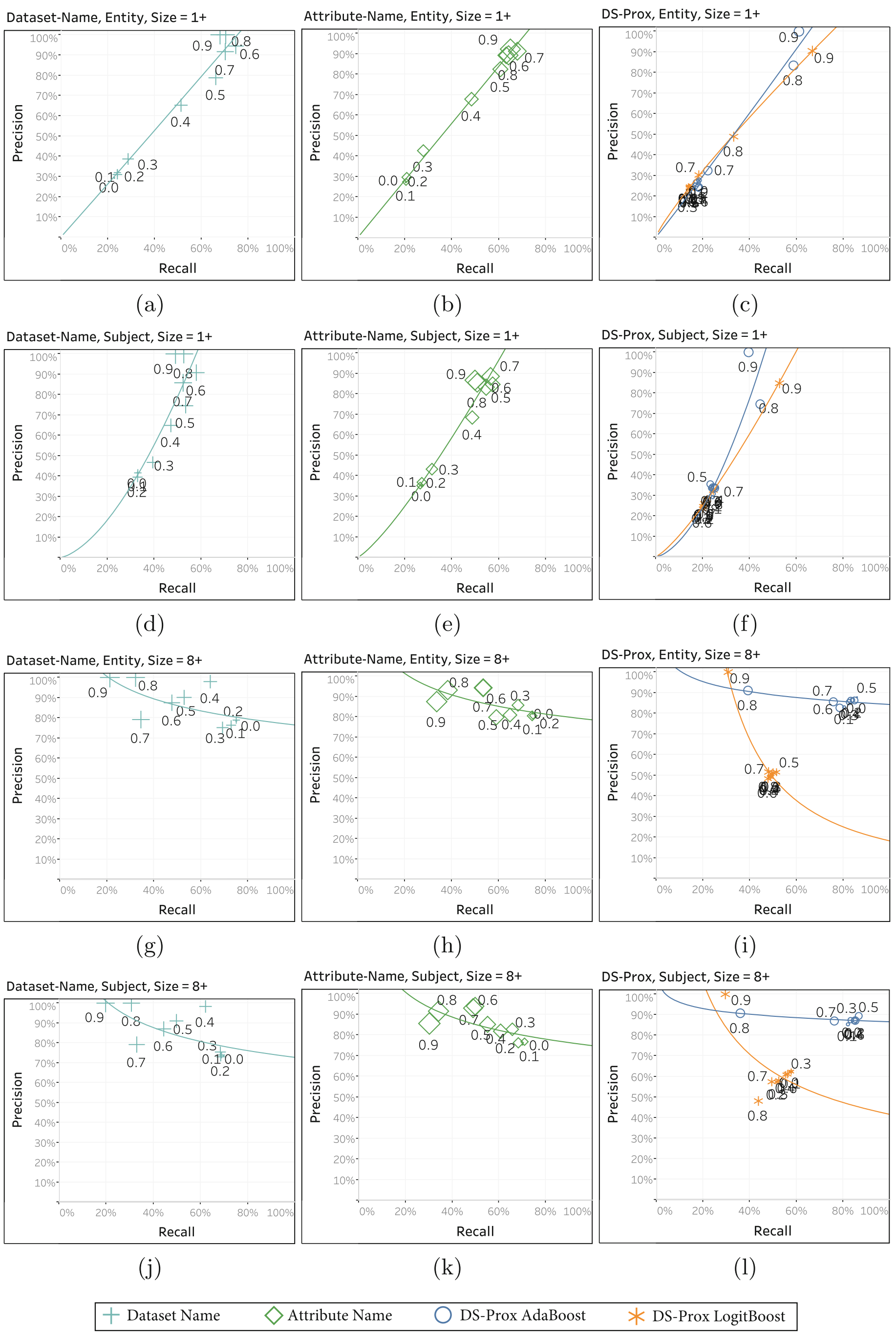


Fig. 3. Performance of DS-kNN using different models, different ground-truths, and different category sizes

4.3 Results

We present the precision-recall curves from our experiments in Fig. 3. Each graph plots the macro-averaging performance resulting from leave-one-out cross-validation of a specific model for a specific ground-truth type and category sizes (which are labelled above the chart). For all our results we use percentages for the performance metrics. Here, we plot recall against precision for each of the different model types used in DS-kNN and the different minimum category sizes we use in the experiment. The numbers annotated on the points indicate the similarity threshold (also indicated by the size of the points, where bigger size indicates a higher similarity threshold). We show the results for the non-restricted (category size = 1+) which includes outliers and the biggest category sizes (category size = 8+). Each model type has a different symbol and colour. For DS-Prox, circles indicate AdaBoost-based model and stars indicate LogitBoost-based model. We also present in Table 3 the evaluation metrics for the top performing parameters for DS-kNN (in terms of F1-scores) for each model type, category sizes, and ground-truth type.

As could be seen from the results, DS-kNN performs comparatively well with the attribute-name and the DS-Prox content-based models for category size 1+, but for larger category sizes the DS-Prox content models are better in assigning the correct categories. For example, DS-Prox content leads to a precision of about 90% and recall higher than 80% for category sizes of at least 8 members and the entity-based ground-truth, while attribute-name model can only achieve 82% precision and 75% recall. Dataset-name based model performs worse in terms of recall with 64% but much better precision with 98%. The results also indicate that the choice of the similarity threshold can affect the performance of DS-kNN.

In general, DS-kNN performs better with bigger category sizes than smaller category sizes as it becomes easier for the algorithm to find relevant top-k nearest neighbours. However, it is still good in detecting outliers and other categories as seen for the performance for *'min. category size'* = 1, for example a recall of 75% and precision of 95% for dataset-name based model. The dataset-name model performs better in detecting outliers as seen from this result. The DS-kNN algorithm performed equally good with both ground-truth types under the same settings and with the same parameters, yet slightly better with the more specific entity-based ground-truth with small category sizes and outliers. This indicates the adaptability of DS-kNN to different DL settings and properties.

5 Related Work

Categorization of datasets from heterogeneous domains is an emerging research topic, and relevant previous research include [11], where they utilise the attribute names to cluster the datasets into categories using a probabilistic model. Datasets are assigned to different categories using different probabilities. They tackle the multi-label classification of datasets and retrieval of datasets from relevant domains by querying systems. Our approach improves this approach by using a machine-learning based approximate proximity mining technique instead

Table 3. The evaluation of DS-kNN for $k = 3$ and different model types, ground-truth types and minimum category sizes. For each setting, we only show here the best performing similarity threshold based on F1-scores.

Model type	Ground truth type	Min. category size	Similarity threshold	Recall	Precision	F1-score
Attribute Name	Entity	1	0.7	67.9	91.3	77.9
Attribute Name	Entity	3	0.3	55.8	75.3	64.1
Attribute Name	Entity	5	0.3	57.5	78.3	66.3
Attribute Name	Entity	8	0	74.6	81.5	77.9
Attribute Name	Subject	1	0.7	56.5	88.6	69
Attribute Name	Subject	3	0.4	47.2	67	55.4
Attribute Name	Subject	5	0.3	55.1	71.4	62.3
Attribute Name	Subject	8	0.1	70.9	76.7	73.7
Dataset Name	Entity	1	0.6	74.7	94.4	83.4
Dataset Name	Entity	3	0.4	49.9	90.1	64.4
Dataset Name	Entity	5	0.4	59.6	98.5	74.3
Dataset Name	Entity	8	0.4	64	98	77.4
Dataset Name	Subject	1	0.6	58	90.8	70.7
Dataset Name	Subject	3	0.4	47.9	74.5	58.3
Dataset Name	Subject	5	0.4	55.7	98.9	71.3
Dataset Name	Subject	8	0.4	62.1	98.3	76.1
DS-Prox Content	Entity	1	0.9	66.8	90.5	76.8
DS-Prox Content	Entity	3	0.2	53.9	61.2	57.4
DS-Prox Content	Entity	5	0.3	68.4	81.9	74.6
DS-Prox Content	Entity	8	0	85.9	87.8	86.8
DS-Prox Content	Subject	1	0.9	52.8	84.8	65.1
DS-Prox Content	Subject	3	0.1	44.5	54.4	48.9
DS-Prox Content	Subject	5	0.2	58.7	70	63.9
DS-Prox Content	Subject	8	0.5	86.8	89.4	88.1

of the Jaccard similarity of exact values. We also use content-based metadata for categorizing and not only name-based metadata. This is important for DLs where datasets are not well maintained with meaningful attribute names.

Clustering could also be applied to other types of semi-structured datasets like ontologies [1] and XML documents [2,10], etc. In [1], they propose an algorithm to cluster instances from different ontologies based on their structural properties in the ontology graphs. Their goal is to facilitate ontology matching rather than domains discovery. Similarly, in [2,10] they cluster the semi-structured documents based on their structure similarity and linguistic matchers.

Clustering free-text without any structure is also possible. For example, [5] aims to cluster short text messages by computing TF-IDF word similarity between free-text documents. Similarly, [8] categorizes free-text documents using a k-NN based algorithm by first extracting TF-IDF weighted labels and feeding them to the algorithm. Another specific application would be clustering streaming data where a sliding window algorithm could be used [9], where they also use k-NN when finding relevant clusters for a given data instance ingested in a stream of data points.

6 Conclusion

We proposed DS-kNN, a categorization algorithm for classifying datasets into pre-defined topic-wise groups. Our algorithm can be applied in a DL environment to support users in finding relevant datasets for analysis. Our algorithm uses extracted metadata from datasets to compute their similarities to other datasets in the DL using a proximity mining model and name strings comparisons. Those similarity scores are fed to DS-kNN to decide on the most relevant category for a dataset based on its top-k nearest neighbours. Our algorithm was effective in categorizing the datasets in a real-world DL and detecting outliers, yet our results can be improved to achieve better performance. In the future, we will test the same k-NN algorithm but using different proximity models based on finer granularity metadata extracted about the content of attributes in the datasets. We also seek to improve our algorithm with semantic analysis of values found in the attributes to complement the syntactical comparisons we compute in the proximity models.

References

1. Algergawy, A., Massmann, S., Rahm, E.: A clustering-based approach for large-scale ontology matching. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 415–428. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23737-9_30
2. Algergawy, A., Schallehn, E., Saake, G.: A schema matching-based approach to XML schema clustering. In: Proceedings of the International Conference on Information Integration and Web-based Applications & Services, pp. 131–136. ACM (2008)
3. Alserafi, A., Abelló, A., Romero, O., Calders, T.: Towards information profiling: data lake content metadata management. In: DINA Workshop, ICDM (2016)
4. Alserafi, A., Calders, T., Abelló, A., Romero, O.: DS-prox: Dataset proximity mining for governing the data lake. In: Beecks, C., Borutta, F., Kröger, P., Seidl, T. (eds.) SISAP 2017. LNCS, vol. 10609, pp. 284–299. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68474-1_20
5. Baralis, E., Cerquitelli, T., Chiusano, S., Grimaudo, L., Xiao, X.: Analysis of Twitter data using a multiple-level clustering strategy. In: Cuzzocrea, A., Maabout, S. (eds.) MEDI 2013. LNCS, vol. 8216, pp. 13–24. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41366-7_2

6. Friedman, J., Hastie, T., Tibshirani, R., et al.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Stat. **28**(2), 337–407 (2000)
7. Gallinucci, E., Golfarelli, M., Rizzi, S.: Schema profiling of document-oriented databases. Inf. Syst. **75**, 13–25 (2018)
8. Han, E.-H.S., Karypis, G., Kumar, V.: Text categorization using weight adjusted k-nearest neighbor classification. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, pp. 53–65. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45357-1_9
9. Hentech, H., Gouider, M.S., Farhat, A.: Clustering heterogeneous data streams with uncertainty over sliding window. In: Cuzzocrea, A., Maabout, S. (eds.) MEDI 2013. LNCS, vol. 8216, pp. 162–175. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41366-7_14
10. Lee, M.L., Yang, L.H., Hsu, W., Yang, X.: XClust: clustering XML schemas for effective integration. In: Proceedings of the International Conference on Information and Knowledge Management, pp. 292–299. ACM (2002)
11. Mahmoud, H.A., Aboulnaga, A.: Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 411–422. ACM (2010)
12. Manning, C.D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval, no. c (2009)
13. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
14. Shvaiko, P.: A survey of schema-based matching approaches. J. Data Semant. **3730**, 146–171 (2005)
15. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education, New York (2006)
16. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data wrangling: the challenging journey from the wild to the lake. In: 7th Biennial Conference on Innovative Data Systems Research CIDR 2015 (2015)

Lavoisier: High-Level Selection and Preparation of Data for Analysis

Alfonso de la Vega(✉), Diego García-Saiz, Marta Zorrilla, and Pablo Sánchez

Software Engineering and Real-Time, University of Cantabria, Santander, Spain
{delavegaa,garciasad,zorrillm,p.sanchez}@unican.es

Abstract. Most data mining algorithms require their input data to be provided in a very specific tabular format. Data scientists typically achieve this task by creating long and complex scripts, written in data management languages such as SQL, R or Pandas, where different low-level data transformation operations are performed. The process of writing these scripts can be really time-consuming and error-prone, which decreases data scientists' productivity. To overcome this limitation, we present *Lavoisier*, a declarative language for data extraction and formatting. This language provides a set of high-level constructs that allow data scientists to abstract from low-level data formatting operations. Consequently, data extraction scripts' size and complexity are reduced, contributing to an increase of the productivity with respect to using conventional data manipulation tools.

Keywords: Data selection · Domain-specific languages · Data mining

1 Introduction

Despite the current popularity of data mining techniques, executing data mining processes still requires dealing with multiple low-level details [15]. Consequently, data scientists must work at a very low abstraction level, which decreases their productivity. This phenomena is noticeable from the very first stages of a data mining process, where data to be analysed is selected and processed to be digested by data mining algorithms. Most of these algorithms require their input data to be arranged in a very specific two-dimensional tabular format, where all the information related to each entity under analysis must be placed in a single row. For instance, if we were analysing businesses, all the information about each one of these businesses has to appear in a single row of the table describing input data. This means that these algorithms cannot work with hierarchical or linked data such as JSON files, XML files, or relational tables connected by means of foreign keys, which are common examples of representations in which information is typically made available. Therefore, to execute a data mining algorithm, we first need to transform data stored in these representations into the specific tabular format that these algorithms can process.

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 50–66, 2019.
https://doi.org/10.1007/978-3-030-32065-2_4

Data scientists achieve this transformation task by creating long and complex scripts, written in data management languages such as SQL (*Structured Query Language*) [1] or Pandas (i.e. a well-known Python data manipulation library) [10]. These scripts extract data from the sources and, through a set of low-level operations, arrange these data into the required tabular format. The correct elaboration of these scripts, which is very important for the outcome of any analysis [3,11], can be a tedious, time-consuming and error-prone process.

To alleviate this situation, we present *Lavoisier*, a language that provides a set of high-level constructs to specify which data, among the available in a certain domain, should be included in a concrete analysis. These high-level constructs are automatically transformed into a set of data transformation operations that arrange the selected data into an appropriate tabular format. Therefore, using this language, data scientists can focus on the selection of relevant data for an analysis and forget about the technical details of the process required to transform the selected data, which contributes to increase their productivity.

To define Lavoisier, we needed to determine how the data available in a domain is represented. We opted for using object-oriented models, since these models are nowadays widely used to specify *domain models* [6].

Expressiveness and effectiveness of Lavoisier were assessed using a data mining open challenge belonging to the domain of business reviews [16]. Using this challenge, we specified different data selection and transformation processes. Then, we compared the generated Lavoisier specifications with their SQL and Pandas counterparts. As a result of this comparison, we concluded that Lavoisier's dataset specifications are more compact, less verbose and allow working at a higher abstraction level.

After this introduction, the article is structured as follows: Sect. 2 exposes the motivation behind this work. Section 3 presents the different features of the Lavoisier language. Next, Sect. 4 comments on related work, and Sect. 5 discusses strengths and weaknesses of Lavoisier. Finally, Sect. 6 summarises this article and outlines future work.

2 Case Study and Problem Statement

This section describes with more detail the motivation behind this work. To illustrate it, we used the *Yelp Dataset Challenge*, which is introduced next.

2.1 Running Example: The Yelp Dataset Challenges

Yelp is a company that provides an online business review service. Using this service, owners can describe and advertise their businesses and customers can write their opinions about these businesses. Yelp periodically collects and makes available bundles of data for academic usage in the form of data analysis challenges[1]. We use these data and challenges as running example throughout this article.

[1] https://www.yelp.com/dataset/challenge.

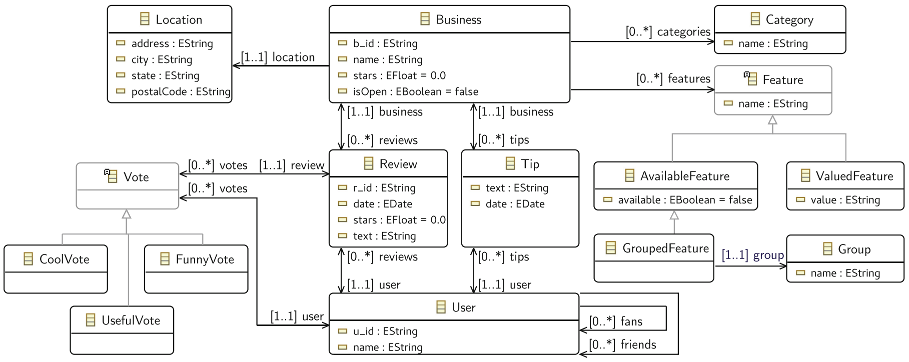


Fig. 1. Conceptual Model of the Yelp Dataset Challenge.

Yelp provides their data as a bundle of interconnected JSON (*JavaScript Object Notation*) files. To help visualise these files, we inferred the conceptual object-oriented model to which these files would conform to. This model is depicted in Fig. 1.

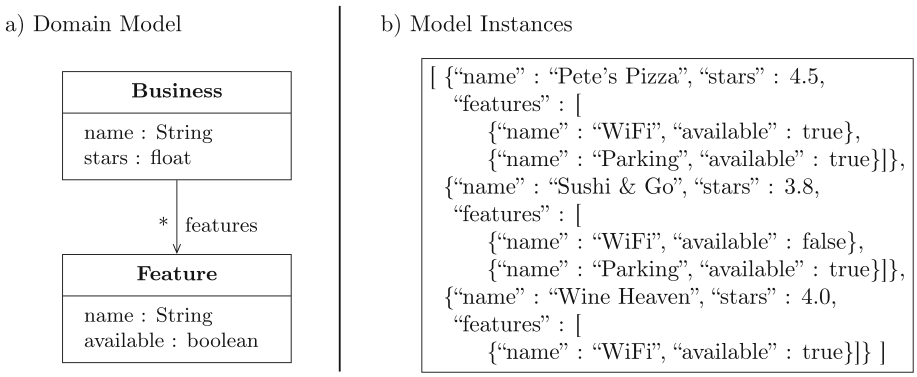
As it can be seen, for each registered *business*, Yelp provides information about its *location*; the different *features* it offers, such as the availability of Wi-Fi or a smoking area; and the *categories* which best describes it, e.g., Cafes, Restaurant, Italian, Gluten-Free, and so on. *Users* can *review* these businesses, rate them and introduce a text describing their experience. Additionally, users can write *tips*, which are small pieces of advice about a business, such as *do not miss its salmon!* Yelp also provides some social network capabilities, so users can have *friends* or *fans*. Users can also receive *votes* in their reviews in case other users found these reviews *funny*, *useful* or *cool*.

Using these data, Yelp proposes as challenges analysis tasks like identifying reasons behind a business becoming successful, or finding what kind of opinions are most likely to set a new trend.

2.2 The Data Reformatting Problem

Before executing a data mining algorithm over Yelp's data to, for instance, seek for the reasons that made a business successful, we need to transform the data to be used for that analysis into the previously mentioned tabular format. It must be remembered that this format, in addition to being tabular, imposes a non-trivial constraint: all the information about each instance of the domain entity being analysed must be placed in a single row of the data table to be analysed. In the data mining community, this table is usually referred to as a *dataset*.

To clarify this problem, let us suppose that, in the context of the Yelp challenge, we want to identify business features, or combinations of features, that might lead a business to have a high stars rating. We decide to use as information for the analysis the businesses name, so they can be identified; their stars



```
[ {“name” : “Pete’s Pizza”, “stars” : 4.5,
   “features” : [
       {“name” : “WiFi”, “available” : true},
       {“name” : “Parking”, “available” : true}]},
  {“name” : “Sushi & Go”, “stars” : 3.8,
   “features” : [
       {“name” : “WiFi”, “available” : false},
       {“name” : “Parking”, “available” : true}]},
  {“name” : “Wine Heaven”, “stars” : 4.0,
   “features” : [
       {“name” : “WiFi”, “available” : true}]} ]
```

Fig. 2. (a) Yelp model fragment; (b) JSON data conforming to (a).

BName	Stars	WiFi	Parking
Pete’s Pizza	4.5	true	true
Sushi & Go	3.8	false	true
Wine Heaven	4.0	true	

Fig. 3. A tabular dataset of businesses’ data.

rating as a success measure; and their set of available features[2]. Figure 2(a) shows the fragment of the original domain model that contains this information, whereas Fig. 2(b) shows some data, in JSON format, conforming to this model fragment. As it can be seen, this information is hierarchical, and each business might include several features nested in their data structure. Therefore, we need to face the problem of how to convert this hierarchical information into a single vector of data that can occupy a dataset row, such as depicted in Fig. 3.

To address this formatting problem, data scientists currently rely on data management languages or libraries, such as SQL [1] or Pandas [10]. Using these technologies, data scientists perform several low-level operations, such as *filters*, *joins*, *pivots* or *aggregations*.

As an example, Listing 1.1 shows a SQL script where, by employing a *CASE* operator in an aggregation query, the tabular representation of Fig. 3(a) can be obtained from the equivalent relational tables to the domain model fragment of Fig. 2(a). As it can be seen, this script is quite complex for this toy example and it does not scale appropriately, since we need to add a new aggregation query per each new feature included to the system. Thus, this strategy might be prohibitive when processing a larger number of features. The same or similar problems are found when other data management languages, such as Pandas [10] or R [12], are used.

[2] The *Features* inheritance of Fig. 1 has been omitted from this initial example.

Listing 1.1. SQL flattening operation using aggregation queries and the case operator.

```
select b.b_id, b.name, b.stars,
       max(case features.name when 'Parking'
           then available end) as feature_parking,
       max(case features.name when 'WiFi'
           then available end) as feature_wifi,
       max(case features.name when 'Smoking'
           then available end) as feature_smoking
from business as b
left join features on b.b_id = features.b_id
group by b.b_id, b.name, b.stars
```

To alleviate this situation, we studied the following ideas: first, a mechanism to increase the abstraction at which users select data for an analysis should allow working over a high level representation (i.e. a conceptual model) of the available data. From the different notations that could be used to represent these data, we focused on object-oriented models such as the one in Fig. 1, because of their popularity to define domain models [6]. However, other conceptual model notations could also have been employed.

Second, if we had a way to automatically convert selections performed by data scientists over the conceptual model into the dataset format required by data mining algorithms, then these data scientists would be relieved of this repetitive and prone-to-errors task.

As a result of our work, we have created a new language, called *Lavoisier*, which provides high-level declarative primitives to specify which elements of an object-oriented domain model should be considered for an analysis. These specifications are then processed by the language interpreter, which automatically transforms [14] the employed primitives into a set of low-level data transformation operations that produces as output the required tabular representation. This language is described in the next section.

3 Lavoisier: Dataset Extraction Language

We describe Lavoisier by example, showing how the language can deal with data transformation scenarios of increasing complexity.

3.1 Single Class Selection

Listing 1.2 shows the most basic Lavoisier snippet. In Lavoisier, data selection processes are expressed by defining *dataset specifications*, which are expressed with the *dataset* keyword followed by the *name* for the dataset (for instance, *yelp_reviews* in Listing 1.2), and a *body* block surrounded by braces (lines 1–3).

A dataset specification must always declare a *main class* (line 2), which is the class whose instances would be placed in the rows of the output dataset.

Listing 1.2. Lavoisier's simplest dataset specification.

```
1  dataset yelp_reviews {
2    mainclass Review
3  }
```

By default, if we do not provide any further information, Lavoisier considers that all attributes of a class must be included in the output dataset, whereas references to other classes are excluded. Taking into account these considerations, the process for generating a dataset from the specification of Listing 1.2 is straightforward: a table is created with a column for each attribute contained in the *Review* class. Then, each instance of the *Review* class is processed, generating a new table row where the instance attributes are assigned to the corresponding columns.

If we do not want to include all attributes of a class in the resulting dataset, we must specify those of our interest as a list after the class name, between brackets, such as illustrated in Listing 1.3. In this example, just the *r_id* and *stars* attributes would be selected. The resulting dataset is created following the previous process, but this time only columns for those attributes specified in the list between brackets would be generated.

Listing 1.3. Dataset specification with attribute selection.

```
1  dataset yelp_reviews {
2    mainclass Review [r_id, stars]
3  }
```

Finally, it might also be the case that we are not interested in including all instances of a class in a particular dataset. For these cases, Lavoisier provides filters. A *filter* is specified after a class name using the *where* keyword, and it must contain a predicate that is evaluated for each class instance. If the predicate evaluates to *true*, then the instance is processed; otherwise it is discarded. Listing 1.4 shows an example of filter for considering just reviews belonging to businesses placed in Edinburgh (line 3).

Listing 1.4. Dataset specification with a filter.

```
1  dataset yelp_reviews {
2    mainclass Review [r_id, stars]
3        where business.location.city = "Edinburgh"
4  }
```

3.2 Single-Bounded Reference Inclusion

The inclusion of references is more challenging than the selection of attributes because of two main reasons: (1) referenced types might have their own extra attributes and nested references that we may need to manage; and (2) data of the referenced class must be combined with the main class in order to create a unique table containing all the information. The complexity of this data merging process depends on the cardinality of the included references. We distinguish between two cases: single-bounded references, i.e., references with 1 as upper bound; and multi-bounded ones, i.e., references with an upper bound greater than 1.

Single-bounded references can be incorporated to a dataset through the *include* keyword. For example, in Listing 1.5, line 3, the *user* reference of the *Review* class is included in the dataset through the *include user* statement. As in previous section, if no extra information is given, all attributes of the included class (e.g. *User* in this case) would be placed in the output dataset.

Listing 1.5. Reviews data with some attribute and reference selections.

```
1 dataset yelp_reviews {
2   mainclass Review [r_id, stars]
3   include user
4 }
```

The data merging process in the case of single-bounded references is rather simple. Attributes of the reference class are added to the main class, and then the resulting class is processed as a single one, following the procedure described in the previous section. Conceptually, this would be equivalent to performing a *left outer join* between the main class and the referenced class, and then transforming the resulting class to a table.

3.3 Multi-bounded Reference Inclusion

The problem when processing references with upper bounds greater than one is that each instance of the main class might be related with several instances of the referenced class. This can be seen in Fig. 2(a), where each business has several associated features. To combine both classes, as all the information about each instance of the main class must be placed in a single row, we need to spread the information about the referenced instances on different columns of the output table, so that it ends up flattened. This is, for each business, we need to spread the information about each potential feature it may have over different columns. To achieve this goal, Lavoisier performs a *join* between both classes, followed by a *pivot*. A pivot is an operation that reshapes tabular data, so that data information scattered over several rows is condensed in a single one by generating new columns.

These concepts are illustrated with the help of Fig. 4. Figure 4(a) shows, for the data of Fig. 2(b), the result of joining each business with their associated features. As it can be seen, after performing this join several rows refer to the

(a) After join: business information ends in several rows.

BName	Stars	Feature	Available
Pete's Pizza	4.5	WiFi	true
Pete's Pizza	4.5	Parking	false
Sushi & Go	3.8	WiFi	false
...	...	...	....

(b) After join and pivot: Information of each business placed in a single row.

BName	Stars	WiFi	Parking
Pete's Pizza	4.5	true	true
Sushi & Go	3.8	false	true
Wine Heaven	4.0	true	

Fig. 4. Data merging by combining *join* and *pivot* operations.

same business, which violates the constraint of requiring all the information of each business in a single row. To solve this issue, we execute a pivot, which accepts three parameters: a set of static properties, a set of pivoting properties, and a set of pivoted properties. In our case, the static properties will be always the properties of the main class, the pivoting properties a set of properties that can identify each instance of the referenced class, and the pivoted properties the remaining properties of the referenced class. For the concrete example of Fig. 4, we used as pivoting property the *name* of the feature, so *available* would be the pivoted property.

With these parameters, the pivot works as follows. Firstly, the structure of the output table is determined. To do it, all static properties, which would be *BusinessName* and *Stars* in our example, are added as columns to the output table. Then, all distinct tuples of the pivoting properties are calculated. In the example, these tuples would be (*WiFi*) and (*Parking*). Then, each one of these tuples is combined with the pivoted properties to conform the new set of columns that should be created. If we perform this process, we would add the *WiFi_available* and *Parking_available* columns to the output table.

Once this table structure is created, it is populated with data. Each distinct tuple for the static properties is added as a new row in the output table. In our example, the tuples *(Pete's Pizza, 4.5)*, *(Sushi & Go, 3.8)*, and *(Wine Heaven, 4.0)* would be added. At this point, these rows are incomplete, as the newly created columns, i.e., *WiFi_available* and *Parking_available*, would still need to be filled. For this purpose, the pivot operator checks, for each newly added row, whether the input table contains a row that has the static values plus the corresponding pivoting column value. If such a row is found, the pivot operator copies the value of the pivoted column of this row into the corresponding cell of the output table. For instance, continuing with our example, to calculate the value of the *WiFi* column for the *(Pete's Pizza, 4.5)* row, the pivot operator checks if the input table has a row containing the values *(Pete's Pizza, 4.5, WiFi)*. If so, the value of the *available* column for that row is copied into the cell corresponding to the *WiFi_available* column.

Listing 1.6. Including references of different cardinalities.

```
1 dataset yelp_reviews {
2   mainclass Business
3   include location
4   include features by name
5 }
```

As it can be seen, by pivoting the result of a join between classes with one or more properties of the referenced class as pivoting column, an appropriate tabular representation of the input data is obtained. Oppositely to single-bounded references, in this case we need to explicitly specify which property or properties of the referenced class should be used as pivoting ones. For that reason, Lavoisier requires these properties to be specified after including an unbounded reference, such as illustrated in Listing 1.6 (line 4), which corresponds to the Lavoisier specification for producing the dataset of Fig. 4(b). These pivoting properties are indicated as a list after the *by* keyword.

Listing 1.7. Lavoisier's *calculate* construct to perform aggregations.

```
dataset yelp_businesses {
  mainclass Business[name, stars]
  calculate numReviews
    as count(reviews)
  calculate numPositiveReviews
    as count(reviews) where stars >= 4
}
```

Finally, it is worth to mention that Lavoisier allows selecting multiple references from a type by using an include primitive for each reference, such as it is shown in Listing 1.6 (lines 3–4). If that is the case, each reference is processed individually by the interpreter with the appropriate pattern for each case, resulting in the generation of a new set of columns for each processed reference.

3.4 Aggregated Values

It could be the case that we are not interested in analysing individually each instance of an unbounded reference, but in summarising the information contained in these instances by means of some formulae. For this purpose, Lavoisier provides the *calculate* primitive and some pre-built aggregation functions. Listing 1.7 shows an example where this primitive is used.

In this example, together with the business name and stars, two columns, *numReviews* and *numPositiveReviews*, are added to the output dataset. These columns collect global information about each business's review, specifically, the number of reviews received by each business, and the number these reviews

Listing 1.8. Reviews data with nested references.

```
1 dataset yelp_reviews {
2   mainclass Review [r_id, stars]
3   include user
4   include business {
5     include location[address, postalCode]
6     include categories by name
7   }
8 }
```

that were positive (i.e. those that at least have a 4-stars rating). To make these calculations, the *count* function and a filter are used.

3.5 Nested References

As commented, when including a reference, the default behaviour involves appending all attributes of the target class in the output dataset, whereas references are ignored. Nevertheless, we might be interested in customising which information of a reference is included to, for instance, omit certain target class' attributes, or to include some nested reference to yet another class. We can do this by adding a block to the *include* construct. Inside this block, we can use the same modifiers as in the main class to include more references. These *include* blocks can be consecutively used to select references up to the required nesting level for the extraction.

For instance, in Listing 1.8, inside the *business* inclusion block, the *location* and *categories* references are included (lines 5 and 6). From the *location* reference, just the *address* and *postal code* attributes are selected. If the *Location* or *Category* classes had references of their own, we could also select them by using new inclusion blocks.

3.6 Inheritance Management

When a class that appears in a Lavoisier specification is included in an inheritance hierarchy, as expected, all the attributes and references of their superclasses are considered attributes and references of that class. Therefore, they can be managed by Lavoisier as regular ones. In addition, all attributes belonging to subclasses are also merged with the class being processed, following a strategy similar to the *Single Table* pattern of Object-Relational Mappers (ORMs) [8].

We can customise this default behaviour to: (1) include just some subclasses in the resulting dataset; (2) include some references belonging to certain subclasses; or (3) select just a subset of attributes from certain subclass for their inclusion. For these purposes, a new pair of constructs were introduced: *as* and *only as*. Listing 1.9 shows a dataset specification example where the *only as* construct is used.

Listing 1.9. Selection of only two subclasses from the inheritance for the reduction.

```
1 dataset yelp_businesses {
2   mainclass Business[name, stars]
3   include features by name {
4     only as GroupedFeature {
5       include group
6     }
7   }
8 }
```

In this example, *features* is a reference to a class, *Feature*, which belongs to an inheritance hierarchy. For the extraction, we only want to include information from the *GroupedFeature* subclass and we indicate it using the *only as* keyword inside an inclusion block. Once a subclass is added to the inclusion block, we can customise it as in the case of the main class or a reference. For instance, in Listing 1.9, a *include* clause is used to specify that the *group* reference of the *GroupedFeature* must be also added to the output dataset (line 4).

If we were interested in customising a subclass without excluding other subclasses from the output dataset, we must use the *as* keyword alone, instead of *only as*. For example, in Listing 1.9, using *as* in line 4, both the *ValuedFeature* and *AvailableFeature* subclasses would be also included in the output dataset, while at the same time customising the inclusion of the *GroupedFeature* subclass.

3.7 Implementation

The code that conforms the Lavoisier language is available in an external repository under a free license[3]. Lavoisier has been implemented with the Xtext framework [7]. By using Xtext, we obtain a full-fledged editor where it is easy to include facilities such as syntax highlighting or autocompletion.

The execution of a Lavoisier script generates a *CSV (Comma-Separated Values)* file for each dataset specification. First, the instances of the main class to be included in this output dataset file are gathered, taking into account the specified filters. Then, the attribute and reference inclusion constructs are processed to determine the different sets of columns that must be generated, in the same order that they were defined. Lastly, these columns are calculated for each gathered instance of the main class, and placed as rows in the output dataset.

4 Related Work

To the best of our knowledge, this is the first language designed to select data from domain models and generate tabular datasets from them. Nevertheless, the issue of how to automatically reduce multi-relational data to a single-table

[3] https://github.com/alfonsodelavega/lavoisier.

structure that can be digested by data mining algorithms, which is known in the community as *propositionalisation*, has been previously addressed by different researchers [2,9,13]. Generally speaking, these approaches work as follows: starting from an entity of interest, e.g., Business of the Yelp case study, an algorithm randomly generates dataset columns by applying aggregation functions, such as *count*, *average*, *max* or *min*, over the relationships between the selected entity and other ones in the model.

This random exploration has the potential to discover previously unknown aggregate values that might be relevant for data analysis. On the other hand, domain experts and data scientists cannot have a fine-grained control of which data would be included in the output dataset. In addition, unbounded references cannot be analysed at the instance level, since their information needs always to be summarised by means of aggregation functions. This limitation might hamper finding patterns related to values of these individual instances, such as that most of successful pubs have happy hours from 20:00 to close. Moreover, it should be taken into account that this random exploration might exhibit scalability and performance problems. The exploration takes place over an enormous search space of candidate columns, from which many of them may not be useful at all.

Other researchers have tackled the problem from a different angle. Instead of focusing on producing tabular datasets from linked and hierarchical data, they have modified some data mining algorithms so that they accept data in their original structure as input, which is known as *Multi-Relational Data Mining (MRDM)* [5]. However, at the time of writing this work, most of these modified algorithms are not as powerful and versatile as those data mining algorithms that work with single-table datasets. While this is the case, we consider that a language like Lavoisier can be helpful for assisting data scientists in the generation of tabular datasets.

5 Discussion

Lavoisier allows data scientists to create scripts for data selection, extraction and preparation using a set of primitives that abstract from the low-level operations often used for this purpose. We comment here on the benefits offered by this language when compared to other state-of-the-art technologies that are commonly used to perform dataset extractions.

For example, we might be interested in studying business' reviews of the city of Edinburgh, to find whether the final score of a review is related to any of the following indicators: (i) the user performing the review; (ii) the business' location in the city; (iii) the features that the reviewed business offers (or not); and (iv) the number of positive reviews that the business has previously received. Moreover, in the case of *Grouped Features*, we might also want to study if the feature group has any influence on the review score. This data can be selected and tabulated using the script of Listing 1.10.

As commented over the paper, Lavoisier's primitives allow data scientists to rid of performing low-level operations such as filters, join and pivots manually.

Listing 1.10. Lavoisier script for analysing review scores.

```
 1 dataset yelp_reviews {
 2   mainclass Review
 3     where business.location.city = "Edinburgh"
 4   include user
 5   include business {
 6     include location[address, postalCode]
 7     include features by name {
 8       as GroupedFeature {
 9         include group
10       }
11     }
12     calculate numPositiveReviews
13       as count(business.reviews) where stars >= 4
14   }
15 }
```

If the same data selection and extraction task of Listing 1.10 had been specified using SQL or Pandas, we would have had to manually perform one pivot, nine joins and two filters. In addition, we should also have taken into account the order in which these operations are executed, and of storing the intermediate results. It is also worth mentioning that these low-level operations are more verbose than Lavoisier's primitives, which leads to larger and more complex scripts.

For instance, to process a single-bounded reference in Lavoisier, we only need to specify a keyword and the name of the reference, i.e., *include user* in line 4 of Listing 1.10. In the SQL counterpart, we would need to specify an SQL query over the tables to be joined and the name of the joining columns.

In the case of unbounded references, the situation becomes worse. First of all, unbounded references need to be processed by means of two operations, these are, a join and a pivot; whereas in Lavoisier we only need to specify the reference to be processed and the properties of the referenced class to be used as pivoting ones. In those data management languages that provide a full-fledged pivot implementation [4,10], a pivot operation can be specified in the best-case scenario by using two parameters: the set of static columns, and the set of pivoting columns. Nevertheless, most data management languages provides limited implementations of the pivot operator that add some extra complexity. For instance, SQL Server [4] extends the SQL standard with a proprietary version of the pivot operator, where the structure of the output table is not calculated by the operator itself, but it has to be manually specified. Listing 1.11 illustrates how an unbounded reference can be managed in SQL Server by pivoting the result of a join between two tables. As it can be observed, in the pivot operation (lines 8–11) we must explicitly specify the values of the pivoting properties for which new columns would be generated, these are, the *WiFi* and *Parking* available

Listing 1.11. SQL Server's pivot operation example.

```sql
select bName, stars, "Wifi", "Parking"
from (
    select b.name as bName, b.stars,
           f.name as fName, f.available
    from Business b, Feature f
    where f.business = b.id
) as p
pivot (
    max(p.available)
    for p.fName in ("Wifi","Parking")
) as pivotTable;
```

Table 1. Character count of SQL, Python Pandas and Lavoisier when performing different dataset extraction scenarios over the Yelp Dataset Challenge Data.

Scenario	Extraction Description	SQL	Pandas	Lavoisier
Single Class	Just Reviews	17	7	33
Unary Reference	Reviews and user	54	30	45
Multi-Bounded Reference	Business and categories	294	155	66
Multi-Bounded Reference	Business and AvailableFeatures	240	115	65
Inheritance	Features Inheritance	132	200	71
Inheritance	Business and Features	591	649	101
Nested References	Review, Business and Categories	359	181	82
Geometric mean of Lavoisier's script size reduction:		58.6%	36.2%	

features that a business may have in this example. Unfortunately, if new feature values are added to the system, we must update this script to deal with them, which is clearly a less maintainable solution as compared to Lavoisier or other implementations of the pivot operator. Finally, in the case where a data management language does not provide a pivot operator, different workarounds, such as the one illustrated in Listing 1.1, might be required. These workarounds clearly have a higher complexity than the alternatives offered by Lavoisier.

Summarising, as it has been discussed, Lavoisier helps reduce size and complexity of data selection and transformation scripts. This contributes to increase data scientists' productivity, who can use a more compact and high-level syntax, avoiding caring about technical details of the data transformation operations required to generate each particular dataset.

To provide with an objective validation of the discussion presented above, we are in the process of performing a more systematic evaluation where Lavoisier is compared against two well-known technologies often used for dataset extraction operations: the SQL language [1] and the Pandas data management library [10].

As a starting point in this evaluation, we have used language conciseness as an indirect indicator of the abstraction level provided by the language. In our case, a higher conciseness might indicate that some low-level operations are hidden to the end user by the language. So, we have assessed Lavoisier's better conciseness by measuring the script size in characters (ignoring whitespace) of Lavoisier, SQL and Pandas scripts for different dataset extraction scenarios of increasing complexity. These scenarios took place over data from the Yelp Dataset Challenge [16]. In the case of SQL and Pandas, as these technologies do not allow working directly against object-oriented domain models such as the one depicted in Fig. 1, their dataset extraction scripts operated over an equivalent relational database. This relational counterpart has been created following the typical object-to-relational transformation rules [8]. The obtained character-size measurements can be consulted in Table 1.

According to these results, Lavoisier can reduce scripts' sizes on average by 58% when compared with SQL, and by 36% with respect to Pandas. Lavoisier performs clearly better in those cases with unbounded references, nested structures or where dealing with inheritance hierarchies is required. In those cases, Lavoisier can achieve size reductions of up to ∼80%.

As commented, this is an ongoing work, and the current resources corresponding to the presented evaluation can be found in an external repository[4]. We are working on increasing the number of case studies employed in the evaluation, so that the obtained results can be better generalised no matter the domain where Lavoisier is being employed. Also, the usability and learning curve of the language are important aspects to evaluate when comparing the process of learning and using Lavoisier against existing solutions, so we plan to perform empirical experiments to determine whether end users find Lavoisier's high-level primitives easier to apply when carrying out dataset extraction operations.

6 Summary and Future Work

This work has presented Lavoisier, a language for assisting data scientists during the creation of datasets according to the format accepted by data mining algorithms. Firstly, the data selection and transformation problem was presented. As it was commented, data mining algorithms can only receive data arranged in a specific tabular format. Therefore, before executing a data mining algorithm, we need to select and rearrange the data to be analysed into the accepted format. Data scientists typically perform this task by writing scripts in a data management language such as SQL or Pandas, which means they have to perform several low-level data transformation operations manually and take care about their details. This leads to the creation of large and complex scripts, which might be hard to maintain.

As a solution to alleviate this problem, we have created a language that provides a set of high-level constructs for selecting data from object-oriented domain models. These constructs, when processed by the language interpreter,

[4] https://github.com/alfonsodelavega/lavoisier-evaluation.

are transformed in a set of low-level data transformation operations that generate tabular datasets ready to be digested by data mining algorithms. Lavoisier is able to deal with the different transformation scenarios that can be found in a object-oriented model, such as the processing of single-bounded references, multi-bounded references, nested structures or inheritance hierarchies.

When compared with data management languages, Lavoisier allows working at a higher abstraction level and reducing script size and complexity, achieving average reductions of $\sim$50%, and of up to 80% in some cases. Related to other research approaches, such as multi-relational data mining [5] or propositionalisation [9], Lavoisier provides a more fine-grained control of the data to be analysed, can deal with multi-bounded references at the instance level and allows using robust and well-known data mining algorithms.

As future work, we plan to perform some empirical experiments to assess Lavoisier' usability and syntax, as well as to study how to extend this language in order to cover other formalisms to represent domain data.

Acknowledgements. Funded by the University of Cantabria's Doctorate Program, and by the Spanish Government under grant TIN2017-86520-C3-3-R.

References

1. Beighley, L.: Head First SQL. O'Reilly (2007)
2. Boullé, M., et al.: A scalable robust and automatic propositionalization approach for Bayesian classification of large mixed numerical and categorical data. Mach. Learn. (2018). https://doi.org/10.1007/s10994-018-5746-9
3. Crone, S.F., Lessmann, S., Stahlbock, R.: The impact of preprocessing on data mining: an evaluation of classifier sensitivity in direct marketing. Eur. J. Oper. Res. **173**(3), 781–800 (2006). https://doi.org/10.1016/j.ejor.2005.07.023
4. Cunningham, C.: PIVOT and UNPIVOT: optimization and execution strategies in an RDBMS. In: International Conference on Very Large Data Bases, pp. 998–1009 (2004)
5. Džeroski, S.: Relational data mining. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 887–911. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-09823-4_46
6. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, Boston (2004)
7. Eysholdt, M., Behrens, H.: Xtext: implement your language faster than the quick and dirty way. In: Companion to the 25th Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications (SPLASH/OOPSLA), pp. 307–309 (2010). https://doi.org/10.1145/1869542.1869625
8. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
9. Knobbe, A.J., de Haas, M., Siebes, A.: Propositionalisation and aggregates. In: De Raedt, L., Siebes, A. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 277–288. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44794-6_23
10. McKinney, W.: Data structures for statistical computing in Python. In: Proceedings of the 9th Python in Science Conference, pp. 51–56 (2010)

11. Munson, M.A.: A study on the importance of and time spent on different modeling steps. SIGKDD Explor. Newsl. **13**(2), 65–71 (2012). https://doi.org/10.1145/2207243.2207253
12. R: The R Project for Statistical Computing. https://www.r-project.org/
13. Samorani, M.: Automatically generate a flat mining table with dataconda. In: IEEE International Conference on Data Mining Workshop, pp. 1644–1647 (2016). https://doi.org/10.1109/ICDMW.2015.100
14. de la Vega, A., García-Saiz, D., Zorrilla, M., Sánchez, P.: On the automated transformation of domain models into tabular datasets. ER FORUM **1979** (2017)
15. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical Machine Learning Tools and Techniques, 4th edn. Morgan Kaufmann Publishers Inc., San Francisco (2016)
16. Yelp: Dataset Challenge. https://www.yelp.com/dataset_challenge

Discovering Platform Government Research Trends Using Topic Modeling

Sun-Young Shin and Chang-Kyo Suh(✉)

Kyungpook National University, 80 Daehakro, Bukgu, Daegu, Korea
kitty13210@gmail.com, ck@knu.ac.kr

Abstract. Platform government is a key trend in the 4th industrial revolution. This paper presents an empirical analysis of 5810 articles in the Science Direct Database yielded by a search for the keyword 'platform government' from 1998 to 2017. Applying topic modeling to the article abstracts identified 9 key topics that were both representative and meaningful, and essentially corresponded to established sub-fields in platform government research. Measuring the variation of topic distributions over time revealed various rising research trends, such as data analytics and IoT, and a recent increasing popularity of business and governance. The identified key topics and Korean platform government projects were also compared. In conclusion, this study attempted to improve the identification, quantification, and understanding of the themes and trends in platform government over the last 20 years in order to provide a valuable tool for researchers and government agencies to make more informed decisions.

Keywords: Platform · Government · Topic modeling · Text mining · Research trends

1 Introduction

The reason why artificial intelligent (AI) was able to emerge as a general-purpose technology for the 4th industrial revolution was because the Internet of Things (IoT), big data, and the development of cloud computing all played an important role in the development and use of information. Thus, the 4th industrial revolution and spread of intelligent information technology have heralded the current transformation into a data-driven society. ICT (Information Communication Technology) is changing beyond the means of government efficiency to expand the platform of open innovation and creative cooperation. Data is generating information, this information is gathered to form knowledge, and that knowledge lives in an environment where decisions are based on such platforms. As a result, platform-based government operation and management are very important factors. In 2018, the Korean government emphasized the importance of creating an environment that includes envisioning and developing an infrastructure for Korea to develop a cycle of innovative growth. Moreover, a platform economy, which is critical for the 4th industrial revolution, requires an infrastructure, technology, and ecosystem for utilizing big data and AI [1]. Accordingly, this study analyzed the concept changes and policy point changes related to platform government using a content analysis of previous related research. Text mining was used to search the

https://doi.org/10.1007/978-3-030-32065-2_5

Science Direct DB for platform government-related research papers. Topic modeling was then applied to identify and compare the trends and changing focus of platform government-related research over time. As a result, this study is important for interpreting the meaning of market interest in platform government and its changing directions.

This paper is organized as follows. The Introduction explains why platform government is an interesting issue and government trend. The Background introduces the concept of topic modeling, Latent Dirichlet Allocation (LDA), and Structural Topic Model (STM) to quantify the topic distribution by aggregating the results from journal abstracts over time. Various findings are also presented about platform and platform government. The Research Design introduces the article/abstract data extracted from the Science Direct DB, the key research questions, and R package used for the topic modeling. The Research Results provide an extensive analysis of the extracted topics and word distributions using defined measures. The Discussion suggests some themes and topic trends in platform government over the last 20 years as a valuable tool for researchers and government agencies to make more informed decisions. Finally, the Conclusion summarizes our study and suggests some future research directions.

2 Background

2.1 Topic Modeling

Text mining is a typical method for analyzing data, covering a buzz analysis, topic frequency analysis, and topic modeling. In particular, Latent Dirichlet Allocation (LDA) topic modeling is an algorithm for locating topics in a large unstructured document set. LDA uses probabilistic clue values to group words with similar meanings in an unsupervised model [2]. Widely used in research to derive issues or track research trends and technology trends, LDA is a post-conference method that analyzes the distribution of words in a given document using probability calculations to generate a semantic coherent theme for the words and thereby determine the topics in that document. Figure 1 shows the graph models of the LDA documentation process, where a shaded node is an observation variable, a non-shaded node is a random variable, a non-node is a potential variable, a node-to-node relationship is shown by an arrow, and a rectangle shows the number of each document, topic, and word. This is a stochastic generation model for finding potentially meaningful topics in many documents. As such, the probabilities that the words in the text document are included in a certain topic are calculated, where each document is stochastically represented by several topics rather than one topic [3]. The subject of each document is composed of a small number of word mixtures, and topic modeling is performed assuming that each word is determined by the document subject. In other words, the algorithm performs a probabilistic calculation assuming that the words in a specific document constitute a set of specific topics and then extracts the results as a set of topic words. In topic modeling, posterior probabilities are inferred according to word generation conditions based on the assumption that words are not stand-alone (Dirichlet distribution), and they are expressed as probabilistic graph models [2, 4].

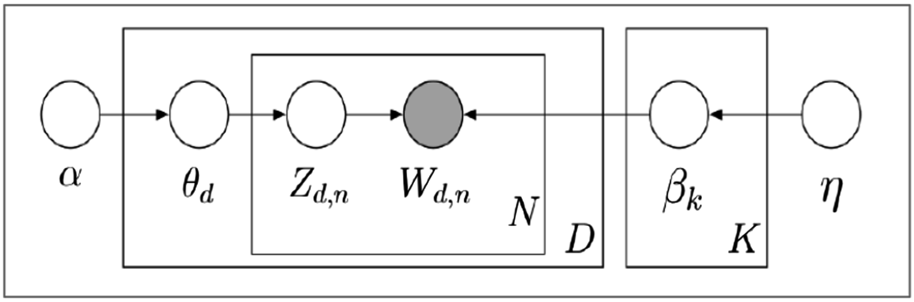


Fig. 1. Graph models of LDA

Qualitative research methods, such as literature reviews and expert evaluation, are conventionally used to identify new areas of interest and trend analyses. However, this requires a lot of time and resources to derive results from large amounts of information, plus it is essentially a subjective evaluation. Thus, a paradigm shift in qualitative research methodology is being attempted that is complementary to quantitative research methods, such as text mining [5]. Topic modeling techniques have already been applied to analyze scientific trends and hot topics in 2,620 abstracts from the PNAS (Proceedings of the National Academy of Sciences), and interestingly the results were reflected in the 2002 Nobel Prizes [4]. The Topics over Time (TOT) model also uses LDA topic modeling to view how topics change over time in literature and information research trends [3]. The current authors also applied the TOT model to identify the topics and time series trends of personal e-mails, the presentations at the Natural Information Processing Systems (NIPS) conferences from 1987 to 2003, and speeches by the U.S. President over 200 years. Mimno and McCallum [6] demonstrated that a topic model can also be useful in measuring influence when applying TNG (Topical N-Grams), a phrase-based topographic model, to 300,000 papers in the field of computer science collected from an automated human index system [6]. Meanwhile, Blei [7] used the LDA model to analyze 17,000 papers in the journals ‘Science’ and ‘Yale Law’, and showed that a topic model can be used for both social science and quantitative analyses [7]. Plus, Reich et al. [8] applied the Structural Topic Model (STM) that use LDA natively to compare time series topics and topics, and also utilized metadata for analyzing additional statistics [8, 9]. Gohr et al. [10] applied Probabilistic Latent Semantic Analysis (PLSA) to the SIGIR conference paper from 2000-2007, indicating that new words and topics emerged every year. Through Dynamic Topic Modeling, this paper shows the trend of key words on a 10-year basis for the Science magazine from 1880 to 2000. It also showed topics changes by the time such as Darwin and Einstein word [11]. Topic evolution maps were created from large corpus of 120,000 ACM papers from 1952 to 2007. Unlike general topic modeling, Jo et al. [12] created maps for each topic by using 290,000 citation links. This makes it easy for the person who starts this area research to recognize the emerging fields.

The STM method has been recently utilized to analyze various meta information, such as dates and politics, in documents. If the composition or weight of a topic changes in a specific way according to this kind of meta information, for example, if the proportion of the topic changes systematically over time, or if the organization of the topic changes according to the organization that produced the document, it is reasonable and accurate to estimate the composition and weight of the topic. The STM technique extracts topics using additional information when compared to the LDA model by inserting the assumption that various information in a document may affect the weight and composition of topics [6, 13]. In topic modeling, the number K of all the topics should be determined. In this case, STM provides a semantic coherence method for the optimal K by obtaining semantic consistency [14]. As such, topic modeling involves analyzing data to derive the major issues by period, studying the process of dynamic variation between individual issues, and analyzing the changes after extracting issues from various sources, such as news articles and Twitter. In order to grasp research trends in individual fields, topic modeling is utilized in various fields, such as library information science, information systems, aviation, industrial engineering, and marketing [5]. However, few studies have yet applied topic modeling to international research trends in order to contribute to government policies.

2.2 Platform and Platform Government

The original meaning of the word platform is a segmented area of land. In other words, a platform provides a framework for the values and interests that stakeholders can interrelate with each other, and can be defined as a medium for creating added value through mutual linkage [15]. A business is a platform where a facility or infrastructure is built up that can be used simultaneously by many customers. Through the platform, the interconnections of stakeholders can create network effects and form ecosystems that cooperate with one another. In other words, the interconnection function allows the subject to establish a platform so that consumers and suppliers can connect and exchange mutually owned values [16]. The use of the platform reduces costs or creates new added value that has not occurred before, which is the most important element of the platform. The platform defines an open infrastructure to encourage outside producers and consumers to interact and create value, and corresponding governance to create value for all participants [17, 18]. Platforms are defined from a manufacturing perspective to a transaction point in the ICT industry. Platforms are also sometimes expressed as components, modules, parts, objects, subsystems, interfaces, and structures, where products, services, and technology serve as intermediaries for groups of more than platform [19, 20].

A government provides various types of digital services to the public and builds platforms to use data and information as a source for new value-added production. Therefore, a ‘platform government’ is where the functions and roles of the government are constructed and used on a platform basis. This suggests the promotion of government digitization, reduced processes, and increased public participation [21]. Since O’Reilly [22] claimed Government as a Platform, the terms Open Gov and Gov 2.0 have been widely used [22]. Gartner’s technology for government, announced in 2018, is also an emerging technology related to platform government [23]. Platform-based government claims much more than existing technology-driven government by inducing public participation, including participation in civil society. Moreover, the existing developers, service providers, private platforms, and sustainable orchestra roles among the people are most important for platform-based governments. Platform-based governments also believe that the role of a small but intelligent government is important to create information from existing governments, namely various services using data [24, 25]. Currently, automation and efficiency are emphasized in the intelligent information society, and the demand for greater government ability to solve problems and efficiency in providing public services is increasing. As an alternative to this, the role and response of platform government are emphasized [26]. Using a platform, the government maximizes the effects of interacting with the private sector based on networks and data. In order to continuously create new value, the key is to create a platform that connects the stakeholders related to the public service in one chapter. To improve the problem-solving ability and efficiency of the government in various changing environments, a platform government needs to strengthen and cope with various governmental capacities [27]. When analyzing the present government service type in terms of setting up the government role as a platform, the results emphasize the importance of interactive communication so the opinions of the citizens are not made as policy by concentrating only on unilateral information provision. In order to participate in online citizenship, a paradigm shift is required, such as a platform government that can lead to innovation in government services based on public data [15, 28, 29]. The Korea National Informatization Strategy Committee, which discussed the direction of e-government, classified the components of a platform government as infrastructure, governance, and services including technology, and defined the requirements for each component [30].

A platform government is based on the establishment of collaboration and a governance system of the main categories of government and the people. This will require systematic support, interdepartmental collaboration, private-public cooperation, government and civilian cooperation, standardized work to induce a natural ecosystem, and the preparation of exemplary agreements. A previous study already classified government information based on the timing of informatization, which classified 1998 and thereafter as e(lectronic)-government, and t(ransformational)-government after 2005 as the time after the expansion of services, and l(ean)-government after 2010 with process establishment and the expansion of governance discussions as platform-based data and citizen participation became important [31]. There are some activity between citizen and government though platform after the e-government. Therefore collected data period is 1998–2017 for 20 years in this paper.

Gartner suggests a government maturity model that is divided into five phases: 1. Initial, 2. Development, 3. Defined, 4. Managed, 5. Optimizing, where the elements measuring platform government are identified as IT-centric, customer-centric, things-centric, and ecosystem-centric [23]. As shown in Fig. 2, there are nine main components of platform government: Customers/Business, Partners, Things, Employees, Customer-Experience, Ecosystems, Data & Analytics, IoT, and Information Systems [23].

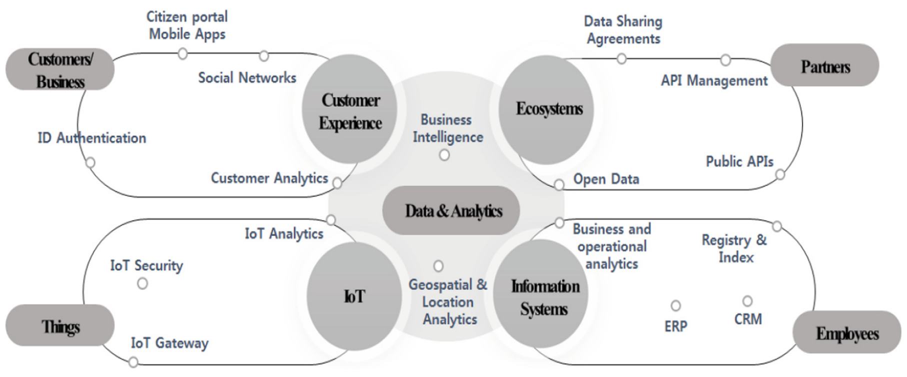


Fig. 2. Platform government components

3 Research Design

3.1 Research Topic

This study investigated the changes in platform government using data analysis approaches from previous studies. In particular, we looked at the emergence of the term "platform," which first appeared in terms of private services, along with how researchers have interpreted the meaning of "platform government", which could give meaning to the direction of the implementation of platform government. So it was discussed as research questions when the platform government was actively debated, the classification of the discussion topic, and how the classification changed over time. Therefore, the research project was conducted as follows:

Research Question 1: Is the discussion on platform government increasing?
Research Question 2: What are the key topics related to platform government?
Research Question 3: What are the changes over time in the key topics related to platform government?

3.2 Data Collection and Analysis

To analyze the research trends on platform government, an initial search for the keyword 'platform government' in the Science Direct DB over a 20–year period (1998–2017) collected 5,810 paper titles and abstracts. The research trends were then analyzed using topic modeling and a crawling engine, and the analysis infrastructure for collecting papers utilized the Korea Information Agency big data analysis system and text mining topic modeling packages, such as tm, ldavis, and stm based on R. For the data pre-processing process, non-English language removal and NLP (Natural Language Process) were used. This study also conducted a basic frequency analysis using existing research results in the field of platform government, and then extracted topics on platform government using a topic analysis text mining technique. STM is similar to LDA but provides more useful statistical data. In particular, statistical significance can be verified through the prevalence from STM. In this paper, prevalence was used the possibility of expressing the topic over time. It also provides a textprocessor() function to preprocess text for text minig. There are several graphical functions that are basically provided to compare topics. It is also known that STM is usually three times faster than LDA [8].

Based on the keyword ‘platform government’, 40,412 words were extracted from 5,810 papers that appeared during the last 20 years (1998–2017) after removing the stop words. Next, the relationship between the word frequency in each abstract and the subcategories of platform government was analyzed (Fig. 3).

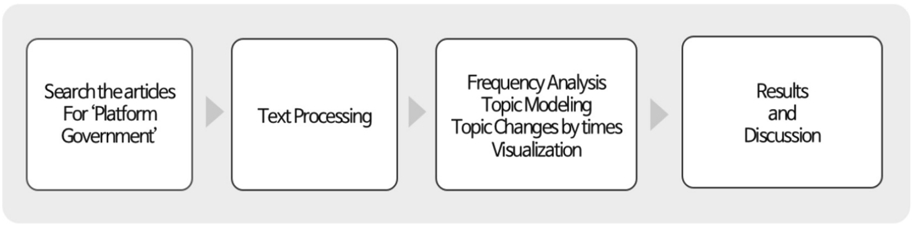


Fig. 3. Research and analysis procedure

4 Research Results

Research Question 1: Discussions about platform government are on the rise.
In terms of platform government as a keyword: 2017 (1,235 cases), 2016 (1,177 cases), 2015 (888 cases), 2014 (602 cases), 2013 (482 cases), 2012 (338 cases), and 2011 (242 cases). Platform-related contents appeared in earnest in 2005 (65 cases), and in 2008 (118 cases) the term platform government was already widely used. Research on platform government has since continued to increase, with the largest number of references in 2017 at 1,235 and 1,117 in 2016. Thus, active debate on platforms has been ongoing from 2010 until 2017. Moreover, the government has shown an increasing interest in platform government since 2008, which is consistent with the findings about UK platform government, which started in 2003 and became active after 2008 [25].

Research Question 2: Platform government-based data were classified into nine subtopics.

To determine the optimal number of topics for the topic modeling, two indicators of Held-out Likelihood and Semantic Coherence were simultaneously reviewed. The increase in Held-out Likelihood, considered a stable topic as the number increases, slowed down after nine, while the semantic coherence index, which shows a high value in the case semantic cohesion, indicated semantic cohesion at four and nine semantic when calculating the number of topics [4, 14] (Fig. 4).

In addition to dividing the components of platform government found in the platform government literature study, the value determined by the above Optimal Index was also applied, optimal K = 9 [11]. Topic models library was used for LDA and set α using K. The alpha value was set to $\alpha = 50/K$ by previous studies. Therefore, the value

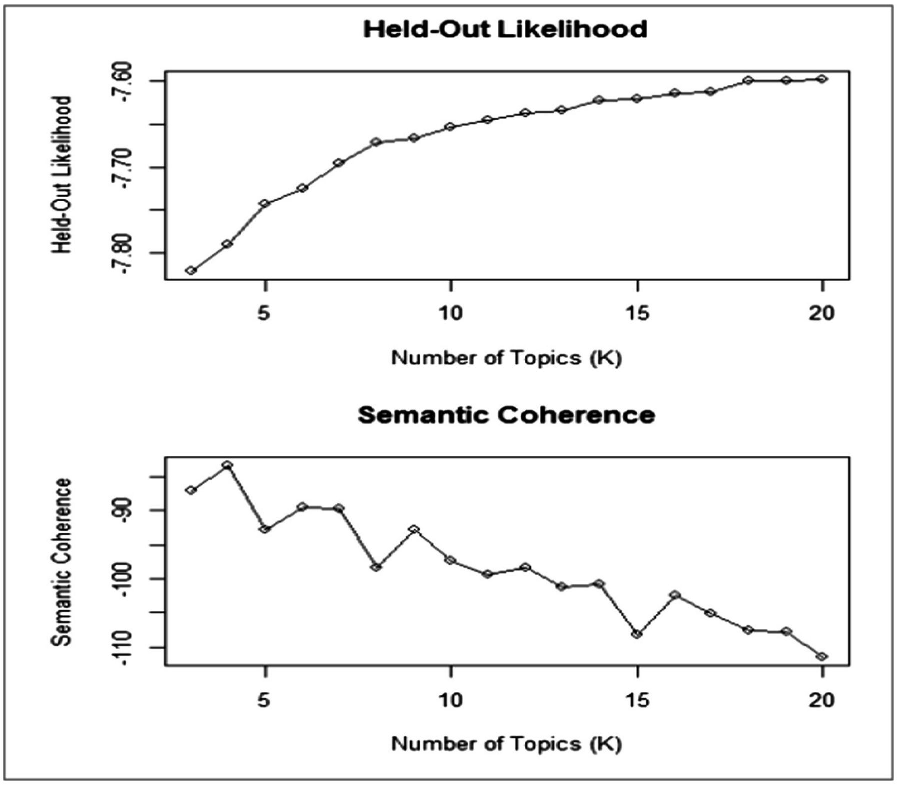


Fig. 4. Diagnostic values by the number of topics

of α was derived by substituting K derived from various indicators [9]. It took 205.71 s using LDA. Also EM (Expectation-Maximization) algorism use 76 times for making topic modeling. As the amount of documents grows, or the number of topics increases, so does the need for parallel processing. In this paper, we do not need to consider experiments or parallel processing such as WisdomLDA. WisdomLDA is one of methods should provide better performance for larger word topic table [32].

The distribution of the collected documents by topic was as follows. Topic 1: Customer Experience (11.77%), Topic 2: Things (10.76%), Topic 3: Partners (9.47%), Topic 4: Information Systems (13.87%), Topic 5: Data & Analytics (8.41%), Topic 6: IoT (7.57%), Topic 7: Business (12.69%), Topic 8: Ecosystems (14.22%) and Topic 9: Employees (11.23%). The keywords for each topic are listed below in the order of an increasing beta value per topic. The length of the small straight line in the table shows the proportions by topic. The word next to the topic shows the word with the highest frequency per topic (Fig. 5).

The distribution of each topic is shown in the figure below using LDAvis, together with the frequency of the keywords [33]. The distribution of topics derived from the LDAvis is shown on the left side of Fig. 6, and the top 30 words such as data, local, innovation, market, and governance are shown on the right side.

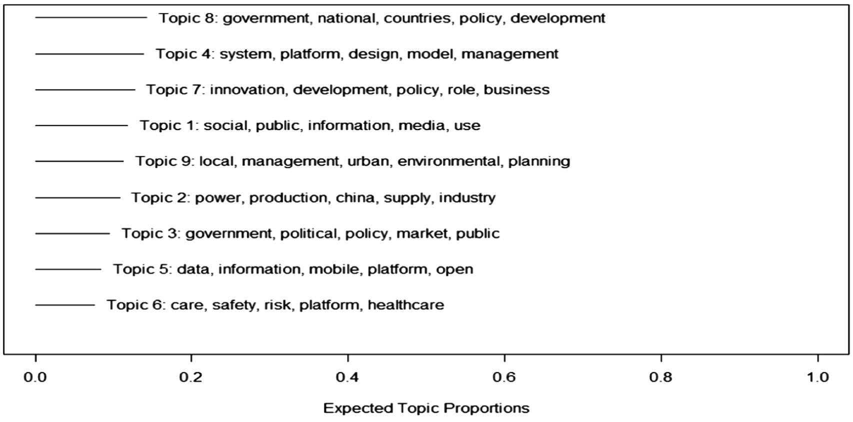


Fig. 5. Topics with keywords

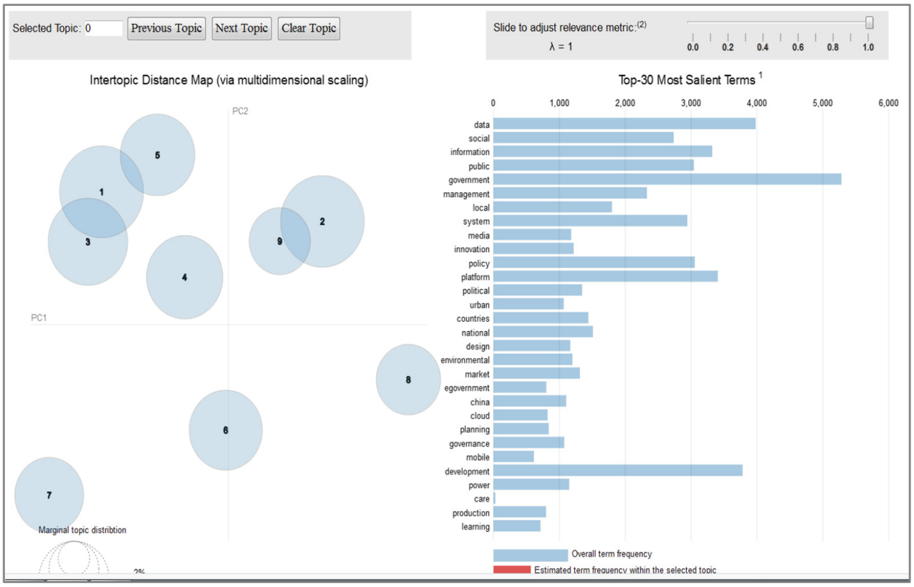


Fig. 6. Topics with LDAvis

Some details of the topics were as follows. Ecosystem (Topic 8) showed a high frequency of such words as national, policy, governance of platform government. The word probability distributions of education and economy related to platform governance were also high, which was attributed to the fact that this emphasizes the economic aspect of the platform structure [18, 34]. Plus, important discussions were included on governance-related policies, costs, and the role of government, along with the construction of information management systems, policies, services to manage information, linkage between government and industry, and an evaluation of the level of information governance construction. The Employees (Topic 9) topic analysis suggested an interim compatibility between central and local government services needed to be considered [35]. Plus, cloud computing and the emergence of a mobile computing environment have had a great impact on the platform government service environment. The distribution of e-government and its management often included discussions on the services of platform government [20], while discussions related to various stakeholders, such as suppliers and users, and control over information such as the Internet, security, and information access management were also found. The central government should be transformed into a cloud-centric sharing and cooperation structure, local governments should compete in good faith through responsibility and innovation, central government should be decentralized to become platform-based government, and local administrative information should be as open as possible to encourage the participation of residents [35]. Information Systems (Topic 4) highlighted many words, such as process, system, and management, along with references to systems, technology, design and organization for information services, information management, references to quality assurance, and utilization of data. Moreover, the Korean government and other governments are in the process of promoting the open use of public data and promoting the establishment of a data platform for scientific administrative and government innovation [36].

A comparison of the major topics can be summarized as follows. Topic 1 (Customer Experience) and Topic 5 (Data & Analytics) show that sharing and using information and word platform are connected and two topics, so data open and sharing by platform users is important. The comparison results between topics are shown in Fig. 7 below.

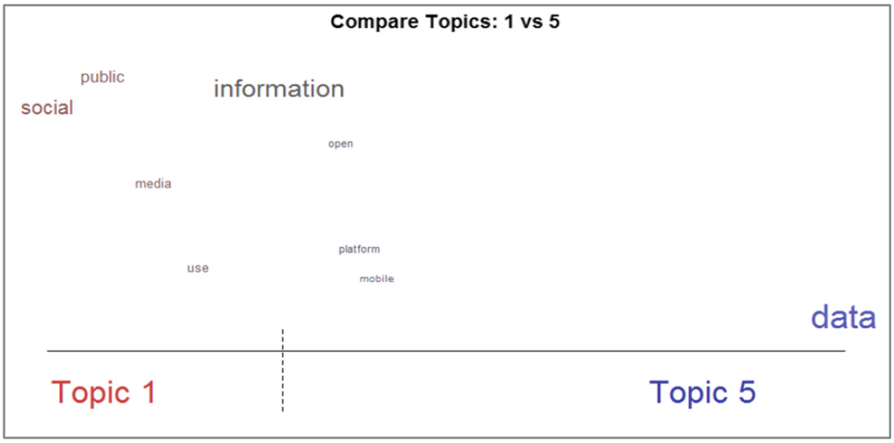


Fig. 7. Comparison between topics

Research Question 3: What are the time-based changes in key areas of platform government?

Topic modeling was used to investigate the temporal flow of each topic over 20 years using STM. Jassen and Estevez [31] divided government developments into three phases, e-government emphasizing technologies from 1998 to 2005, t-government focused on process from 2005 to 2010, and l-government using platforms for strong empowerment following the financial crisis in 2010 [31]. These time periods are shown by a dotted line in the table below. Topic 9 (Employees) shows a steady increase in interest over time, with a growing need to improve sustainability, such as cross-platform policies and relationships with stakeholders. Information Systems (Topic 4) shows a continuous decrease over time due to the appearance of new technologies, such as cloud computing and mobiles, indicating technical stability. This is consistent with the results of Jassen and Estevez [31] who reported a reduced importance of information systems as lean government uses platforms to engage the surrounding partners (Fig. 8).

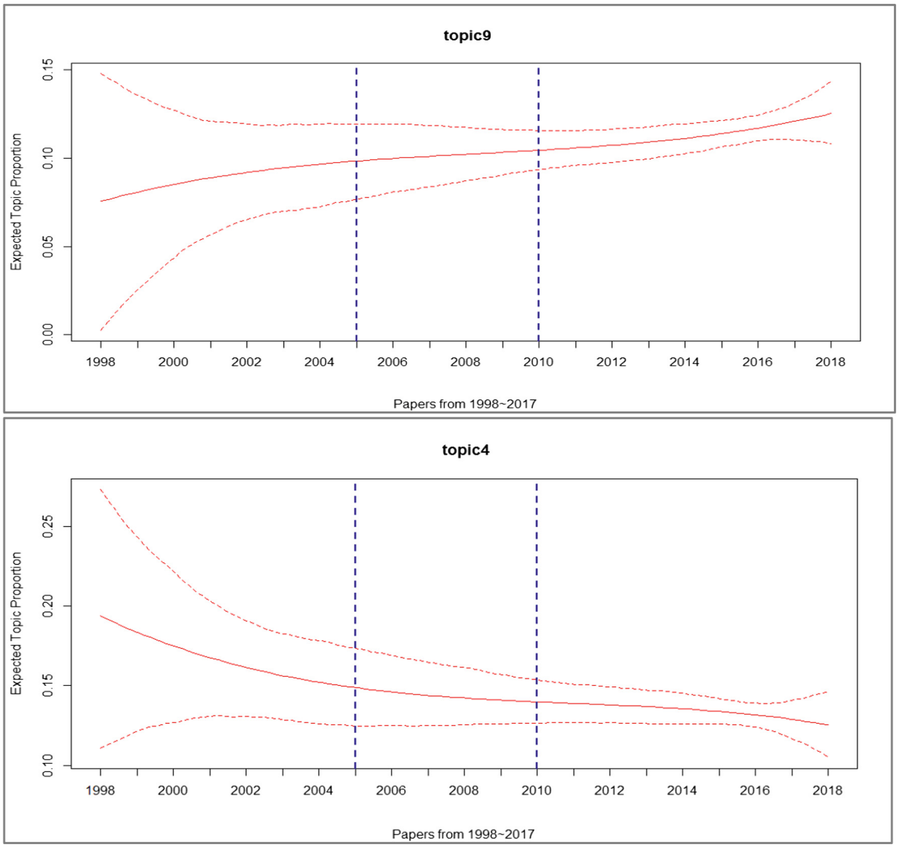


Fig. 8. Topics over time (1998–2017)

5 Discussion

This study investigated the trends of platform government based on a topic model analysis of interest areas and changes in related research. The trend analysis of platform government discussions showed a continuous increase from 2008, with an active increase from 2010 that has been sustained until 2017 [25]. This means a growing interest in platform government, a search for ways to effectively build and utilize platform government, and a market need for effective operation with a platform government base. A recent OECD report discussed platforms in the context of governments Going Digital, and that the role of future central government should be a platform for creating, evaluating, and coordinating competition rules between local governments, with major agendas at its heart: civic engagement, public data utilization, and sharing cloud, policy, and institutional practices [37].

For the topics related to research question 2, platform government was classified into 9 topics: Customer Experience, Things, Partners, Information Systems, Data & Analytics, IoT, Business, and Ecosystem. For Ecosystem, which is the governance part of platform government, there was a high frequency of the words policy, government, the roles related to government, and service interoperability. The topic modeling also revealed the need to make use of various stakeholders and organizations. For Information Systems, the keywords were process, system, and management, along with references to relevant systems and equipment, technology, structure and organization for information services, information management, and quality assurance. In a comparative analysis of the two topics, Topic1 (Customer Experience) and Topic 5 (Data & Analytics) were highly related with information and openness, indicating the importance of data sharing to platform users.

Research question 3 examined the trends of the topics, including expression over time using the STM prevalence in existing literature on government research. The results showed that existing platforms changed the characteristics of government in 2005 and 2010 [31, 34]. In particular, the field of interest shifted from platform construction to utilization of the data coming through the platform. This also reflects the shift of Korean government, which recently opened a data platform to provide more public information [36]. Moreover, the government use of social media is also effective for platform government [38]. The topic trends showed an increasing interest for each topic over 20 years, suggesting that monitoring and benchmarking the driving topics in other countries can be useful for making decisions. As shown in Fig. 9, the red arrows next to certain topics indicate a recent increase of interest based on the current topic analysis. The following table also shows a comparison of the topics derived in this study and pilot innovation projects to promote platform government by Korean government. This type of analysis can be valuable when making major policy decisions, especially in the case of longer trends and extensive data.

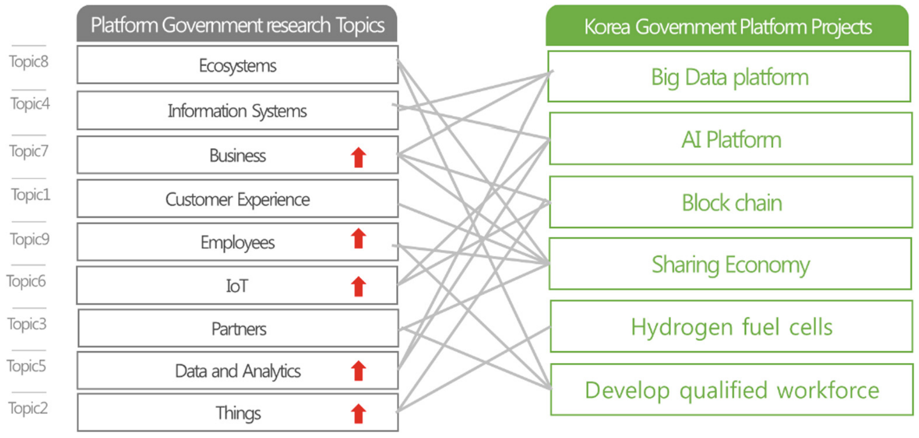


Fig. 9. Matching platform government topics and Korea platform projects (Color figure online)

6 Conclusion

Various efforts for more efficient government operation have been made using technological development. The present study analyzed the market perception of platform government by analyzing previous studies on platform government, analyzing the data of related research papers, and analyzing the operation of platform government and present future directions. A search for the keyword ‘platform government’ in the titles and abstracts of the Science Direct DB for the period 1998–2017 yielded 5,810 papers, and the research trends were then analyzed using text mining and topic modeling as tools for big data analysis.

First, the text mining showed an active increase in discussions on platforms from 2010 to 2017, indicating that platform government is being perceived as an important element and requires continued application and research on how to utilize it efficiently.

Second, the topic modeling revealed nine topics and multiple indicators for platform government. The keywords for each topic were determined using a probability distribution, and the relative interest among the topics was shown based on text mining. The implications for the growing ecosystem of platform government were focused on managing the relationship with stakeholders related to the progress of informatization, the regulation of industry based on new technology, and the role of the government and enhancing transparency. This is clearly a shift from discussing the provision of substantial services in a government-led infrastructure to balancing central and local government through the role of platforms and the range of control over the role of the government to reflect the needs of diverse stakeholders.

Third, the trend of each topic over 20 years showed a recent increased interest in platform government and the key focus of related research. In particular, platform government under the 4th industrial revolution emphasized the importance of IoT, considering the relationship between data and other stakeholders and objects. However, recent changes in the 4th industrial revolution are taking more advantage of data due to the development of relevant technologies, indicating that platform government should pay more attention to data rather than existing information and seek ways to open and utilize data so it can be used by more people and actively engage citizens.

In this paper, topic modeling is applied to the research area of platform government, but it can be border another area. Also future research will consider clustering or labeling using such as citation information and visualizing evolution map creation. The results of this study on platform government can also be used to understand the content landscape, as a starting point for discussions and inquiry of the field at large, and as a model for other fields to perform a similar analysis. Platform government is not a specific technology, so the number of documents was initially relatively small. While this data analysis of 20 years of research was able to provide the flow of one field, the results can also be extended by trawling thesis DBs, adding other papers, and comparing with other related data such as newspaper articles. Thus, further investigation and analysis are required in future studies.

References

1. The Innovation Growth Engine. Korea Government, Seoul (2018)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003). https://doi.org/10.1145/1015330.1015439
3. Wang, X., McCallum, A.: Topics over time: a non-Markov continuous-time model of topical trends. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 424–433, ACM (2006). https://doi.org/10.1145/1150402.1150450
4. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. Natl. Acad. Sci. **101**, 5228–5235 (2004). https://doi.org/10.1073/pnas.0307752101
5. Amado, A., Cortez, P., Rita, P., Moro, S.: Research trends on big data in marketing: a text mining and topic modeling based literature analysis. Eur. Res. Manage. Bus. Econ. **24**, 1–7 (2018). https://doi.org/10.1016/j.iedeen.2017.06.002
6. Mimno, D., McCallum, A.: Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. UAI, pp. 401–418, arXiv preprint (2008)
7. Blei, D.M.: Probabilistic topic models. Commun. ACM **55**, 77–84 (2012). https://doi.org/10.1145/2133806.2133826
8. Reich, J., Tingley, D., Leder-Luis, J., Roberts, M., Stewart, B.: Computer-assisted reading and discovery for student generated text in massive open online courses. J. Learn. Anal. **2**, 156–184 (2014). https://doi.org/10.2139/ssrn.2499725
9. Steyvers, M., Griffiths, T.: Probabilistic topic models. In: Handbook of Latent Semantic Analysis, vol. 427, pp. 424–440 (2007). https://doi.org/10.4324/9780203936399.ch21
10. Gohr, A., Hinneburg, A., Schult, R., Spiliopoulou, M.: Topic evolution in a stream of documents. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 859–870. Society for Industrial and Applied Mathematics (2009). https://doi.org/10.1137/1.9781611972795.74
11. Blei, D.M., Lafferty, J.D.: Dynamic topic models. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 113–120. ACM (2006). https://doi.org/10.1145/1143844.1143859
12. Jo, Y., Hopcroft, J.E., Lagoze, C.: The web of topics: discovering the topology of topic evolution in a corpus. In: Proceedings of the 20th International Conference on World Wide Web, pp. 257–266. ACM (2011). https://doi.org/10.1145/1963405.1963444

13. Margaret, E., Roberts, B., Dustin, T., Christopher, L., Jetson, L., Shana, K., David, G.: Rand structural topic models for open-ended survey responses. Am. J. Polit. Sci. **1**, 1–19 (2014). https://doi.org/10.1111/ajps.12103
14. Mimno, D., Wallach, H.M., Talley, E., Leenders, M., McCallum, A.: Optimizing semantic coherence in topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 262–272. Association for Computational Linguistics (2012)
15. Lathrop, D., Ruma, L.: Open Government: Collaboration, Transparency, and Participation in Practice. O'Reilly Media, Inc. (2010)
16. David, S.E., Hagiu, A., Richard, S.: How Software Platforms Drive Innovation and Transform Industries. The MIT Press, Cambridge (2006)
17. Baldwin, C.Y., Woodard, C.J.: The Architecture of Platforms: A Unified View. Platforms Markets Innovation, pp. 19–44 (2009). https://doi.org/10.2139/ssrn.1265155
18. Gawer, A.: Platforms, Markets and Innovation. Edward Elgar Publishing (2011). https://doi.org/10.4337/9781849803311
19. Choi, B.S.: Value creation platform. Donga Business Review: Platform Leadership, 2 (2012)
20. Galloway, S.: The Four: The Hidden DNA of Amazon, Apple, Facebook, and Google. Penguin (2017)
21. Deloitte Consulting: Gov2020: A Journey into the Future of Government (2015)
22. O'Reilly, T.: Government as a Platform. Open Government: Collaboration, Transparency, and Participation in Practice. O'Reilly Media, Sebastopol (2010)
23. A Digital Government Technology Platform is Essential to Government Transformation Gartner (2018)
24. Chun, S., Stuart, W.S., Eduard, H.: Government 2.0: making connections between citizens, data and government. Inf. Polity **2**, 1–9 (2010)
25. Brown, A., Fishenden, J., Thompson, M., Venters, W.: Appraising the impact and role of platform models and Government as a Platform (GaaP) in UK government public service reform: towards a Platform Assessment Framework (PAF). Gov. Inf. Q. **34**, 167–182 (2017). https://doi.org/10.1016/j.giq.2017.03.003
26. Master Plan for the intelligent information society. Ministry of Science and ICT (2017)
27. Lee, G.: An Exploration of Next Generation's eGovernment Focused on Platform Perspectives: The Possibilities and Limits in Korea. Korean Public Administration Association (2012)
28. Robinson, D.G., Yu, H., Zeller, W.P., Felten, E.W.: Government data and the invisible hand. Yale J. Law Tech. **11**, 159 (2008)
29. Danneels, L., Viaene, S., Van den Bergh, J.: Open data platforms: discussing alternative knowledge epistemologies. Gov. Inf. Q. **34**, 365–378 (2017). https://doi.org/10.1016/j.giq.2017.08.007
30. Shin, I.H.: e-Gov Platform Policy for Future Governments. The Korea National Informatization Strategy Committee (2012)
31. Janssen, M., Estevez, E.: Lean government and platform-based governance—doing more with less. Gov. Inf. Q. **30**, 1–8 (2013). https://doi.org/10.1016/j.giq.2012.11.003
32. Wisdom LDA: https://github.com/crabyh/WisdomLDA. Accessed 10 Aug 2019
33. Sievert, C., Shirley, K.: LDAvis: a method for visualizing and interpreting topics. In: Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, pp. 63–70 (2014). https://doi.org/10.3115/v1/w14-3110
34. Martin Kenney, J.Z.: Choosing a future in the platform economy: the implications and consequences of digital platforms. In: Proceeding of Conference Kauffman Foundation New Entrepreneurial Growth (2015). https://doi.org/10.4324/9781315717128-8
35. Lee, M.H.: Innovation policy for local government. CREN (2017)

36. Data Platform using Public Data. Ministry of the interior and safety of Korea (2018)
37. Measuring online Platforms and Cloud Computing in National Accounts. OECD (2018)
38. Panagiotopoulos, P., Bowen, F., Brooker, P.: The value of social media data: Integrating crowd capabilities in evidence-based policy. Gov. Inf. Q. **34**, 601–612 (2017). https://doi.org/10.1016/j.giq.2017.10.009

J2J-GR: Journal-to-Journal References by Greek Researchers

Leonidas Pispiringas[1], Dimitris A. Dervos[2](✉), and Georgios Evangelidis[1]

[1] University of Macedonia, 54636 Thessaloniki, Greece
lpispir@gmail.com, gevan@uom.gr

[2] International Hellenic University, 57400 Sindos, Greece
dimitris.dervos@gmail.com

Abstract. The Hellenic Academic Libraries Link (HEAL-Link, https://www.heal-link.gr/) operates since 1998. Its members include all the academic and/or research institutions operating under the auspices of the Hellenic Ministry of Education, plus the Academy of Athens, the National Library, and the Institute of Educational Policy. The present paper reports on the research effort undertaken in order to facilitate the decision-making process and the development of a HEAL-Link strategy plan. The focus is on scientific journals the Greek academic/research community publish their works with, and on the journals they reference. It is assumed that when a researcher makes a reference to an article in a scientific journal, the latter is considered to comprise a valuable source of information. The more references made to articles published with a given journal, the higher the value of the latter as a source of scientific information to the HEAL-Link user community. In order to exploit the aforementioned research goal, bibliographic metadata from nearly 63,000 research publications have been collected and pre-processed. The publications involve at least one (co-)author affiliated to a Greek academic/research institution. They span over a period of nine years (2010–2018), and nearly 10,000 journals. The bibliographic data include metadata on subject (discipline) area(s). The findings are made public via a Web application (http://j2j.heal-link.gr/). The latter utilizes interactive graphs that facilitate the interpretation of the relevant bibliography data, and it is seen to comprise a springboard for conducting further data analytic and mining tasks.

Keywords: Bibliography metadata · Journal-to-Journal references · Data visualization · Journal evaluation

1 Introduction

The HEAL-Link Consortium actively participates in projects, initiatives, and relevant developments that shape the scientific publications landscape in Greece. This is done in a way that is beneficial to its members and to the scientific community, in general. Its main goals are:

- to establish common policies on journals subscriptions, and promote the rational growth of its journals collection,
- to combine financial savings with access to a large number of electronic resources that meet the educational and research needs of its user community, and
- to negotiate the signing of joint subscription agreements with publishers, and provide/manage remote access to electronic resources and information services (the electronic journals included) to its members.

The HEAL-Link Consortium receives central funding from the Hellenic Ministry of Education to provide access to electronic scientific subscriptions to its members. In 2019, the latter represent 661,647 users, 13,536 of which are academic faculty members. During the 1998–2018 period, the HEAL-Link users have conducted nearly 85 million article downloads from a total of 17,100 scientific journal titles[1]. Access to the latter is made possible via license agreements made with twenty-three (23) e-journal publishers. To effectively negotiate with the latter, HEAL-Link elaborate on a strategy plan involving the evaluation of journals on the basis of their usage as reference (source) material by the Greek academic/research community.

In order for the HEAL-Link consortium to exploit its financial resources and meet the needs of its academic community when it comes to accessing scientific publications, a set of criteria have been depicted that are taken into consideration and shape HEAL-Link's actions in relation with the evaluation of the value a scientific journal represents to the Consortium's members and user community. Criteria that relate to the drafting of a strategic plan for HEAL-Link when it comes to negotiating with the publishers who provide access to electronic subscriptions. The scientific journals evaluation process emphasizes on the usage of each journal as reference (source) material by the Greek academic/research community. The criteria used include:

- What journals do the Greek researchers publish with, and what (other) journals do they reference in the works? Once this question is addressed, its outcome could very well comprise a means for "quantifying" the value a journal represents to the Greek research community. Having done so, it will then become possible to quantify the effect/impact the discontinuation of a subscription will have to HEAL-Link members.
- The number of full-text downloads per scientific journal, conducted by HEAL-Link users over given time periods. This information is provided by the corresponding publishers.
- The journal's interdisciplinary value.

A journal can be rated to be important for HEAL-Link by also considering its scientific impact at the international level. It represents added value for HEAL-Link when Greek researchers reference its articles in their works, and, to a lesser extent, when they publish their works with it. The present paper focuses on

[1] HEAL-Link, https://www.heal-link.gr/.

scientific journals the Greek academic/research community publish their works with, and on the journals they reference in their works. The terms "citation" and "reference" are often used interchangeably. Some researchers define them as two complementary actions: "reference" as "acknowledgement to" and "citation" as "acknowledgment from" [7]. It is assumed that when a researcher makes a reference to a scientific article in his work, the journal the latter is published with deserves the credit for being a valuable source of scientific information. In this respect, the more references made by researchers to articles published with a given journal, the higher is the value the latter as a source of scientific information to the HEAL-Link user community.

2 Related Work

Data visualization is important because it gives the data a specific form of expression so that they are broadly readable and understandable, and it helps to interpret and analyze results. Data visualization is the interpretation of information in visual terms by forming a mental picture based on data and the application of suitable methods to put data into visible form [5]. Additionally, [1] defines data visualization as "the use of computer supported, interactive, visual representations of data to amplify cognition", while [8] point out that "data visualization is for exploration, for uncovering information, as well as for presenting information. It is certainly a goal of data visualization to present any information in the data, but another goal is to display the raw data themselves, revealing their inherent variability and uncertainty."

Most related research work in this field and especially in bibliometrics has been done on science mapping. Science mapping aims to build science/bibliometric maps that describe how specific disciplines, scientific domains, or research fields are conceptually, intellectually, and socially structured [3]. It comprises an important research topic in the field of bibliometrics [4]. Science mapping involves a number of discrete steps: data retrieval, preparation and preprocessing, network extraction, mapping, analysis and visualization [3]. The preprocessing step is the most critical one. Different types of maps have been studied, some revealing relations among authors, documents, journals, or keywords, usually constructed by utilizing citations, co-citations, or bibliography coupling based on co-occurrences of keywords in documents [4]. Many software tools on science mapping have been developed, reviewed, analyzed and evaluated [3]. The ones most widely used in the literature can be found in [2] and [3].

Visualization tools have always been key elements not only in science and research, but also in any industry relating to the production of data that can be visualized [6]. The aim is to assist anyone who is involved in the production of knowledge, or in the decision making process. Visualization reveals information that cannot be found as such in raw data. The ability to see data clearly creates a capacity for building intuition that is unsurpassed by summary statistics [6].

The two most popular subscription-based bibliographic databases are (a) the Web of Science[2], an online subscription-based scientific citation indexing service maintained by Clarivate Analytics, and (b) Scopus[3], Elsevier's abstracts and citations database. Both vendors have built online applications for data visualization on various aspects of the scientific environment, and on their publications.

InCites from Clarivate Analytics[4] is a citation-based evaluation tool for academic and government administrators to analyze institutional productivity and benchmark output against peers and aspirational peers in the national or international context. It provides the means to gather and analyze data with multiple visualization types that communicate effectively the collected information, helping the subscribers make informed decisions, and assisting strategic initiatives. In higher education, an institution may specify various criteria in order to determine its peer and aspirational peer institutions. Peer institutions are other institutions that are at a similar institutional level and they have similar institutional characteristics. Aspirational peer institutions are institutions with similar institutional characteristics, yet they have better key performance indicators, such as significantly higher graduation rates and higher level performance. They are a want to be state of the institutions that set them as asparational. Peer and aspirational peer institutions can be defined for the overall institution, but they can also be defined for different schools or departments. Peer and aspirational peer institutions are appropriate for benchmarking purposes.

SCImago Journal & Country Rank from Elsevier[5] is a publicly available portal that includes journal and national scientific indicators developed from the information contained in the Scopus database. These indicators can be used to assess and analyze scientific domains from journals to country rankings. The SCImago Journal & Country Rank network visualization tools maps multiple networks based on each journal selection. The information visualization projects *Shape of Science*, *Subject Bubble Chart* and *World Report* aim to reveal the structure of science. They involve the construction of real time maps and bubble charts offering detailed information for the analysis of the world as whole, plus for each one of eight large geographic regions.

3 Data Collection, Preparation and Unification

In the direction of exploiting the aforementioned research goal, bibliographic metadata from nearly 63,000 research publications have been collected and processed. The collection consisted of research publications involving at least one (co-)author affiliation to a Greek HEAL-Link member academic/research institution. The Scopus database was used to retrieve research publications spanning a period of nine years (2010–2018) and involving nearly 10,000 journals. The PostgreSQL RDBMS was used to organize the bibliographic data collection.

[2] https://clarivate.com/products/web-of-science.
[3] https://www.scopus.com.
[4] https://clarivate.com/products/incites.
[5] https://www.scimagojr.com/.

A number of models exist that classify scientific journals (Library of Congress classification, Web of Science, Scopus ASJC schema, etc.). We adopted the Scopus ASJC (All Science Journal Classification System) schema[6]. Our decision was based on the fact that the ASJC schema involves a small and manageable number of subject areas facilitating the handling of important cognitive areas/categories. ASJC is a formal model used by many researchers. It involves 3 levels of classification. We have decided to use the second level which comprises of 27 subject categories. Bibliographic data are coupled with metadata on subject (discipline) area(s). This information will be used to identify journals of interdisciplinary value during the next stages of our research.

To harvest the publications made during the 2010–2018 period by Greek researchers as well as the references they make to other journals, two API services were used: the Crossref REST API[7], and the Scopus APIs[8].

In order to make the dataset consistent and to further improve the accuracy of our research results, a set of data preparation/cleansing tasks were conducted. During the data harvesting stage, the Scopus APIs service was used to harvest all publications involving at least one (co-)author who is affiliated to a Greek academic/research institution.

Table 1 presents the number of publications retrieved, grouped by publication type.

Table 1. Information about the publications

Publication type	Count
Article	47035
Article in Press	1320
Book	341
Chapter	3437
Conference Paper	15271
Editorial	1644
Erratum	242
Letter	813
Note	467
Review	4094
Short Survey	96
Total	74760

The book and chapter type publications, plus any other publication not assigned ISSN number were removed from the dataset. ISSN is a unique 8-

[6] https://service.elsevier.com/app/answers/detail/a_id/15181/supporthub/scopus/.
[7] https://api.crossref.org.
[8] https://dev.elsevier.com/sc_apis.html.

digit identifier assigned to periodical publications, irrespective of their medium (print or electronic)[9]. For the publications data used, ISSN primarily identifies scientific journals.

The publications harvesting stage was followed by the harvesting of metadata relating to the references made by the published works. The references harvesting process was split into two sub-processes. One for works published with Elsevier journals, and one for all other publications. This became necessary due to Elsevier's not making available to the Crossref REST API the references data of their published works. A separate data harvesting stage was conducted using the Scopus API in order to harvest the Elsevier journals references data. Most of the publications data records retrieved via the Crossref REST API include the corresponding references lists. The latter are of a diverse format, one that lacks a specific pattern even for works published with the same publisher. As an example, the digital object identifier (DOI)[10], which is the most important field as it is explained below, is either not present in the Crossref record, or it exists in a field of its own, or it is incorporated alongside with other information in other fields.

Today, nearly all published works are assigned a unique DOI. The latter is a unique alphanumeric string assigned by a registration agency (the International DOI Foundation[11]). It is used to identify content and provide a persistent link to its location on the Internet. As such, the DOI identifier facilitates the retrieval of the corresponding scientific journal metadata. In this respect, for the Crossref records that would not conform to a pre-specified pattern, the DOI identifiers were used to track down and retrieve the corresponding (referenced) journal metadata. For the Crossref records that lacked a DOI identifier, a separate data harvesting stage was conducted in order to retrieve the ISSN number of the journal each (referenced) work was published with.

The Scopus API is more coherent than the Crossref REST API. Almost all of Scopus API retrieved works include the corresponding references list, including unique identifiers for each one member of the latter (DOI, or Scopus identifier[12]). Unfortunately, it imposes quotas that restrict API-based data retrieval operations for an extended number of records. This is the reason why the Scopus APIs were used for all the references that could not be harvested from the Crossref REST API. An effort was made to use the Scopus APIs for as small of a number of records as possible and by doing so remain within the Scopus imposed quotas.

The next stage involved the harvesting of journal ISSNs, for journals the referenced works are published with. For references with DOIs, the ISSN identifiers were harvested from the Crossref REST API. For references lacking their DOI identifiers, data harvesting was conducted by using the Scopus APIs, plus the Arxiv electronic preprints repository[13]. Additional checking was conducted to

[9] https://www.issn.org/understanding-the-issn/what-is-an-issn/.
[10] https://en.wikipedia.org/wiki/Digital_object_identifier.
[11] https://www.doi.org.
[12] https://www.elsevier.com/solutions/scopus/how-scopus-works/high-quality-data.
[13] https://arxiv.org.

ensure that each sought for bibliographic record existed in the harvested dataset. As a follow up, additional techniques were applied during the ISSN harvesting stage. They included similarity checking on other fields, e.g. the journal title. When the number of records to be checked was reduced to a small value, even manual checking was conducted in order to identify the target journal.

4 The J2J-GR Service

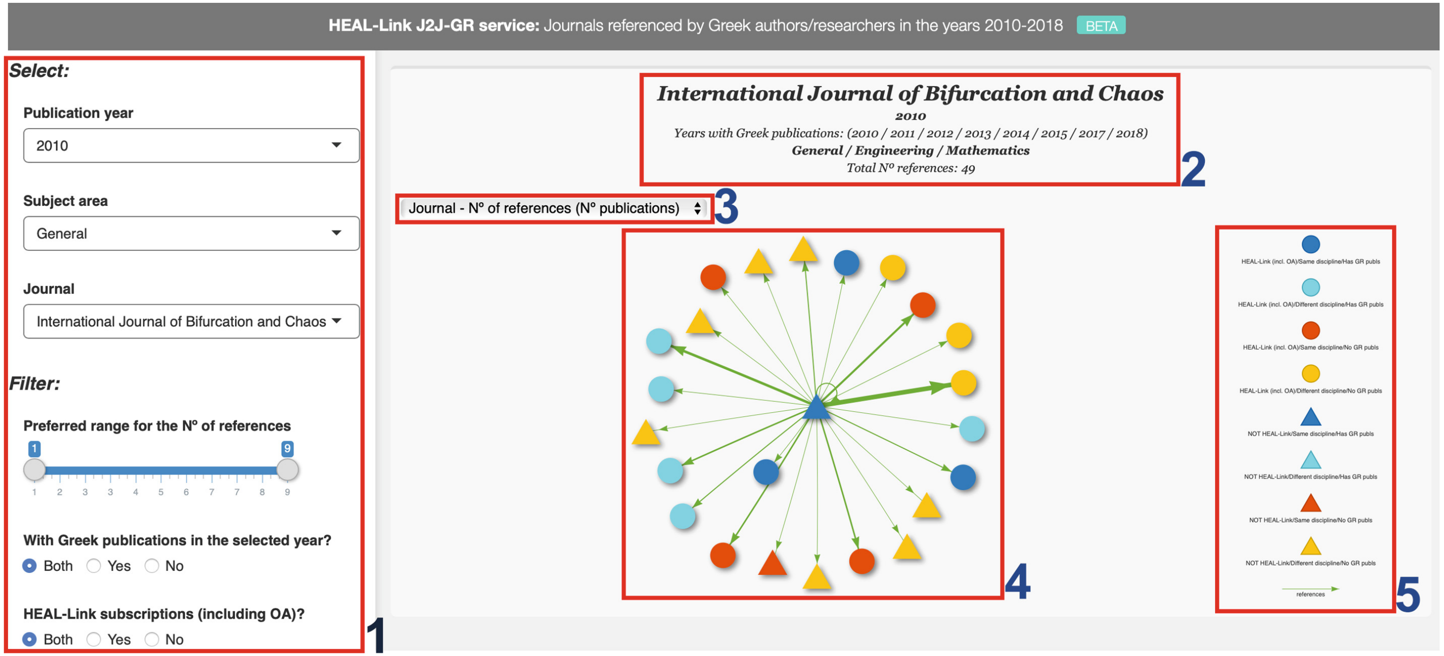
In addition to conducting the relevant data (pre-)processing and unification tasks, a Web application has been developed and implemented utilizing interactive graphs to facilitate the interpretation of the findings[14]. Aiming for the development of a powerful and informative Journal-to-Journal (J2J) associations service, the current version comprises one first step. It involves a dynamically generated set of graph networks from the journal to journal references data, focusing on references originating from publications involving at least one (co-)author who is affiliated to a Greek HEAL-Link member institution (target authors population). In this respect, the current version of the service targets a specific population of authors by considering scientific publications the latter have made during the 2010–2018 years period. The interactive graphs generated present the references made by the published works of the targeted authors population.

The application has been coded in R[15], utilizing the RStudio[16] project development environment. To construct the interactive Web apps straight from R, the Shiny[17] R package and shinydashboard[18] have been adopted.

As it is shown in Fig. 1, the user interface involves two sections. The sidebar on the left, where the publication year, the subject area and the journal title are set together with some relevant fine tuning parameters, and the main section where the generated graph is presented.

The sidebar, marked as (1) in Fig. 1, includes three selection menus: one for the publication year, one for the subject area, and one for the journal title of interest. In addition, the sidebar includes three filters for narrowing down the scope of the journals referenced by works published with the journal of interest. Filter number one is used to specify a range for the number of references made to other journal publications by the works of the target authors group published with the set journal during the set publication year. Filter number two enables the user to specify whether or not the list of referenced journals includes publications of the target authors group during the set publication year. Filter number three enables the user to include or exclude HEAL-Link subscriptions/journals.

[14] http://j2j.heal-link.gr/.
[15] https://www.r-project.org.
[16] https://www.rstudio.com.
[17] https://shiny.rstudio.com.
[18] https://rstudio.github.io/shinydashboard.



HEAL-Link J2J-GR service: Journals referenced by Greek authors/researchers in the years 2010-2018 BETA

Select:

Publication year: 2010

Subject area: General

Journal: International Journal of Bifurcation and Chaos

Filter:

Preferred range for the Nº of references: 1 – 9

With Greek publications in the selected year? [✓ Both] Yes No

HEAL-Link subscriptions (including OA)? [✓ Both] Yes No

International Journal of Bifurcation and Chaos

2010

Years with Greek publications: (2010 / 2011 / 2012 / 2013 / 2014 / 2015 / 2017 / 2018)

General / Engineering / Mathematics

Total Nº references: 49

Journal - Nº of references (Nº publications)

1
2
3
4
5
HEAL-Link (incl. OA)/Same discipline/Has GR publs
HEAL-Link (incl. OA)/Different discipline/Has GR publs
HEAL-Link (incl. OA)/Same discipline/No GR publs
HEAL-Link (incl. OA)/Different discipline/No GR publs
NOT HEAL-Link/Same discipline/Has GR publs
NOT HEAL-Link/Different discipline/Has GR publs
NOT HEAL-Link/Same discipline/No GR publs
NOT HEAL-Link/Different discipline/No GR publs
references

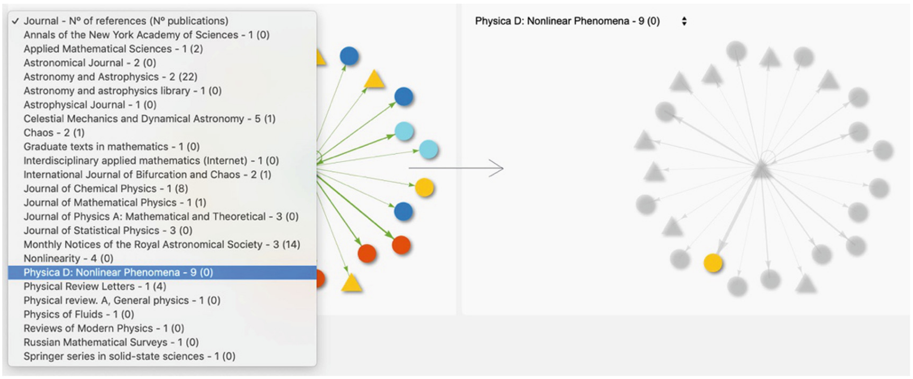
Fig. 1. The J2J-GR Web application user interface screen (Color figure online)

The main page displays the graph generated in accordance with the settings made in the sidebar. The graph consists of four parts: the journal's title and information marked as (2), a drop down list of referenced journals marked as (3), the graph itself marked as (4), and its informative legend marked as (5) in Fig. 1.

The journal selected in the sidebar becomes the central node in the graph. Its title section lists the year of interest, the years that this journal includes publications made by Greek researchers, the subject area(s) covered, and the total number references made to other journals from works (co-)authored by Greek researchers during the selected year. The graph is re-generated each time a (new) filter is applied on the sidebar.

The thickness of the graph's edges in Fig. 1 encodes the number of references made to the corresponding journals. By hovering the mouse over any (target) journal the user can see information, such as its title, the number of works (co-)authored by Greek researchers published with it in the year of interest (if any) as well as the number of such publications during the 2010–2018 period, plus the number of references the central node (reference) journal makes to the journal in question.

The content of the dropdown list marked as number 2 in Fig. 1 is displayed in Fig. 2. It involves the same information displayed in the interactive graph, this time presented in textual form. When the user selects a (target) list element/journal, the corresponding graph node gets highlighted for the user to click on and have it become the new central node, if so desired. In this respect, the interactive graph supports a fan-out type of functionality. This is useful especially in cases of the user having to navigate himself in a large numbers of referenced (target) journals.


✓ Journal - N° of references (N° publications)
Annals of the New York Academy of Sciences - 1 (0)
Applied Mathematical Sciences - 1 (2)
Astronomical Journal - 2 (0)
Astronomy and Astrophysics - 2 (22)
Astronomy and astrophysics library - 1 (0)
Astrophysical Journal - 1 (0)
Celestial Mechanics and Dynamical Astronomy - 5 (1)
Chaos - 2 (1)
Graduate texts in mathematics - 1 (0)
Interdisciplinary applied mathematics (Internet) - 1 (0)
International Journal of Bifurcation and Chaos - 2 (1)
Journal of Chemical Physics - 1 (8)
Journal of Mathematical Physics - 1 (1)
Journal of Physics A: Mathematical and Theoretical - 3 (0)
Journal of Statistical Physics - 3 (0)
Monthly Notices of the Royal Astronomical Society - 3 (14)
Nonlinearity - 4 (0)
Physica D: Nonlinear Phenomena - 9 (0)
Physical Review Letters - 1 (4)
Physical review. A, General physics - 1 (0)
Physics of Fluids - 1 (0)
Reviews of Modern Physics - 1 (0)
Russian Mathematical Surveys - 1 (0)
Springer series in solid-state sciences - 1 (0)

Physica D: Nonlinear Phenomena - 9 (0)

Fig. 2. The dropdown list. (Color figure online)

The graph's legend marked as element number 5 on the screen shown in Fig. 1 encodes characteristics of the graph's nodes/journals. Journal characteristics are encoded by combining geometrical shapes and colors. Geometrical shapes depict the inclusion or not of the journal in the HEAL-Link subscriptions list: a circle indicates a journal that HEAL-Link members have access to. This includes HEAL-Link paid subscriptions, and HEAL-Link listed Open Access (OA) journals. On the other hand, a triangle indicates a journal outside the HEAL-Link paid subscriptions and OA list. Colors encode additional journal characteristics as follows: (a) blue indicates a journal that shares a common discipline area with the central node in the graph the Greek researchers have published works with in the year considered/selected, (b) turquoise indicates a journal that shares no common discipline area with the central node, yet one where Greek researchers have published works with in the year considered/selected, (c) red indicates a journal that shares a common discipline area with the central node with no works published by Greek researchers in the year considered/selected, and (d) yellow indicates a journal that shares no common discipline area with the central node with no works published by Greek researchers in the year considered/selected.

All journal nodes in the graph are clickable. When a user clicks on a node, a new graph is generated with the clicked journal as the (new) selected central node/journal. When this happens, the year selected remains the same, and the (new selected) journal title appears in the areas marked (1) and (2) in Fig. 1. If the new central node shares the same subject (discipline) area considered so far then the name of the latter remains selected in the area marked (1) in Fig. 1, otherwise a new subject area name (of the new central node) is selected automatically. Lastly, the three filter settings in the area marked (1) in Fig. 1 they are all reset. Figures 3 and 4 demonstrate the stated fan-out type of functionality of the interactive graph shown in Fig. 1.

5 Future Work

Our next goal is to conduct exploratory analysis and data mining operations on the HEAL-Link bibliographic dataset. It will be interesting, for example, to consider the journals Greek researchers reference to next to the references made by of all the authors who publish in the same journals with the former. The exploratory analysis outcome could then be used to proceed and conduct predictive analytics in order to identify new potential additions to the HEAL-Link journals subscriptions list. Equally well, the same type of processing could facilitate the HEAL-Link decision making process for cancelling subscriptions dictated by budget cutbacks. Last but not least, the aforementioned analytical processing is expected to reveal journals of an interdisciplinary value to the HEAL-Link user community, i.e. journals used as a scientific information source material by researchers representing diverse scientific discipline areas. Such journals usually represent a high reference value to the HEAL-Link user community.

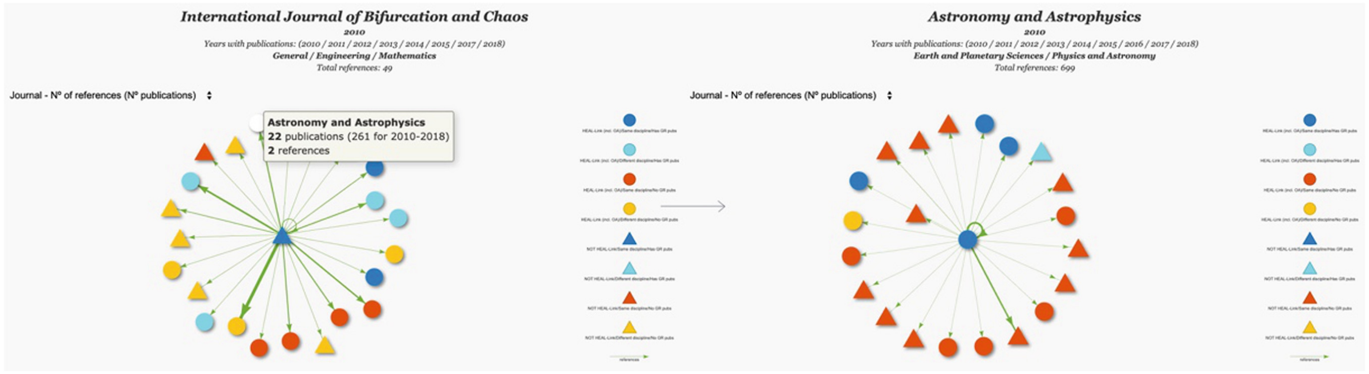

International Journal of Bifurcation and Chaos
2010
Years with publications: (2010 / 2011 / 2012 / 2013 / 2014 / 2015 / 2017 / 2018)
General / Engineering / Mathematics
Total references: 49
Journal - Nº of references (Nº publications)
Astronomy and Astrophysics
22 publications (261 for 2010-2018)
2 references
Astronomy and Astrophysics
2010
Years with publications: (2010 / 2011 / 2012 / 2013 / 2014 / 2015 / 2016 / 2017 / 2018)
Earth and Planetary Sciences / Physics and Astronomy
Total references: 699
Journal - Nº of references (Nº publications)
references

Fig. 3. Graph generation after clicking on a referenced journal Greek researchers have published with during the selected year. (Color figure online)

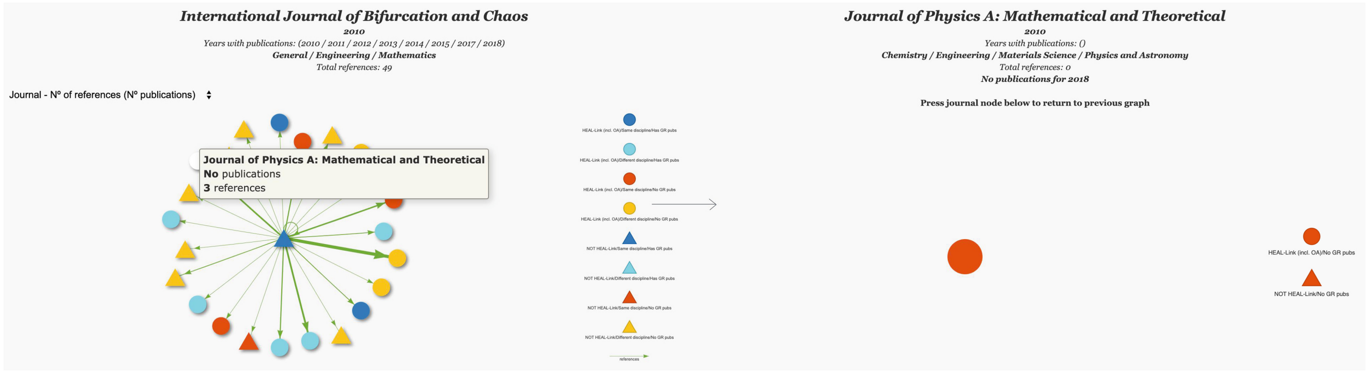

International Journal of Bifurcation and Chaos
2010
Years with publications: (2010 / 2011 / 2012 / 2013 / 2014 / 2015 / 2017 / 2018)
General / Engineering / Mathematics
Total references: 49
Journal - Nº of references (Nº publications)
Journal of Physics A: Mathematical and Theoretical
No publications
3 references
HEAL-Link (incl. OA)/Same discipline/Has GR pubs
HEAL-Link (incl. OA)/Different discipline/Has GR pubs
HEAL-Link (incl. OA)/Same discipline/No GR pubs
HEAL-Link (incl. OA)/Different discipline/No GR pubs
NOT HEAL-Link/Same discipline/Has GR pubs
NOT HEAL-Link/Different discipline/Has GR pubs
NOT HEAL-Link/Same discipline/No GR pubs
NOT HEAL-Link/Different discipline/No GR pubs
references
Journal of Physics A: Mathematical and Theoretical
2010
Years with publications: ()
Chemistry / Engineering / Materials Science / Physics and Astronomy
Total references: 0
No publications for 2018
Press journal node below to return to previous graph
HEAL-Link (incl. OA)/No GR pubs
NOT HEAL-Link/No GR pubs

Fig. 4. Graph generation after clicking on a referenced journal Greek researchers have not published with during the selected year. (Color figure online)

6 Conclusion

We report on the current of a research undertaken in order to exploit journal-to-journal references by considering publications made by a specific target authors group, namely researchers affiliated with academic institutions and research centers in Greece. The usage value each scientific journal represents is quantified by considering the references made to it by published works involving at least one (co-)author affiliation to a Greek academic and/or research institution. It should be borne in mind that a specific journal may represent a different usage value when considered in relation with the references received by works published by Greek researchers, as compared to when the same journal is considered in relation with the references received by all works published in journals the Greek researchers publish with. This possibility is worth exploiting further by conducting exploratory and predictive analytics type of processing to bibliographic datasets the Hellenic Academic Libraries Link (HEAL-Link) Consortium has access to.

The work reported herewith comprises a first step towards the research goal outlined in the previous paragraph. Having collected and pre-processed a first collection of releveant bibliographic data, an attempt has been made to realize the inherent journal-to-journal associations with respect to the references they make to one another. A first (beta) version of the J2J-GR Web application has been implemented and it is publicly available on the Internet. The focus is on works published in the 2010–2018 years period, and the references made by Greek researchers in works they have published with research journals during the stated period. It is anticipated that the presentation of the relevant information in the form of interactive graphs as in[19] will evolve into a bibliographic data exploratory tool useful to both the HEAL-Link Consortium and its user community.

References

1. Arnheim, R.: Visual Thinking. University of California Press, Berkeley (1969)
2. Borner, K., et al.: Rete-netzwerk-red: analyzing and visualizing scholarly networks using the network workbench tool. Scientometrics **83**, 863–876 (2010). https://doi.org/10.1007/s11192-009-0149-0
3. Cobo, M.J., Lopez-Herrera, A.G., Herrera-Viedma, E., Herrera, F.: Science mapping software tools: review, analysis, and cooperative study among tools. J. Am. Soc. Inf. Sci. Technol. **62**(7), 1382–1402 (2011). https://doi.org/10.1002/asi.21525
4. van Eck, N.J., Waltman, L., Dekker, R., van den Berg, J.: A comparison of two techniques for bibliometric mapping: multidimensional scaling and VOS. J. Am. Soc. Inf. Sci. Technol. **61**(12), 2405–2416 (2010). https://doi.org/10.1002/asi.21421
5. Hinterberger, H.: Data visualization. In: Liu, L., ÖzsU, M.T. (eds.) Encyclopedia of Database Systems, pp. 652–657. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-39940-9
6. Moody, J., McFarland, D., BenderdeMoll, S.: Dynamic network visualization. Am. J. Sociol. **110**(4), 1206–1241 (2005). https://doi.org/10.1086/421509

[19] http://j2j.heal-link.gr/.

7. Narin, F., Carpenter, M.P.: National publication and citation comparisons. J. Am. Soc. Inf. Sci. **26**(2), 80–93 (1975). https://doi.org/10.1002/asi.4630260203
8. Unwin, A., Theus, M.: Graphics of a Large Dataset, pp. 227–249. Springer, New York (2006). https://doi.org/10.1007/0-387-37977-0_11

Deep Learning for French Legal Data Categorization

Eya Hammami[1](✉), Imen Akermi[2], Rim Faiz[1], and Mohand Boughanem[2]

[1] LARODEC Laboratory, University of Manouba, Manouba, Tunisia
eyahammami9@gmail.com
[2] IRIT Laboratory, University of Toulouse 3, Toulouse, France
imen.akermi@irit.fr

Abstract. In current years, deep learning has showed promising results when used in the field of natural language processing (NLP). Neural Networks (NNs) such as convolutional neural network (CNN) and recurrent neural network (RNN) have been utilized for different NLP tasks like information retrieval, sentiment analysis and document classification. In this paper, we explore the use of NNs-based method for legal text classification. In our case, the results show that NN models with a fixed input length outperforms baseline methods.

Keywords: Natural Language Processing · Deep learning · Convolutional Neural Networks · Document categorization · Legal domain

1 Introduction

An automatic text classification system is a significant task. Indeed, there are several applications that require the partition of natural language data into groups, e.g. classifying opinions retrieved from social media sites, or filtering spam emails, etc. In this work, we assert that law professionals would considerably gain advantage from the type of automation supplied by machine learning. This is especially the case of legal research, where the preparation of a legal practitioner has to be assumed before defending a case, law professionals have to take complicated decision regarding several aspects of a given case. Given the data accessible on court decisions and machine learning techniques, it is possible to train text categorization systems to predict some of these decisions. Such system can act as a decision support system for law professionals. Many popular approaches have been utilized in text classification like, Naive Bayes classifier, Support Vector Machine, Logistic Regression... and most recently deep learning methods such as Convolutional Neural Network (CNN) [7], Recurrent Neural Network and Long-Short Term Memory (LSTM) [24,25]. Most of these approaches are not particularly designed for the legal domain and are usually trained with English text, which make them not appropriate to be used for French text and particularly legal French text. Indeed, French is a language

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 96–105, 2019.
https://doi.org/10.1007/978-3-030-32065-2_7

with a richer morphology and a more flexible word order, for this we need more pre-processing to achieve good accuracy results and capture the hidden semantics specially when dealing with legal texts.

In this paper, we propose NN-based model with dynamic input length layer to process French legal data. We also present a comparative study between the proposed approach and several baseline models.

This paper is organized as follows: we present in Sect. 2 a literature review that examines the different approaches for text classification. In Sect. 3, we describe our proposed model. Experiments are presented in Sect. 4.

2 Related Work

Text classification is a necessary task in Natural Language Processing. Traditionally, linear classifiers are frequently used for text classification [1,2]. Joulin et al. [3] indicate that linear models could scale up to a very huge dataset rapidly with a proper rank constraint and a fast loss approximation. Recently deep learning methods, such as recurrent neural networks: [5,6] and, Long short Term Memory (LSTM) have been used in language modeling. Those methods are adapted to natural language processing because of their ability to extract features from sequential data. Convolutional Neural Network (CNN) [7–9], usually used for computer vision tasks, has been adopted in NLP for the first time in [4]. The authors presented a new global max-pooling operation, which is revealed to be efficient for text, as an alternative to the conventional local max pooling of the original LeNet architecture [10]. Furthermore, they suggested to transfer task-specific information by co-training different deep models on many tasks. Inspired by the original work of kim [7], Ronan et al. [11] introduced a simpler architecture with modifications consisting of fixed pretraining word2vec embeddings [12]. The author demonstrates that this model can already achieve state-of-the-art performances on many small datasets. Dynamic Convolutional Neural Network (DCNN) is a type of CNN which is introduced by [13]. Their approach outperforms other methods on sentiment classification. They use a new pooling layer called a dynamic K-max pooling. This dynamic k-max pooling is a generalization of the max pooling operator, which computes a new adapted K value for each iteration. Thus, their network can read any length of an input. Zhang et al. [14] introduced a character-level Convolutional Neural Network (Char-CNN); Fig. 1 gives an illustration of their approach. Their model yields a better result than other methods including a word-level CNN on sentiment analysis and text classification. A one-hot encoding has been utilized as an input for this network.

Koomsubha et Vateekul [15] proposed a new Char-CNN model inspired by the work presented in [14] with a capability to accept any length of input by employing k-max pooling before a fully connected layer to categorize Thai news from a newspaper. Kim et al. [16] presented a character aware neural language model by combining a CNN on character embeddings with a highway LSTM on subsequent layers. In addition, [17] analyzed a multiplicative LSTM (mLSTM) on character embeddings and found that a basic logistic regression learned on

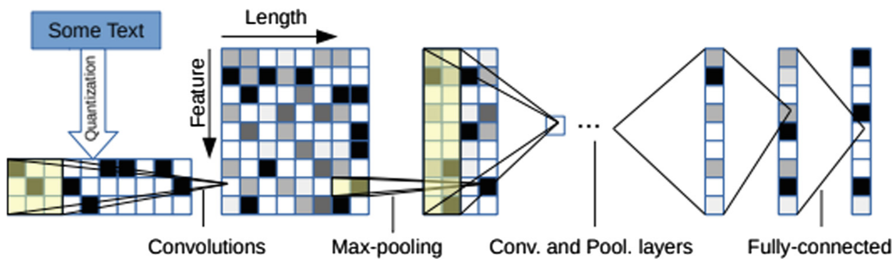


Fig. 1. Char-CNN introduced by Zhang et al. [14].

this representation can reach state-of-the art results on the Sentiment Tree Bank dataset [18] with a few hundred labeled examples. We have found a rather small body of previous works about automatic text classification of legal documents. For example, support vector machines (SVMs) have been used to classify legal documents like legal docket entries [19]. Sulea et al. [20] developed a mean probability ensemble system combining the output of multiple SVM classifiers to classify French legal texts. Wei et al. [23] note preliminary studies in utilizing deep learning for text classification in legal documents. They organized their researches to compare deep learning results with results obtained using SVM algorithm on four datasets of real legal documents. Results demonstrated that CNN present better accuracy score with training dataset of larger size and can be improved for the text classification in legal industry. Furthermore Neural Networks such as CNN, LSTM and RNN have been used for classifying English legal court opinions of Washington University School of Law Supreme Court Database (SCDB) [21]. The authors compared a few of machine learning algorithms with the late Neural Networks systems and they found that a CNN network with Word2vec vector performed better compared to the other and gave an accuracy around 72.7%. Silva et al. [22] applied CNN on Brazilian court's document and they achieved good results. However, all of these works are generally based on CNN model for text classification in legal domain and usually use static input length. Therefore, we propose to experiment this model with dynamic input length on French legal data. Experiments on real datasets highlight the relevance of our proposal and open up many perspectives.

3 Proposed Model

Our suggested CNN is based on Undavia's model [21]. In this model there is a max pooling layer called temporal max pooling. It carries out an operation on 1-D. It is calculated by the following formula [15]:

$$P_{r,c} = max_{j=1}^{s} M_{r,s(c-1)+j}$$

where:

- M is an input matrix with a dimension of $n \times l$
- s is a pooling size
- P is an output matrix with a dimension of $n \times \frac{l}{s}$
- c is a column cell of matrix P
- r is a row cell of matrix P

Our contribution regards the pooling layer. We use k-max pooling layer rather than max-pooling layer. The k-max pooling operation enables to pool the k maximum active features in p. It keeps the order of the features, but it is insensitive to their accurate positions. It can also detect more delicately the number of times where the feature is very activated in p. The k-max pooling operator is used in the network after the highest convolutional layer. This allows the input to the fully connected layers to be separate from the length of the input sentence. In the middle of convolutional layers, the pooling parameter k is not fixed, but is selected in a dynamic way to enable a sleek extraction of longer-range and higher order features [13]. This pooling layer is calculated by the following formula [15]:

$$P_{r,*} = kmax_{j=1}^{l} M_{r,j}$$

where:

- M is an input matrix with a dimension of $n \times l$
- K is an integer value
- P is an output matrix with a dimension of $n \times k$
- $*$ shows that all columns in a row are calculated together
- r is a row cell of matrix P

The main differences between these two types of pooling layer consists in the use of a gliding window. Max pooling is a method for down sampling data by utilizing a gliding window on a row of data and choosing a cell which includes a maximum value to be moved to the next layer. Differently, k-max pooling doesn't have a window. But it has a choosing operation carries out all data in a row. Then, top k cells which have maximum value are chosen to be utilized in the upcoming layer [15].

By applying K-max pooling in a convolutional neural network, as we propose, we can definitely have a matrix which is able to fit into a fully connected layer regardless of the length of an input. Figure 2 illustrates our proposed method. On the convolutional and pooling layers, the length of data belongs to the input. Whereas after the k-max pooling layer, the length of data in each document is coequal. Thus, our neural network classification model is a little bit similar to the one introduced by kim [24], but we modified the layers, by adding other layers and modifying some of the original hyperparameters. Our model first makes an embedding layer using word2vec as a pre-trained word embeddings, and next makes a matrix of documents represented by 300 dimensional word embeddings. We incorporate three sets of the following: a dropout of 0.5, a convolution layer of 128 filters with a filter size of 3, and we set k value of k-max pooling to 5. We also add a dense layer consisting of 128 units between two dropouts of 0.5 to prevent overfitting. Finally, the last layer is a dense layer with size of 6 equal to the number of labels (categories) of our corpus.

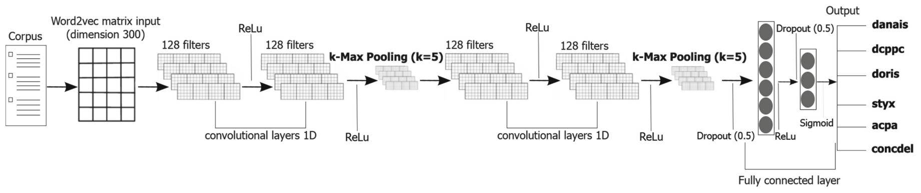


Fig. 2. Proposed-CNN-architecture.

4 Experiments and Results

4.1 Dataset

We train and test our model on French legal dataset collected from data.gouv.fr, it is a documentary collection of case law decision from the French courts of appeal and the courts of the first degree which is composed of a selection of decisions in civil matter and criminal. The dataset includes 452 documents (txt files) organized into 6 legal categories, see Table 1. The number of documents was limited because the project is in progress, and the annotation of documents is done manually and exclusively by legal experts. Work is underway to try to expand the corpus of learning. After the processing, the vocabulary size is 794659. We randomly divides it into training and test set with 80% and 20% split.

Table 1. Legal categories (denomination of the classes was carried out by a legal expert)

Number of documents	Label
198	danais
91	dcppc
59	doris
50	styx
30	concdel
24	acpa

4.2 Pre-processing

Our model first removes special characters like punctuation, stopwords, numbers and whitespaces. Second, we use French Spacy and NLTK modules of Python to recognize the named entity, then we remove it from our corpus, on the assumption that a smarter text categorization technique would be able to interpret the layout more accurately. Third, we also use the TreetaggerWrapper module of

Python for lemmatization (the process of converting a word to its base form). The reason why we choose lemmatization rather than stemming is because lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors. Finally, each word in the corpus is mapped to a word2vec vector (we use pretrained word2vec models) before being fed into the convolutional neural network for categorization.

4.3 Experiments

In this paper, we use accuracy as a measure in order to evaluate the proposed model and the baseline models, which is calculated by:

$$Accuracy = \frac{number\, of\, correctly\, classified\, documents}{total\, number\, of\, classified\, documents}$$

We also compare our results by F_1. Given $F_{1,i}$ is F_1 of class i. Suppose there are C classes. The F_1 using macro average is calculated by:

$$F_1 = \frac{\sum_{i=1}^{C} F_{1,i}}{|C|}$$

Where:

$$F_{1,i} = 2 \cdot \frac{precision_i \cdot recall_i}{precision_i + recall_i}$$

We experimented 3 CNN based architectures: proposed approach (CNN-k-max pooling), (CNN-max pooling) and (CNN-global max pooling). In CNN with max pooling we use the same hyperparameters as CNN with global max pooling, but we change the pooling size to 3. The implementation of this three architecture is done using Keras which allows users to choose whether the models they build are running on Theano or TensorFlow. In our case the models run on TensorFlow

Regularization of Hyperparameters: In our experiments, we tested our model with a set of various hyperparameters. The model performed best when using 128 filters for each of the convolutional layers. For these settings we experienced values of 32, 64, 128 and 256. Fundamentally, the 128 gave better results than the inferior settings. It is likely that 128 is simply the largest setting that is convenient to use given the available dateset. In addition, each of the models are adjusted with a dropout [26], which works by "dropping out" a proportion p of hidden units throughout training. We discovered that a dropout of 0.5 and batch size of 256 worked best for our CNNs models, along with the Adem optimizer [27].

Table 2. Result of different CNNs architectures

Method	Accuracy (%)	F1 (%)
CNN with k-max pooling	80,35	80,20
CNN with max pooling	84,46	84.46
CNN with global max pooling	81,94	82,10

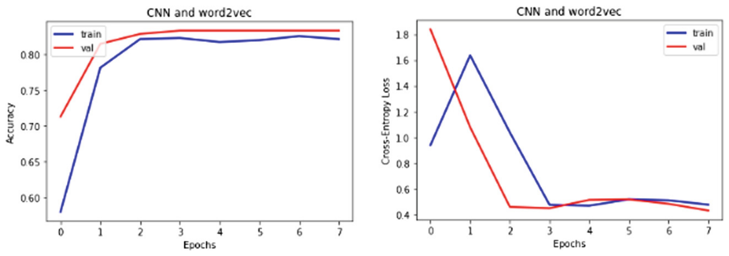


Fig. 3. CNN with max pooling: line plot of Cross Entropy Loss and Classification Accuracy over Training Epochs

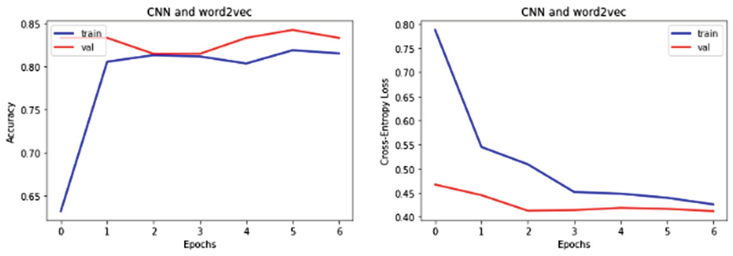


Fig. 4. CNN with k-max pooling: line plot of Cross Entropy Loss and Classification Accuracy over Training Epochs

Results 1: The results are shown in Table 2. The best model in this experiment is CNN with max pooling. It can achieve accuracy of 84,46%, which outperforms our proposed approach, the CNN with k-max pooling and the CNN with global max pooling.

The plots in the Figs. 3 and 4 illustrates that the models seems to have converged. The line plots for both cross-entropy and accuracy both show perfect convergence behavior for the two models, despite somewhat bumpy. The models probably well configured given no sign of over or under fitting.

Results 2: We also compared the CNN based models to two traditional methods Naive Bayes Classifier (with TF-IDF) and Word2vec embedding with Logistic Regression . The results are shown in Table 3. CNN with max pooling also outperforms non NN based models.

Table 3. Result of different CNNs methods

Method	Accuracy (%)	F1 (%)
Naive Bayes class, TF-IDF	41,91	42,00
Word2vec and Logistic Regression	80,88	80,01
CNN with k-max pooling	80,35	80,20
CNN with max pooling	84,46	84,46
CNN with global max pooling	81,94	82,10

5 Discussion

Dynamic max pooling [13,15], usually proved to perform much better compared to classic max pooling and other baseline methods. But surprisingly, in our case, the static (max pooling) outperforms all other methods with an accuracy of 84,46%. We think that this is due to given as input the pre-trained word2vec model. In this paper we considered that the words contained in the pre-trained word embedding may not capture the specificity of languages in our specific legal domain. So maybe for this reason our results may not be optimal due to the generality of the downloaded word embedding model.

6 Conclusion

In this paper, we experienced the use of CNN with dynamic input length to French legal data classification. In our case, our suggested approach, which can accept a longer input, didn't outperform the original model with a fixed input length in terms of accuracy. Therefore, we plan to re-adjust the network architecture, so it can better capture the characteristic of our French legal data.

References

1. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026683
2. McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-1998 Workshop on Learning for Text Categorization, vol. 752, pp. 41–48. Citeseer (1998)
3. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. CoRR, abs/1607.01759 (2016). http://arxiv.org/abs/1607.01759, arXiv:1607.01759. https://dblp.org/rec/bib/journals/corr/JoulinGBM16, dblp computer science bibliography, https://dblp.org. Accessed 13 Aug 2018
4. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)

5. Yogatama, D., Dyer, C., Ling, W., Blunsom, P.: Generative and discriminative text classification with recurrent neural networks. arXiv preprint arXiv:1703.01898 (2017)
6. Xiao, Y., Cho, K.: Efficient Character-Level Document Classification by Combining Convolution and Recurrent Layers, CoRR, abs/1602.00367 (2016). http://arxiv.org/abs/1602.00367, arXiv:1602.00367, https://dblp.org/rec/bib/journals/corr/XiaoC16, dblp computer science bibliography, https://dblp.org. Accessed 13 Aug 2018
7. Kim, Y.: Convolutional Neural Networks for Sentence Classification, CoRR, abs/1408.5882 (2014). http://arxiv.org/abs/1408.5882, arXiv:1408.5882, https://dblp.org/rec/bib/journals/corr/Kim14f, dblp computer science bibliography, https://dblp.org. Accessed 13 Aug 2018
8. Zhang, X., Zhao, J.J., LeCun, Y.: Character-Level Convolutional Networks for Text Classification, CoRR, abs/1509.01626 (2015). http://arxiv.org/abs/1509.01626, arXiv:1509.01626, https://dblp.org/rec/bib/journals/corr/ZhangZL15, dblp computer science bibliography, https://dblp.org. Accessed 13 Aug 2018
9. Conneau, A., Schwenk, H., Barrault, L., LeCun, Y.: Very Deep Convolutional Networks for Natural Language Processing, CoRR, abs/1606.01781 (2016). http://arxiv.org/abs/1606.01781, arXiv:1606.01781, https://dblp.org/rec/bib/journals/corr/ConneauSBL16, dblp computer science bibliography, https://dblp.org. Accessed 13 Aug 2018
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
11. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, Helsinki, Finland, vol. 8, pp. 160–167. ACM, New York (2008). http://doi.acm.org/10.1145/1390156.1390177, https://doi.org/10.1145/1390156.1390177.1390177. ISBN: 978–1-60558-205-4
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. MKalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
14. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, pp. 649–657 (2015)
15. Koomsubha, T., Vateekul, P.: A character-level convolutional neural network with dynamic input length for Thai text categorization. In: 2017 9th International Conference on Knowledge and Smart Technology (KST), pp. 101–105. IEEE (2017)
16. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
17. Radford, A., Jozefowicz, R., Sutskever, I.: Learning to generate reviews and discovering sentiment. arXiv preprint arXiv:1704.01444 (2017)
18. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
19. Nallapati, R., Manning, C.D.: Legal docket-entry classification: Where machine learning stumbles. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 438–446. Association for Computational Linguistics (2008)

20. Sulea, O.-M., Zampieri, M., Malmasi, S., Vela, M., Dinu, L.P., van Genabith, J.: Exploring the use of text classification in the legal domain. arXiv preprint arXiv:1710.09306 (2017)
21. Undavia, S., Meyers, A., Ortega, J.E.: A Comparative study of classifying legal documents with neural networks. In: 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 515–522. IEEE (2018)
22. Da Silva, N.C., et al.: Document type classification for Brazil's supreme court using a convolutional neural network. In: The Tenth International Conference on Forensic Computer Science and Cyber Law-ICoFCS, pp. 7–11 (2018)
23. Wei, F., Qin, H., Ye, S., Zhao, H.: Empirical study of deep learning for text classification in legal document review. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 3317–3320. IEEE (2018)
24. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
25. Wang, X., Liu, Y., Chengjie, S.U.N., Wang, B., Wang, X.: Predicting polarities of tweets by composing word embeddings with long short-term memory. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), vol. 1, pp. 1343–1353 (2015)
26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014). JMLR.org
27. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Metadata Discovery Using Data Sampling and Exploratory Data Analysis

Hiba Khalid[1,2](✉), Robert Wrembel[2], and Esteban Zimányi[1]

[1] Université Libre de Bruxelles, Brussels, Belgium
{hiba.khalid,esteban.zimanyi}@ulb.ac.be
[2] Poznan University of Technology, Poznań, Poland
robert.wrembel@cs.put.poznan.pl

Abstract. Metadata discovery is a prominent contributor towards understanding the semantics of data, relationships between data, and fundamental data features for the purpose of data management, query processing, and data integration. Metadata discovery is constantly evolving with the help of data profiling and manual annotators, resulting in various good quality data profiling techniques and tools. Even though, there are different metadata standards specified for distinct fields such as finance, biology, experimental physics, medicine, there is no generic method that discovers metadata automatically or presents them in a unified way. In this paper, we present a technique for discovering and generating metadata for data sources that do not provide explicit metadata. To this end, we apply exploratory data analysis to produce two kinds of metadata, i.e., administrative and technical, in order to find similarities between resources, w.r.t. their structures and contents. Our technique was evaluated experimentally. The results show that the technique allows to identify similar data sources and compute their similarity measures.

Keywords: Data profiling · Metadata management · Discovery · Enrichment

1 Introduction

Data Integration is an inevitable technique for data analysis and data preparation processes. For standard (relational) data integration virtual [5,12,33], and physical (i.e., data warehouse) architectures [20] have been proposed. However with advent of big data, the data integration complexity has increased due to data heterogeneity and data complexity [6,29]. Some big data integration architectures have already been developed [9]. The most viable and industry accepted architecture for physical integration of big data is a data lake (DL) [27,30]. Data lakes have their own challenges (volume, heterogeneity, data quality, data duplication). For data integration, the query engine needs to interpret the data context [9,24] as well as standards and formats that are comprehensible for a

DL query engine [9,24] with appropriate visualization for user/client [7]. This is mostly referred to as metadata. Metadata types (administrative, descriptive, structural) are crucial in comprehending the information structure and alignment. Metadata offer a number of advantages such as data organization, consistency, efficiency analysis [16,22,32], and provenance [34,35]. The availability of metadata is however a very tedious task, the industry accepted methodology to discover metadata is called data profiling [2,11,14].

Data integration for distributed and heterogeneous datasets can be further facilitated by metadata catalogs similar to relational database concept called Common Warehouse Metamodel [26]. Unfortunately, in the context of big data ecosystems, such an industry accepted and deployed standard for metadata is still under development, i.e., only recently, IEEE has started a standardization project on metadata catalog for big data [18]. In most cases metadata has to be manually rendered, this can be replaced by automatic learning algorithms that can learn and understand by metadata samples or use cases. In such cases, metadata must be inferred (discovered) by applying typically statistical analysis or machine learning algorithms. Collecting, defining, and discovering complete, clean, consistent, and easy to use metadata is still one of challenges in data integration and management, cf. Sect. 2. This challenge requires much more sophisticated solutions in the context of big data integration architectures, like data lakes.

In this paper we provide a first step towards semi-automatic metadata discovery for the purpose of data integration, cf. Sect. 3. Our goal is to be able to answer the following questions: (1) Is there any relevance between domains of the considered datasets? (2) Are any of the datasets duplicate of each other? (3) Does any of the datasets fall under an integration case? (4) What is the common topic of datasets analyzed? In our approach, we propose to use exploratory data analysis (EDA) [3] in conjunction with simplistic data profiling. Our approach was tested on road safety dataset retrieved from UK Gov and Kaggle. The experiments show that the proposed method functions adequately for metadata discovery as it can find relevance between different metadata features based on similarity measures, cf. 4. We conclude that with the help of selective labelling of data samples, more enriched metadata can be attained as a part of the EDA process. We also conclude that with the help of EDA and data profiling we were able to discover descriptive and technical metadata.

2 Related Work

Despite the growing interest in metadata, the existing related work and technologies do not provide amble and generic solution to automatic metadata discovery and automatic metadata assisted data integration. The most practical and eminent technique of discovering metadata in practice is data profiling [1]. Standard data profiling [2] techniques allow to discover numerous data characteristics, e.g., ranges of values, histograms, missing and wrong values, outliers, unique values, integrity constraints, functional dependencies, and inclusion dependencies [2,4,11,25]. [25] proposes to apply an inverted index for finding unary inclusion dependencies (UIND) by means of data mining, this improves upon methods from strict data profiling techniques. Similarly, [4] proposes a comparison between SQL (join variant) queries and two algorithms for inclusion dependencies (IND) discovery. The algorithms outperform the traditional SQL statements at the cost of high comparison of dependent values. This cost can be reduced with the use of heuristic algorithms (e.g., random sampling) that can scale down the IND comparisons and by defining properties over inclusion dependencies. [11] proposes an algorithm, called MUDS, that is capable of discovering three types of metadata, namely: inclusion dependencies (IND), multi-column combinations (UCCs), and functional dependencies (FDs), simultaneously using depth-first search approach. The MUDS algorithm starts with SPIDER [4] for INDs, and after this, the DUCC algorithm is applied for discovering UCCs. MUDS does not discover any other type of metadata besides dependencies. [15] propose to discover metadata with the principle of uncertainty in metadata discovery, identify foreign and primary keys as a post-hoc metadata discovery technique. Once ingested, generated, or discovered, metadata must be stored in a repository. To this end, some standard metadata representations were developed. The most frequently referenced ones include: IEEE LOM [17](used for collections definitions, items and learning objects), SCORM [28] is useful for tracking distributed learning systems incorporated with learning objects, Dublin Core [8] provides development of machine readable schema(XML, RDF) for formal structure and syntax of metadata, and Learning Objects (LO) [19] defines metadata structure using conceptual schema for any type of learning object. [10] proposes two approaches(document content analysis, document context analysis) for automated metadata generation (AMG), to classify and retrieve text documents based on rule sets that can understand a data source to infer a schema design from stored and available metadata. This approach is specific to textual metadata inside documents. After analyzing the state of the art, we conclude that the need to develop automated metadata generation techniques for general purpose metadata discovery is still to be developed.

3 Selective Ingestion Technique

It is imperative to distinguish between metadata categories and data features (e.g., attribute names, patterns inside data, foreign keys, primary keys) that are necessary for data integration and analysis. There are some categories of metadata that tend to contribute more towards data integration as compared to others. The decision about the category of metadata required should depend upon the nature of the task. For example, if a simple summary of datasets is required then descriptive metadata is suitable. However, if a feature analysis is required then technical and administrative metadata are more suitable. The challenge, however, resides in the availability of metadata. The problem that we are addressing with this research is to find smart and effective ways to discover metadata without having to annotate or administer metadata manually. Unfortunately, in practice a substantial amount of metadata is still created manually.

We respond to this challenge by the use of exploratory data analysis for metadata discovery. Figure 1 overviews our approach. In the first step, metadata availability in a data source is assessed. A positive assessment results in executing the lower flow, which is: (1) metadata (MD) pre-processing, whose purpose is to ensure that there are no misspelled or incorrect metadata, it validates headers, labels, or any other kind of metadata, (2) MD annotation with context words to provide meaning for feature names, labels, or sections in a metadata file. A negative assessment results in executing an upper flow, whose task is to discover missing metadata. To this end, the following tasks are executed: (1) data sampling to ensure that minimum data is accessed for understanding and discovering metadata, (2) feature extraction to further minimize the actual data access, we focus on extracting features from the dataset for creating reference lists (attribute, keyword) that can be compared with textual descriptions, query text, and query lists, and (3) data analysis and profiling to provide statistical insights, possible dependencies, and integrity constraints. Both flows result in a set of metadata that is stored in a repository.

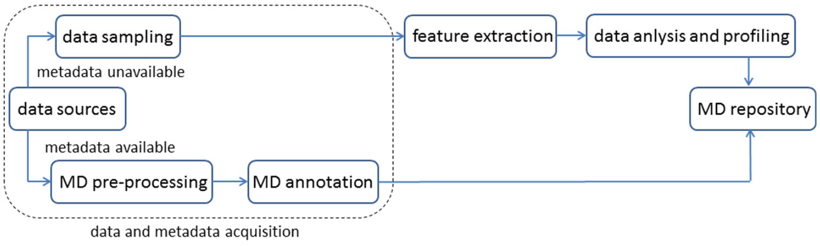


Fig. 1. The overall process for metadata discovery

Data Sources. In this research we use data sources on road safety (as flat files), provided by the *UK Gov* (2015–17) [31] and *Kaggle* [21] open data publishing facilities. The data files from UK GOV do not contain detailed metadata files, but only an overall dataset summary is provided with three primary files, (i.e. accidents, vehicles, and casualties) and a secondary context file containing labels and variables used inside the data. **Kaggle** (2005–2015) data files did contain partial metadata files containing information such as the total number of columns, data types, row count, creation date, and label definitions along with textual metadata(description, the domain of a dataset, the origin of a dataset, files inside a dataset). The metadata extraction and management is based on the identification of headers. Thus, for example, words as context, content, inspiration, and manifest are treated as metadata headers.

3.1 Step 1: Data and Metadata Acquisition

The very first step in the metadata discovery process is to ingest datasets and metadata files (if they exist) from different sources into one common repository.

Second, a mandatory metadata file search is carried throughout the repository. These metadata files have to be identified and tagged to ensure they are evaluated before the datasets are analyzed. These files are typically named as context files, descriptions, and read-me. The naming convention of metadata files helps in distinguishing them from the source file.

All of these are the available metadata and are tagged by our technique in a *descriptor table*. Each dataset has a descriptor table that gets populated based on the available metadata content, i.e., column descriptions, previews, data types, date of creation, data variables, and data labels. If there is no metadata, then it has to be discovered and filled accordingly. In our approach, we apply the so-called *truth table*. It represents the existence of various metadata types that are present in different data sources. If the dataset has a specific kind of metadata, the truth table adds a true value against its entry. If it doesn't have a particular metadata type, it adds a false value against the data source entry. Each metadata category is evaluated against data file, for example File Accidents2017 Administrative MD: 1, Technical MD: 0, Descriptive MD: 0. Similarly, for each file the truth table is populated accordingly. Once the truth table is filled, a need for metadata discovery is evaluated based on the context and domain. This means, that before discovering metadata, a straightforward and short preliminary investigation among datasets titles, files, and label evaluations are made to conclude if there is a possibility of relevance between the datasets.

3.2 Step 2: Feature Extraction

Feature extraction is one of the most important and contributing factors when it comes to understanding datasets with no predefined metadata. In this step, we analyze and manage attributes from different data sources and identify similarities that might be present among data features. For each data file in the repository, data feature extraction is initiated. The extracted lists consist of attribute names that are mostly strings or abbreviated letters. Each data source file is provided a unique ID through which its attribute names can be accessed for evaluation. Since attribute names are primarily composed of strings, an evaluation between different string elements can reflect the degree of similarity or dissimilarity. The techniques utilized to find the degree of similarity between two elements from different feature lists are the standard Jaccard [13] and Levenshtein [13].

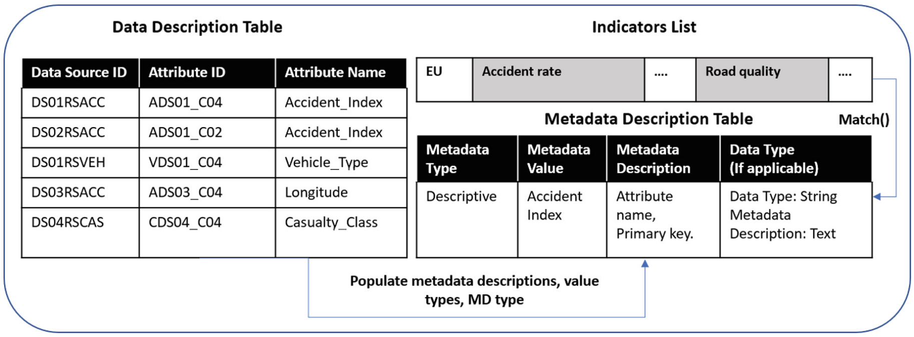
In details, feature extraction is run as follows. First, all columns from each data file are extracted separately into a *feature list* and *keyword list* - for similarity matching. A feature list is derived from attributes and may store combinations of words and special characters, e.g., Accident_Index. This list consists of pairs, triples, quadruples, ... of words that are combined together to represent meaningful information such as Accident_Index, indicating that the column contains index for all recorded accidents. An example feature list is shown below and it consists of composite words. Second, a keyword list is created that contains separate keywords (no pairs, triples, ...of words). For example, the keyword list given below was created by breaking the word pairs, triples and quadruples in the feature list. only singleton elements are permissible in the keyword list. The keyword list is connected with a dictionary that allocates synonyms to a selected keyword. For example, if the keyword ‘casualties’ was selected for comparison, the synonym dictionary would present five most commonly associated words with casualties. The keyword list is used for: (1) identifying if other datasets have similar words occurring in textual metadata or in their keyword lists, (2) checking if a given dataset includes data that could be used for answering a given user query. Let us assume that a user wants to answer the following query: How much road quality contributes to the rate of accidents in EU as compared to rest of the world? or How does gender play a role in accidents in the UK?. To this end, a query is decomposed and a *query indicators list* is generated, by breaking the query sentence into singleton words, excluding pronouns and stop words, by means of text mining rules and standards. Once the query list is populated automatically, we compare it with keyword list (including suggestions from synonym dictionary) and feature list.

feature list: {*Accident_Index, ..., Vehicle_Reference, ... Number_of_Casualties*}
keyword list: {*Accident, Index, Number, Casualties..., Vehicle, Reference,..., Longitude,*}
query indicators list: {*how, much, road, quality, contributes, rate, accidents, EU, compared, world, rest*}

The keyword list and query indicator list can then be compared with feature lists and metadata descriptions to find approximate or exact match. The query list is compared with the keyword list first to find the possibility of similar words indicating there might be relevance between the two lists. The query list and the keyword lists are then matched with textual metadata description (if available) to identify if there are any similar words appearing in the data descriptions, tags, and annotations. As discussed earlier, similarity matching and distance measures [13] are used for these evaluations. Once the evaluations are complete a judgement is provided whether the metadata from these lists are similar, partially similar, dissimilar, partially dissimilar or represent and exact match, based on a given similarity measures. As demonstrated in Fig. 2, the metadata description table (MDT) is created for each dataset. The data description table (DDT) is composed of a unique data source ID, a unique attribute ID, and attribute name. Each DDT has a corresponding metadata description table. The MDT is composed of the following columns: (1) Metadata type that defines the category of metadata (technical, administrative, descriptive, structural, customized), (2) Metadata value that represents the content or value of the metadata, e.g., Accident_Index is a metadata value, (3) Metadata description defines the value and its properties and, (4) Data type provides type specifications for data value and the metadata description. For example, the first dataset in DDT has a data source ID: DS01RSACC, with a corresponding attribute ID: ADS01_C04, i.e., Accident_Index.

It can be understood from Fig. 2 that there is a possible match in the second dataset with data source ID: DS02RSACC an attribute ID: ADS01_C02. Also, in the indicator list, the first letter for index[1], i.e., ‘Accident rate’ is a 50%

match, meaning that word ‘accident’ appears in the indicator list, column list, and keyword list. Thus, after the first match, the process is continued and further metadata is analyzed to identify more similar context based keywords with column names and indicators. The column lists are also compared among each other by the same evaluation measure.


Data Description Table
Indicators List
Metadata Description Table
Match()
Populate metadata descriptions, value types, MD type

Data Source ID	Attribute ID	Attribute Name
DS01RSACC	ADS01_C04	Accident_Index
DS02RSACC	ADS01_C02	Accident_Index
DS01RSVEH	VDS01_C04	Vehicle_Type
DS03RSACC	ADS03_C04	Longitude
DS04RSCAS	CDS04_C04	Casualty_Class

EU	Accident rate	....	Road quality	....

Metadata Type	Metadata Value	Metadata Description	Data Type (If applicable)
Descriptive	Accident Index	Attribute name, Primary key.	Data Type: String Metadata Description: Text

Fig. 2. An overview of descriptor tables and feature evaluations

If a similar attribute is identified between two or more datasets, the attribute is regarded as a qualifying feature, e.g., Accident_Index, which appears in more than one data file, and therefore the state of the feature is maintained temporarily. This state represents that there is an approximate or an exact feature in another dataset. Thus, a simple directed graph is created for that particular attribute, indicating its presence in different datasets. Once all the feature lists under consideration have been evaluated for similarity, the attribute analysis report is generated with their calculated similarities (Jaccard and Levenshtein). Next, we re-analyze the attribute truth table. This is done to make sure that metadata are properly managed and their changes over time are maintained. It also helps in identifying any change in attribute names, data type, values.

To manage a coherent system for attributes from different data sources, we follow IBM guide [23] on attribute management. Not all attribute management guidelines were incorporated as a part of our study. However, the most prominent guidelines were included, namely: (1) define attribute domain details, (2) add or correct attribute titles, (3) manage missing or incorrect value of an attribute, (4) create an attribute group, (5) modify an attribute group, (6) delete an attribute group, (7) delete an attribute, (8) create an attribute, and (9) define value for Boolean attributes.

As an example, let us consider the attribute management for *Accident_Index*, as shown in Table 1. For each attribute the management aspects are evaluated and populated accordingly Accident_IndexDomain (accidents file [01_01] [01_02], [01_03]), (Value: Not Null string), (Management: Convert to string), (Group: NA), (Modify: NA), (Delete: NA), (Delete Group: NA), (Create: NA), (Bool:

NA). If there is a data type change required, it is validated and recorded. Similarly, after data profiling attribute groups and dependencies such as FD, IND become available for use. These can further be utilized to populate metadata files, define data structure and allocate textual descriptions for discovered dependencies. Once all these feature lists have been created, we generate an *attribute management table* to find the change in attributes across different files of these datasets. In our example, three attribute management tables are formed, one for each category of data, i.e., casualties, accidents, and vehicles. Additionally, an *attribute reference table* is created. It indicates the presence of attributes for all datasets and their sub-categories (accidents, vehicles, casualties). For example for column Accident_Index the value for truth table would be (2017:1), (2016:1), (2015:1), (2014:1), (2005–2015:1) Once all attribute reference tables are populated automatically, the very first metadata is now available and, a simple comparison indicates a presence, absence, and change of an attribute across different data files.

Table 1. An example attribute management table

Text	Domain	Value	Mgmt.	Group	Modify	Delete	Del. Group	Create	Bool
Accident_Index	Accidents file [01_01] [01_02] [01_03] [01_04]	Not Null String	Convert to string	NA	NA	NA	NA	NA	False

3.3 Step 3: Data Analysis and Profiling

As the result of this step, selective data are analyzed to discover metadata. In order to avoid analyzing large datasets for metadata discovery, we propose to extract samples of data for analysis (in our experiments we select data portions ranging from 1% to 5% of all data in a given file). Furthermore, it is crucial to analyze the extracted data and extend the metadata files with annotations.

First, data analysis involves standard data profiling tasks such as primary key identification, foreign key identification, value types, ranges, minimum value, maximum value, categories, standard deviation, and missing values. The results are stored in a metadata file. Second, textual metadata, which describe the discovered metadata, are manually tagged in the header of a metadata file. This newly generated metadata file is stored in a metadata repository for comparison with other discovered metadata files using a similarity measure. Now, the discovered metadata are used as follows. First schema metadata are evaluated in order to decide whether the compared data files describe the same domain. If the similarity of schema metadata is above a given threshold (in the current evaluation 40%), it suggests that the files may come from the same domain. Second, the contents of these files are evaluated based on data samples. If the samples contain a given percentage of the same data (in the current evaluation 90%), it means that these files represent the same sets of data (with a high probability). Notice that, estimating the right threshold is not trivial and it is part of our forthcoming work. Step 3 ends the process of metadata discovery and annotation. Now, the data sources (data files) under evaluation are described by descriptive and technical metadata. These metadata can be accessed in order to decide which sources potentially could be used to answer user queries.

4 Experimental Results

The approach presented in this paper was evaluated by experiments. They were designed to test the ability to discover descriptive and technical metadata with minimum data analysis and a degree of uncertainty. In particular, the goals of the experiments were to: (1) discover (gather) structural and descriptive metadata as a part of EDA and data profiling, (2) attain textual metadata to facilitate keyword based matching, (3) analyze how the discovered metadata influence the possibility of identifying similarity between different datasets and queries. All experiments were designed in Python including data cleaning, preparation, and metadata pre-processing based on the numpy, scipy, pandas, and pandas_profiling libraries. The experiments were run on a PC: Intel(R) Core i7, 16 GB of RAM, and Windows 10.

4.1 Gathering Structural and Descriptive Metadata

We start by gathering different data files to classify them under categories (e.g., road safety data) and sub-categories (e.g., vehicles, casualties, accidents). To be able to identify data files that could provide data to answer some queries (without fetching data), we collect descriptive metadata (e.g., title, categories, file names, dates) along with the total row count of data file, as potentially duplicate datasets can be characterized by similar metadata and the number of rows. For each file we collect two measures, i.e., row count and the title of the file and create a set. This is the very first discovery of data characteristics.

The obtained metadata are shown in Table 2. In our experiment, we extract a total number of columns per file and then arrange these according to categories for the generated metadata file. The categories in the metadata file are the necessary descriptive metadata, i.e., context and associated time/date of a dataset creation, and publication.

4.2 Gathering Textual Metadata

The goal of this experiment was to gather textual metadata. They can be acquired through data and resource descriptions, entity documentations, schema definitions, to name a few. In particular our experiments collected textual metadata from attribute names, file names, and query texts. Based on the gathered textual metadata, we compared and cross examined these metadata to provide a measure of similarity for attributes, descriptions, domain names, genre, topic,

Table 2. Data identification for metadata collection

Data file name	Data file year	Rows, columns
Accidents2017	2017	(129982, 32)
Vehicles2017	2017	(238926, 23)
Casualties2017	2017	(170993, 16)
Acc2016	2016	(136621, 32)
Vcc2016	2016	(252500, 23)
Ccc2016	2016	(181384, 16)
Accidents2015	2015	(140056, 32)
Vehicles2015	2015	(257845, 23)
Casualties2015	2015	(186189, 16)
Accidents2005–2015	2005–2015	(1780653, 32)

and file names of different data sources under consideration. Table 3 presents the selected results from comparison of attribute lists. The attribute names were extracted and converted into the *keyword list* and the *attribute list*. These lists were compared with each other to identify similarity between them, using the Jaccard and Levenshtein distances.

Table 3. Attribute similarities comparison by means of the Jaccard and Levenshtein distances

Column from DS 1	Column from DS 2	Jaccard dist.	Levenshtein dist.
Accident_Index	Accident_Index	0.0	0
Local_Authority_(District)	Local_Authority_(District)	0.0	0
2nd_Road_Class	2nd_Road_Class	0.0	0
Junction_Detail	Junction_Detail	0.0	0
Pedestrian_Crossing-Physical_Facilities	Road_Surface_Conditions	0.478	31
Road_Surface_Conditions	Road_Surface_Conditions	0.0	0
id_Police_Officer_Attend_Scene_of_Accident	Did_Police_Officer_Attend_Scene_of_Accident	0.063	1

As discussed earlier, for understanding our data better and to discover different types of metadata we used data sampling. The random data sampling range was between 1% and 5%. The intuition behind selecting a small sample rate was to analyze limited data to attain metadata or develop understanding of data. Our mission is to discover the metadata possibilities such as keywords, sample data types etc. It is thus counter-intuitive to analyze large complex datasets. Thus, we opted for a small sampling percentage. Also, to manage the sampling bias we used ensemble techniques but that is beyond the context of this paper. Table 4 identifies the first pass, which selected only 1% from all accidents file considered from each year (only some results are shown). The table indicates some of the ranges of data selected for different data files from a collection of datasets. It also provides the selected sample data count, the total keywords selected for each sample and a list of features evaluated.

Finally, we run simple (standard) exploratory data analysis to analyze attributes in the test files, w.r.t.: their data types and Null/Not Null characteristic. Figure 3 describes the process of row feature discovery with increasing samples to support our sampling strategy. Also, it provides analytical similarity between query and selected metadata files.

Table 4. The results of 1% data sampling

File name	Total rows	Data sample %	Total selected rows	Keywords	List
Accidents2017	129982	1.5%	1949	82	32 features
Accidents2016	136621	1.5%	2049	82	32 features
Accidents2015	140056	1.5%	2100	82	32 features
Accidents2005–2015	1780653	1.5%	26709	82	32 features

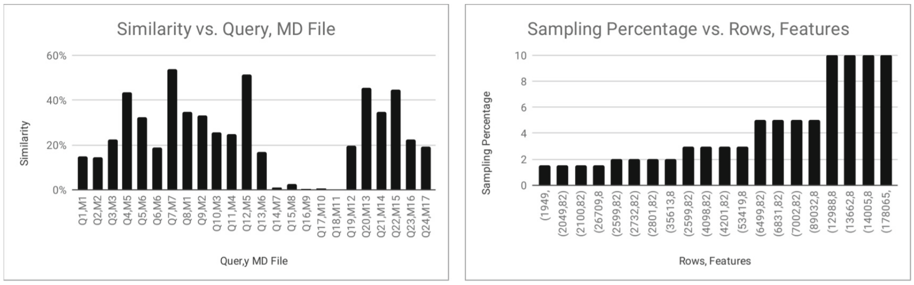


Fig. 3. The overview of data sampling and keyword discovery.

4.3 Metadata Applicability

To verify the applicability of the collected metadata for answering queries, we designed 24 queries that were evaluated against metadata files that our method created. The queries represent 6 different patterns, asking for: (1) the role of gender in accidents in time (queries Q01 to Q05), (2) numbers of accidents in time (Q09 to Q11), (3) increase/decrease in the number of accidents over time (Q06 to Q08), (4) the impact on road quality on accidents (Q14 to Q18), (5) the impact of weather conditions on accidents (Q15, Q19 to Q22), (6) accident analysis in regions over time (Q12, Q13).

Each metadata file was evaluated for each query to see if there were any similarity between the semantics (represented by its predicates) of the query and the content of a metadata file. Table 5 presents the obtained similarity results. Each query is defined by a unique query ID such as Q01, followed by a metadata file ID, e.g., MD_001. The final column indicates the similarity between the semantics of the query and the metadata files. This is attained by evaluating keywords, description paragraphs, the attribute lists, and the dependency list (the list storing all discovered dependency metadata, i.e, inclusion, functional, multi-column).

Table 5. A sample of evaluation by percentage similarity between different metadata files using generated metadata files

Query ID	MD file	Similarity	Query ID	MD file	Similarity
Q01	MD_001	15%	Q13	MD_006	16.87%
Q02	MD_002	14.3%	Q14	MD_007	1.04%
Q03	MD_003	22.5%	Q15	MD_008	2.6%
Q04	MD_004	43.5%	Q16	MD_009	0%
Q05	MD_005	32.5%	Q17	MD_010	0.4%
Q06	MD_006	18.9%	Q18	MD_011	0.005%
Q07	MD_007	54%	Q19	MD_012	19.8%
Q08	MD_001	34.6%	Q20	MD_013	45.6%
Q09	MD_002	33.2%	Q21	MD_014	34.6%
Q10	MD_003	25.6%	Q22	MD_015	44.5%
Q11	MD_004	24.8%	Q23	MD_016	22.5%
Q12	MD_005	51.45%	Q24	MD_017	19.07%

The results clearly demonstrate the advantage of using EDA and data profiling for discovering metadata and for identifying the cases of data integration. To prove that discovered metadata is coherent and relevant, we created manual descriptions for each metadata file associated with a dataset. To evaluate the discovered keywords and description we matched the manual descriptions with our discovered keywords. The discovered keywords for different datasets had different outcomes. For instance, the discovered keywords from year 2017 were a 65% match with the manual descriptions. Similarly, the keywords from year 2015 were a 76.89% match with the manual descriptions.

5 Conclusion and Future Work

In this paper, we proposed the first step towards a (semi-)automatic metadata discovery and generation that allows users to analyze samples of data to produce metadata collectively and provide sufficient annotations. To the best of our knowledge, there are no automatic or semi-automatic tools and algorithms that generate or discover metadata in heterogeneous data sources. In this first attempt, we applied our technique to data sources implemented as data files without or with some metadata available. With the experiments that assessed our method, we demonstrated that descriptive and technical metadata can be automatically discovered with the help of exploratory data analysis and data profiling, identifying similar aspects, features, and constructs. We were able to identify similarities between different files, and we were able to identify domains or topics of various data files. Thus, we were able to suggest (with a given similarity measure) data sources that were the most suitable for given queries, based on query semantics. While metadata discovery should not confine to certain categories, the discovery of essential characteristics with a given uncertainty factor is far more valuable and usable then the case of having no metadata. To be able to discover metadata automatically or semi-automatically for large repositories of heterogeneous data is a fundamental functionality [6,29], required in big data integration architectures, especially in a data lake [27]. Since our approach turned out to provide a promising functionality (although yet simple in this first attempt), in the future we will extend it to apply more advanced metadata discovery techniques based on: (1) clustering to automatically distinguish metadata categories and (2) artificial intelligence, active learning in particular, for automatic data labelling discovered by means of clustering.

Acknowledgements. The work of Hiba Khalid is supported by the European Commission through the Erasmus Mundus Joint Doctorate project *Information Technologies for Business Intelligence-Doctoral College* (IT4BI-DC).

The work of Robert Wrembel is supported from the grant of the Polish National Agency for Academic Exchange, within the Bekker programme.

References

1. Sakr, Sherif, Zomaya, Albert Y. (eds.): Encyclopedia of Big Data Technologies. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-77525-8
2. Abedjan, Z., Golab, L., Naumann, F.: Data profiling. In: IEEE International Conference on Data Engineering (ICDE), pp. 1432–1435 (2016)
3. Aindrila Ghosh, J.M., Nashaat, M.: A comprehensive review of tools for exploratory analysis of tabular industrial datasets. Vis. Inform. **2**, 235–253 (2018)
4. Bauckmann, J., Leser, U., Naumann, F.: Efficiently computing inclusion dependencies for schema discovery. In: International Conference on Data Engineering Workshops, p. 2 (2006)
5. Bouguettaya, A., Benatallah, B., Elmargamid, A.: Interconnecting Heterogeneous Information Systems. Springer, Boston (1998). https://doi.org/10.1007/978-1-4615-5567-4. Kluwer Academic Publishers, ISBN 0792382161
6. Ceravolo, P., et al.: Big data semantics. J. Data Semant. **7**(2), 65–85 (2018)
7. Chen, C.L.P., Zhang, C.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf Sci. **275**, 314–347 (2014)
8. DublinCore: Dublin core metadata initiative. http://dublincore.org/specifications/dublin-core/

9. Duggan, J., et al.: The BigDAWG polystore system. SIGMOD Rec. **44**(2), 11–16 (2015)
10. Edvardsen, L.F.H.: Using the structural content of documents to automatically generate quality metadata. Ph.D. thesis, Norwegian University of Science and Technology (2013)
11. Ehrlich, J., Roick, M., Schulze, L., Zwiener, J., Papenbrock, T., Naumann, F.: Holistic data profiling: simultaneous discovery of various metadata. In: International Conference on Extending Database Technology (EDBT), pp. 305–316 (2016)
12. Elmagarmid, A., Rusinkiewicz, M., Sheth, A. (eds.): Management of Heterogeneous and Autonomous Database Systems. Morgan Kaufmann, San Francisco (1999)
13. Gali, N., Mariescu-Istodor, R., Frnti, P.: Similarity measures for title matching. In: International Conference on Pattern Recognition (ICPR) (2016)
14. Gallinucci, E., Golfarelli, M., Rizzi, S.: Schema profiling of document-oriented databases. Inf. Syst. **75**, 13–25 (2018)
15. Halevy, A.Y., et al.: Goods: organizing google's datasets. In: ACM SIGMOD International Conference on Management of Data, pp. 795–806 (2016)
16. Hewasinghage, M., Varga, J., Abelló, A., Zimányi, E.: Managing polyglot systems metadata with hypergraphs. In: International Conference on Conceptual Modeling (ER), pp. 463–478 (2018)
17. IEEE: IEEE LOM: IEEE standard for learning object metadata. https://standards.ieee.org/standard/1484_12_1-2002.html
18. IEEE Standards Association: IEEE Big Data Governance and Metadata Management (BDGMM). https://standards.ieee.org/industry-connections/BDGMM-index.html
19. IEEELO: IEEE standard for learning object metadata. https://ieeexplore.ieee.org/document/1032843
20. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-662-05153-5
21. Kaggle: UK car accidents 2005–2015. https://www.kaggle.com/silicon99/dft-accident-data
22. Kolaitis, P.G.: Reflections on schema mappings, data exchange, and metadata management. In: ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 107–109 (2018)
23. Kunz, M., Puchta, A., Groll, S., Fuchs, L., Pernul, G.: Attribute quality management for dynamic identity and access management. J. Inf. Secur. Appl. **44**, 64–79 (2019)
24. Liu, M., Wang, Q.: Rogas: a declarative framework for network analytics. In: International Conference on Very Large Data Bases (VLDB), vol. 9, no. 13, pp. 1561–1564 (2016)
25. March, F.D., Lopes, S., Petit, J.-M: Efficient algorithms for mining inclusion dependencies. In: International Conference on Extending Database Technology (EDBT), pp. 464–476 (2002)
26. Poole, J., Chang, D., Tolbert, D., Mellor, D.: Common Warehouse Metamodel. Wiley, Developer's Guide (2003)
27. Russom, P.: Data lakes: purposes, practices, patterns, and platforms (2017). TDWI white paper
28. SCORM: Scorm metadata structure. https://scorm.com/scorm-explained/technical-scorm/content-packaging/metadata-structure/
29. Stefanowski, J., Krawiec, K., Wrembel, R.: Exploring complex and big data. Appl. Math. Comput. Sci. **27**(4), 669–679 (2017)

30. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data wrangling: the challenging journey from the wild to the lake. In: Conference on Innovative Data Systems Research (CIDR) (2015)
31. UK Gov.: Road safety data. https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data
32. Varga, J., Romero, O., Pedersen, T.B., Thomsen, C.: Analytical metadata modeling for next generation BI systems. J. Syst. Softw. **144**, 240–254 (2018)
33. Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Comput. **25**(3), 38–49 (1992)
34. Wu, D., Sakr, S., Zhu, L.: HDM: optimized big data processing with data provenance. In: International Conference on Extending Database Technology (EDBT), pp. 530–533 (2017)
35. Wylot, M., Cudré-Mauroux, P., Hauswirth, M., Groth, P.T.: Storing, tracking, and querying provenance in linked data. IEEE Trans. Knowl. Data Eng. (TKDE) **29**(8), 1751–1764 (2017)

Modelling

Conceptual Models and Their Foundations

Bernhard Thalheim(✉)

Department of Computer Science, Christian-Albrechts University at Kiel, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de
http://www.is.informatik.uni-kiel.de/~thalheim

Abstract. There is no common agreement which artifact should (not) be considered to be a conceptual model although the term 'conceptual model' is used for more than for five decades in computer science and for more than one century in science and engineering. A team from all faculties at our university has been able to develop a notion of model that covers all model notions known in the disciplines of this team. We now introduce three notions of conceptual model in this paper: light, slim, and concise versions of the notion of conceptual model.

The paper answers the following questions: Are all models also conceptual models? What is a conceptual model? Is there a formal notion of a conceptual model? What is not yet a conceptual model? What will never be a conceptual model? What is a concept? Which philosophical and scientific foundations we should consider while modelling? Is the existence of an ontology a necessary prerequisite for the being as conceptual model?

Keywords: Conceptual model · Conceptualisation · Concept · Conceptions · Model theory

1 The Model

Humans have learned to use instruments for handling their issues, tasks, and problems in daily life. Sciences and engineering also widely use instruments. Human evolution, sciences, and engineering have been enabled by wide instrument utilisation. The language is one of these instruments – often seen as one of the main. Models are another main instrument in modern computer science and computer engineering (CS&CE). They are often material artifacts. They might, however, also be immaterial or virtual.

It is surprising that models and modelling (and its variants such as conceptual models) have not yet properly founded. This paper contributes to close this lacuna.

1.1 Models Are Main Artifacts and Universal Instruments

Models have become one of the main artifacts in CS&CE. This wide usage has not led to a common agreement about the notion of a model. The same observation can be made for other scientific disciplines, for engineering, and for daily life. In our area models became as artifacts the main instrument for system and software construction.

Models might be combined with other artifacts[1]. Concept and conception development might be integrated into models. In this case, models might be considered as conceptual models.

A Notion of Model

What is a Model? According to [9,32,35] we define the model notion as follows:

"*A* **model** *is a well-formed, adequate, and dependable instrument that represents origins and that functions in utilisation scenarios.*

Its criteria of well-formedness, adequacy, and dependability must be commonly accepted by its community of practice (CoP) within some context and correspond to the functions that a model fulfills in utilisation scenarios."

Well-formedness is often considered as a specific modelling language requirement. The criteria for adequacy are analogy (as a generalisation of the mapping property that forms a tight kind of analogy), being focused (as a generalisation of truncation or abstraction), and satisfying the purpose (as a generalisation of classical pragmatics properties). The generalisation of [12,19,21,28] is necessary for consideration of model-being.

The model has another constituents that are often taken for granted. The model is based on a background, represents origins, is accepted by a community of practice, and follows the accepted context. The model thus becomes dependable, i.e. it is justified or viable and has a sufficient quality. Justification includes empirical corroboration, rational coherence, falsifiability (in our area often treated as validation or verification), and relative stability. In our area the quality characteristics can be based on software quality characteristics and procedures for evaluating these characteristics.

Scenarios Determine Functions of Models as Instruments. A model is utilised. This utilisation is bound to scenarios in which a model functions. Typical scenarios are system construction (with description, prescription, and coding subscenarios), communication, negotiation, conceptualisation, documentation, and

[1] Due to the utilisation of artifacts as instrument we will concentrate on the instrument being of artifacts. This approach allows us to additionally consider virtual 'artifacts' such as mental models. An artifact is "something created by humans, usually for practical purpose. It is a product of artificial character due to extraneous (as human) agency". [4]. Furthermore, models can be real artifacts as well as thoughts. An additional difficulty is the negative usage of "artifact" in engineering as artificially introduced change (e.g. in presentation, miss or imperfection).

learning. The model might have several functions in complex scenarios. For instance, a model functions as a blueprint for realisation in a prescription scenario. Other typical functioning are the usage as an informative means, as a companion, as a guide for development. The quality of a model must be sufficient for this usage. Therefore, models used for description and models used for prescription might be different.

The main qualification of models is the potential utilisation as an instrument. This utilisation is based on methods which are developed in the discipline.

Models Are Used in Sciences, Engineering, and Daily Life

Models and Model Suites. There is no CS&CE subdiscipline that does not use models. Since models are abstractions and more generally are focused they are far better to use for investigation and system development. They are used in problem solving scenarios, in social scenarios, in engineering scenarios, and in science scenarios in a wide variety of forms. Often, models either consist of sub-models or form a model suite what is a well associated ensemble of sub-models. The models in a model suite [6,7] coexist, co-evolve, and support solutions of subtasks.

Models Are One of the First Instruments Before Languages. [11] Daily life utilisation of models is often unconscious, subconscious or preconscious. One of the first models that is learned by everybody is the 'model of mother'. It is used before we spell the word 'mother'. It has is variety of interpretations depending on the kind of behaviour of the mother. Models might be perception models that allow to summarise observations.

Matrices and Deep Models. Models typically consist of a relative stable part and of a part that is dependent on the actual circumstances. Typical modelling languages in CS&CE are predefined, use a limited vocabulary, have a relatively fixed – at least lexical – semantics, and allow to express certain aspects. They use their own techniques in some stereotype way, i.e. their utilisation follows some mould. Origins of models are often mental models such as perception or domain-situation models. The model background forms the deep sub-model [34]. The current model is then the 'rest', i.e. a normal model. The utilisation and the mould form the matrix of the model.

Memes as Basic and Deep Models. Humans reason, memorise, and express their thoughts based on memory chunks. Some of the chunks are relatively persistent and become memes [3,29] which are then units of cultural evolution and selection. These memes are combined with some identification facilities. They represent a number of properties. They may be combined with other memes. They may be activated and deactivated. They can be grouped. They become reasoning instruments. Memes are thus already models, in most cases primitive or basic mental ones.

1.2 Why There Is No Commonly Accepted Notion of a Conceptual Model: 1001 Notions and 101 Scenarios

Why the Large Variety of Notions of Model? Database and information systems research communities are extensively using the term "conceptual model"[2]. Already [33] discusses 60 different notions of conceptual model. Although ER conferences are organised since 40 years[3], ***no*** notion of conceptual model has been coined by this community.

The variety of notions of model in CS&CE is far larger. Each of these notions concentrates on some aspects and implicitly assumes other properties. Simulation research found a common definition: "*The conceptual model is a concise and precise consolidation of all goal-relevant structural and behavioral features of the system under investigation (SUI) presented in a predefined format.*" [26]

The implicit and hidden usage of deep models and the corresponding matrices is one – if not the main – cause for the manifold of model notions in CS&CE.

May We Develop a Common Understanding of the Notion of Model? The two main sources for the variety of notions allow a systematic harmonisation of model notions. The definition given above is a result of a discussion on models in agriculture, archeology, arts, biology, chemistry, computer science, economics, electrotechnics, environmental sciences, farming, geosciences, historical sciences, languages, mathematics, medicine, ocean sciences, pedagogical science, philosophy, physics, political sciences, sociology, and sports at Kiel university that continues now for almost 10 years. The discussion is summarised in the compendium [36]. We got a shared understanding of the notion of model, of model activities and of modelling. So, we can envision that a common understanding and a coherent collection of notions of model can be developed. The collection supports a coherent notion that allows to concentrate on the specific utilisation scenario and the specific functions that a model has to fulfill.

1.3 The Storyline of the Paper and Our Agenda

Tasks and Foundations of a Theory of Conceptual Models. This paper bases the notion of conceptual models on four observations: (I) Conceptual models integrate concept(ion)s from a conceptualisation into a models. A notion of conceptual model might be a slim, light, or concise one depending on the level of

[2] Facetted search for the term "conceptual model" in DBLP results in more than 6.000 hits for titles in papers (normal DBLP search also above 3.500 titles).

[3] The ER modelling language has been introduced in 1976. The first and primary utilisation scenario was documentation of relational structure. Later, conceptualisation has been considered to be the main issue. The conferences on conceptual modelling started in 1979 (First as ER conferences; since 1996 the series got its name with the ER acronym.) The formalisation and a proper definition of the ER modelling language occurred more than 15 years later [30]. One might claim that it is nowadays too early to define the notion of conceptual model.

detail we need in model utilisation. (II) A conceptualisation is based on a collection of concepts. (III) Origins of conceptual models are perception models and domain-situation models. (IV) These origins are formed by our understanding of the world, i.e. our observations and our compilations of these observations. We shall answer questions 2, 3, 6–8 from the abstract in Sects. 3 and 4 and use these answers for answering questions 1, 4, and 5 at the end.

We start with the last observation that leads us back to Ancient Greece. Next we develop an enhanced theory of concepts for an understanding of a conceptualisation. We may now define what are the components of perception and domain-situation models. Finally we arrive with three notions of conceptual model. We thus head forward to a science and culture of modelling in Fig. 1.

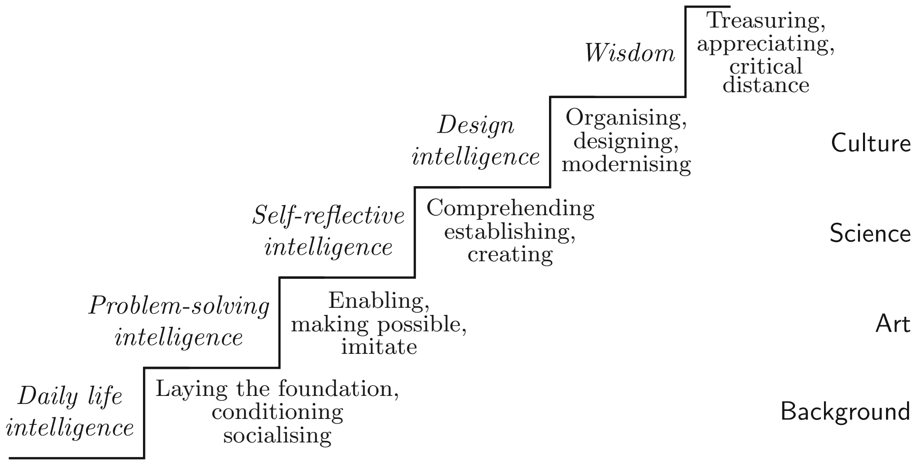

Wisdom
Treasuring, appreciating, critical distance
Design intelligence
Organising, designing, modernising
Culture
Self-reflective intelligence
Comprehending establishing, creating
Science
Problem-solving intelligence
Enabling, making possible, imitate
Art
Daily life intelligence
Laying the foundation, conditioning socialising
Background

Fig. 1. The five levels of modelling as art, science, and culture

Towards a Science and Culture of Models, Modelling Activities, and Modelling. Modelling is currently a creative art, i.e. a skill acquired by experience and observation, and may potentially be enhanced by study. The art extends daily life intelligence that is a part which lays the foundations for modelling, conditions, and socialises. The first level of modelling is based on daily life intelligence between humans or of humans with their environment. Humans become introduced to the deep model and especially the background – in most cases at some preconscious level.

Model science is based on a system of knowledge that is concerned with modelling art and that entails unbiased observations and systematic experimentation. The foundation we envision orients on fundamental laws. We understand, establish, deliberately apply the knowledge, and formalise it. Modelling culture is shared in a community of practice, is based on well-developed principles and methods as well as on established guidelines and practices. Wisdom requires the sapience of schools and matured modelling.

2 Model Theory and Its Philosophical Foundations

The earliest source of systematic model consideration we know is Heraclitus [18] with his concept of $\lambda\acute{o}\gamma o\varsigma$ (logos). Explicit model development and model deployment is almost as old as the mankind, however. For instance, Ancient Egyptians already made sophisticated use of models [8]. So, essentially, it is an old subdiscipline of most natural sciences and engineering with a history of more than 5000 years [22]. The notion of model has not been explicitly used at that historic time. It was, however, the basis of understanding, manipulation, and engineering.

2.1 Plato's Three Analogies

Plato's Republica (for a survey on Politeia see [2] or Lafrance [16]) uses in the sixth book three analogies which can essentially be understood as the underpinning of the concept of model. We follow here Lattmann's [17] investigation that led to a deep revision of the interpretation by Aristoteles.

The three analogies provide a general understanding of the model-being, of models, and of modelling. We may only observe phenomena of reality, form then trusts in beliefs and observations, next develop conjectures, might next judge and hypothise, and finally provide explanations.

The analogy of the sun distinguishes the 'good' visible world and the intelligible world. The sun stands for the visible things (i.e. 'empirical', 'physical', and 'material' world) and gives the light. The opposite world is the world of reasoning, of beliefs, conjectures, ideas, and explanations.

The analogy of the divided line distinguishes the visible world and the reasoning about this world as the thinkable (called intelligible world). The visible world can be separated into the physical things themselves (beliefs (pistis) about physical things) and the reflections and observations about them (called shadows) (eikasia as the illusion of human experience). The intelligible world consists of (mathematical) reasoning and thought (dianoia) and of deep understanding (moesis).

Plato represented these four dimensions by a line (reflections (AB) - physical_things (BC) - thoughts (CD) - understanding (DE)). We shall see in the sequel that a six plane representation allows deeper understanding.

The analogy of the cave explains why humans can only interpret the world based on their observations, i.e. shadows that we can see. The reality cannot be observed.

2.2 Revisiting the Analogies for Understanding the Model World

The four segment line presentation in Plato's analogies (AB, BC, CD, DE) can be transferred to a six plane meta-model with

– a separation into a quantitative area and a qualitative area for meaning and opinion (doxa) and a qualitative area for thoughts (noesis) from one dimension and
– a separation into a perception and pre-image area and an area for conceptualisation from the other dimension.

The intelligible world thus consists of the mental world and the real intelligible world. This observation allows us to reconsider the model-being in the approach depicted in Fig. 2.

The visible model world mainly reflects the observations and their perception as phenomena. So, we can consider models of the visible world as models of the first generation.

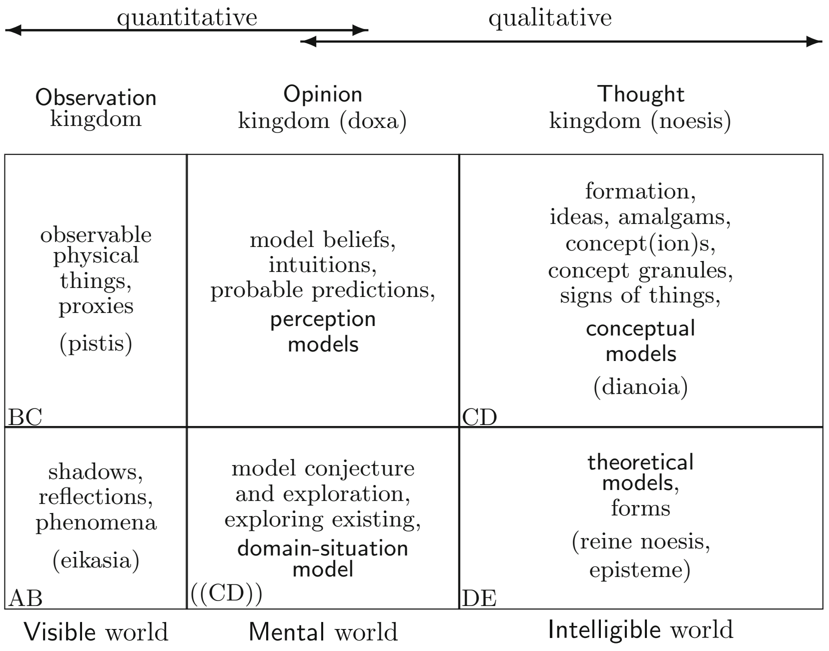

quantitative
qualitative
Observation kingdom
Opinion kingdom (doxa)
Thought kingdom (noesis)
observable physical things, proxies
(pistis)
BC
model beliefs, intuitions, probable predictions,
perception models
formation, ideas, amalgams, concept(ion)s, concept granules, signs of things,
conceptual models
(dianoia)
CD
shadows, reflections, phenomena
(eikasia)
AB
model conjecture and exploration, exploring existing,
domain-situation model
((CD))
theoretical models, forms
(reine noesis, episteme)
DE
Visible world
Mental world
Intelligible world

Fig. 2. The model world with the separation of concern into visible, mental, and intelligible worlds

The mental model world consists already of compilations to perception models that reflect someone's understanding or domain-situation models that represent a commonly accepted understanding of a state of affairs within some application domain.

The intelligible model world includes conceptualisations and theory development.

3 Concepts and Conceptualisations

The separation of model worlds in Fig. 2 provides a means to distinguish clearly between models and conceptual models. With this distinction we may now neglect the hypothesis [24] that any model is a conceptual model. We thus solved the demarkation problem for distinction of models into perception, domain-situation, conceptual, and theoretical models.

3.1 Conceptualisation

Conceptualisation is a reflection and understanding of the world on the basis of concepts from some commonly accepted concept spaces. Similar to ontologies, a conceptualisation is never unique. There is no universal conceptualisation such that every other one can be transformed from it. Conceptualisation means to find adequate concepts and conceptions for representation of a visible and mental world. It aims at the development of knowledge about these worlds. It is based on derivation of abstract concepts and experience, of (scientific) understanding and perception that can be applied in similar worlds, of (pragmatical) experience for modelling, and of reference models for model-driven development (MDD) approaches.

Weakening the Rigidity of Classical Concept Theory

The word 'conceptual' is linked to concepts and conceptions. Conceptual means that a thing, e.g., artifact is characterised by concepts or their conceptions. The word 'conceptional' associates a thing as being or of the nature of a notion or concept. Therefore, we distinguish the 'conceptual model' from 'conceptional modelling'. Classical concept theory and concept systems in mathematical logics are based on a Galois relationship between extensions and intensions of concepts, i.e. a concept is defined as a pair of an intention and of an extension where the intention is fully characterised by the extension and the extension is fully described by the intention. Each definition of a concept is a logical equation consisting of a definiendum and a definiens. We will use here an extension of the classical theory of concepts (e.g. [23]) by R. Kauppi's theory of concept properties [13,31]).

Brentano, Bolzano and Twardowski (e.g. [5,20]) distinguish three kinds of mental phenomena and inner consciousness: ideas, judgements, and volitions. Concepts and conceptions might be based on prototypes that allow to partially characterise the current understanding of the intention but do not provide a complete characterisation. Mental phenomena, beliefs, and intuitions have their prototype view and a representation through best (counter-)examples that a person has been observing. Moreover, they can be represented by an exemplary view that characterises exemplars through similarity relation with measures, weights, and stimuli for their acceptance.

Conceptions are Systems or Networks of Explanation. White [38] has already observed that concepts are not the same as conceptions. Concepts can be used in the meaning of classification and as an abstraction of a set of knowledge a person associates with the concept's name. Conceptions are however systems or networks of explanation. Conceptions are thus far more complex and difficult to define than the either meanings of the concept.

Conceptional modelling is modelling with associations to concepts. A conceptual model incorporates concepts into the model. Conceptual structures include conceptions (concepts, theoretical statements (axioms, laws, theorems, definitions), models, theories, and tools). Concepts are linked together in a complex multi-dimensional network (is-a-kind-of, is-a-part-of, ...). The links are of variable strength.

3.2 Concepts for Conceptualisation

An advanced concept notion must allow to define a concept in a variety of ways. Some definitions might be preferred over others. They can be application and time dependent, might have different level of rigidity, have their validity area, and can only be used with a number of restrictions. We combine R. Kauppi's theory of concept features with the concept treatment by Murphy [23].

The Definition Frame for Concepts [27]: Concepts are given by tree-structured structural expression of the following form

ConceptTree(StructuralTreeExpression (Feature, Modality(Sufficiency, Necessity),
Fuzziness, Importance, Rigidity, GraduationWithinExpression, Category))) .

Features are elements of a concept with some modality, some Fuzziness, importance, rigidity, some graduation and some category. A feature is either a basic feature or is a concept.

Concepts are typically hierarchically ordered by the way how they are defined and can thus be layered. We assume that this ordering is strictly hierarchical and that the concept space can be depicted by a set of concept trees.

A Concept Might be Given by Several Definitions. A concept is also dependent on the community that prefers this concept. Consider, for instance, the mathematical concept of a set by an enumeration of its elements, by inductive definition of its elements, by an algorithm for the construction of the set, or by explicit description of the properties of the set. Which of the definitions is more appropriate depends on the application domain.

Interleaved Meta-hypergraphs form Hyper-networks of Concepts: Our definition frame has the advantage that concepts which share features can be decomposed into the shared feature collection and the rest. Therefore, we may base our concept collection on a number of basic concepts.

The network of concepts is a meta-hypergraph [25]

$$MG = (MG^V, MG^{MV}, MG^E, F^{MG}, \Sigma^{MG}) \tag{1}$$

with a set of meta-hypergraph vertices MG^V, a set of meta-hypergraph meta-vertices MG^{MV} which are subsets of meta-hypergraph vertices, a set of meta-hypergraph edges MG^E connecting vertices. A vertex and an edge is described by a set of features F^{MG}. The semantic restrictions are given by Σ^{MG}. An example of a meta-hypergraph is displayed in Fig. 3[4].

Conceptions Can Now be Defined as a Layered Ensemble of Meta-hypergraphs. We start with a primary network at layer 0 and associate next layer networks by embedding mappings to a hyper-simplex from networks at lower layer. A simple example is displayed in Fig. 4.

3.3 Meta-hypergraph Concept Worlds

Many sciences such as archaeology start with an explorative and investigative theory development [1], i.e. with an investigation of data sources. A middle-range theory offer is going to be developed based on some proxy-based observation concepts. The CRC 1266 [1,15] aims at exploring and explaining transformations in societies as "processes leading to a substantial and enduring re-organisation" [1] of any or all aspects of the human social, cultural, economic, and environmental relations. These concepts are interrelated, partially overlapping, and form

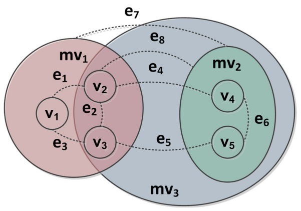


Fig. 3. A meta-hypergraph with vertices $v_1, ...v_5$, meta-vertices mv_1, mv_2, mv_3, and edges $e_1, ..., e_7$ without explicit features.

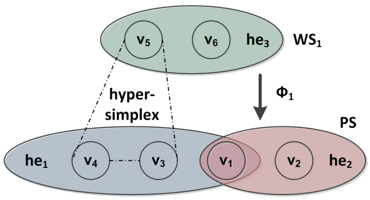


Fig. 4. A meta-hypergraph ensemble associating a simple primary network simplex PS and a first-order network simplex that associates via Φ_1 the vertex v_5 with a hyper-simplex of vertices v_4, v_3.

[4] We acknowledge the communication with J. E. Gapanyuk from Bauman Moscow State Technical University (10.10.2018) who proposed this illustrations in Figs. 3 and 4.

more complex concepts as a combination of less complex ones. A typical concept map is displayed in Fig. 5. It is an example of a meta-hypergraph with concepts that partially form a hypernode which is again interrelated to sub-nodes of a hypernode by meta-hypergraph vertices.

3.4 Concept Granules as Basic Constructs of Conceptualisations

Concept granules are collections of concepts and/or conceptions given as meta-hypergraphs and ensembles with specified typicality of features (typical, moderately typical, atypical, borderline), with specified relevance of concept features, and with assigned importance of concept features.

Conceptualisation enhancements of a given model consist of

(1) a context given for various aspects in dependence on the matrix,
(2) a concept granule with several interrelated expressions as alternatives (competing, ...), with abstracts, with extensions (motivation, explanation, ...), and
(3) witnesses as collections of illustrating best (counter-)examples (potentially with several concept trees) mainly based on images/observations on origins.

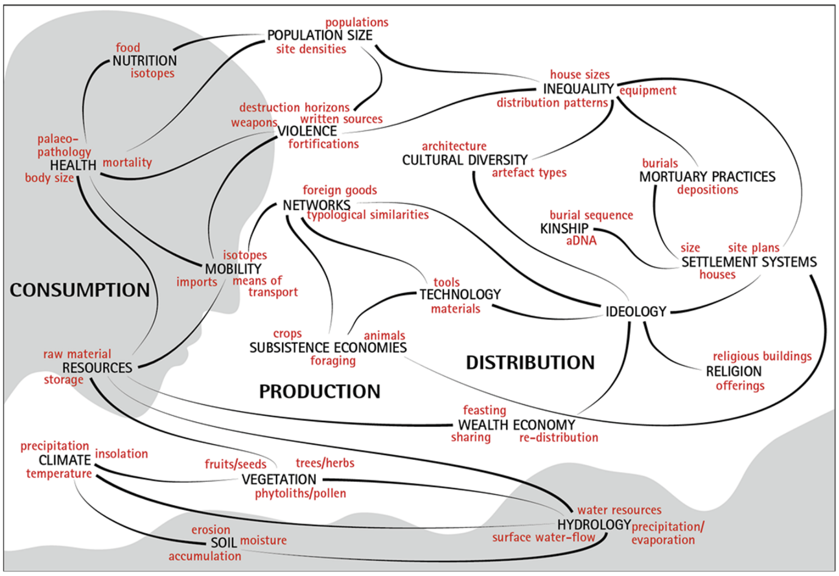


Fig. 5. The network of main concepts investigated in the CRC 1266

4 Conceptual Models

Mental models and their elements may be associated to concepts. The elements of a model are interpreted by concepts and conceptions. This interpretation is based on a judgement by a partner that conceives model elements as concept(ion)s within a certain model utilisation scenario. If the scenario changes then the association to concepts changes as well. [33] categorises more than 50 notions of conceptual model depending on the function that a conceptual model has in a given scenario. We use this categorisation and develop now three complementary notions of conceptual model. Which one is used depends on the complexity of models in scenarios considered.

4.1 Perception and Domain-Situation Models as Origins

Perception and domain-situation models [33] in Fig. 2 are specific mental models either of one member (e.g. perception model) or of the community of practice (domain-situation models) within one application area. It is not the real world or the reality what is represented in a perception model. It is the common consensus, world view and perception what is represented. Perception models are dependent on the observations, imaginations, and comprehension a human has made. Domain-situation models describe the understanding, observation, and perception of a domain situation, e.g. of an application domain. The description is commonly accepted within a community of practice.

4.2 The Notion of Conceptual Model

The large variety of notions of conceptual model is caused by the scope of modelling, by the application case under consideration, by the main scenario in which the model functions, by the variety of origins that are represented by the conceptual model, by modelling languages, by the stand-alone orientation instead of integration into a model suite, and by the focus on normal models without mentioning the underpinning by a deep model. It is now our goal to consolidate three versions in such a way that they form a view depending on the level of detail and abstraction. The notions can be refined to an application domain, e.g. to database modelling: "*A conceptual database model is a conceptual model that represents the structure and the integrity constraints of a database within a given database system environment.*" [35]

The Slim, Light, and Concise Notion of Conceptual Model

Slim Version: Conceptual Model $\sqsupseteq$ *Model* $\uplus$ *Concept(ion)s*[5] [35]: A conceptual model incorporates concepts into the model.

That means that models are enhanced by concepts from a number of concept(ion) spaces.

[5] $\uplus$ is depicts the disjoint union. $\oplus$ denotes the combination. $\bowtie$ denotes the integration (e.g. join) (see [30] for database operations).

Light Version: Conceptual Model $\sqsupseteq$ *Model* $\bigoplus$ *Concept(ion)s* [33]: A conceptual model is a concise and function-oriented model of a (or a number of) perception and/or domain-situation model(s) that uses a concept(ion) space.

This notion generalises and enhances a notion that is used in simulation research [26]: *"A conceptual model is a concise and precise consolidation of all goal-relevant structural and behavioural features of a system under investigation presented in a predefined format."*

Concise Version: Conceptual Model $\sqsupseteq$ *(Model* $\bigoplus$ *Concept(ion)s)* $\bowtie$ *Enabler* [10]: A conceptual model is a model that is enhanced by concept(ion)s from a concept(ion) space, is formulated in a language that allows well-structured formulations, is based on mental/perception/situation models with their embedded concept(ion)s, and is oriented on a matrix that is commonly accepted.

The conceptual model of an information system consists of a conceptual schema and of a collection of conceptual views that are associated (in most cases tightly by a mapping facility) to the conceptual schema [37]. Conceptual modelling is either the activity of developing a conceptual model or the systematic and coherent collection of approaches to model, to utilise models, etc.

Literate programming [14] considers a central program together with its satellite programs, esp. for interfacing and documenting. This paradigm has become the basis for GitHub and model suites. Conceptual modelling is typically explicit modelling by a model suite. Association of conceptual and other models in a model suite might follow the layered approach to model coherence maintenance and to co-evolution of models.

Descriptive and Prescriptive Conceptual Models

A Model Functions in a Number of Scenarios. For instance, the conceptual model is used in documentation, negotiation, learning, communication, explanation, discovery, inspiration, modernisation, reflection, and experience propagation scenarios. We may categorise and enhance the notion of conceptual model depending on given scenarios. The system construction scenario integrates description and prescription scenarios beside specification and coding scenarios.

One Main Scenario for Conceptual Database Models Is the Description Scenario. A conceptual model as a descriptive conceptual model is a deliverable of an understandable (may be, ready to apply or to practise) and formalised (or well-formed) [concept-based], unconditionally acceptable conceptualisation of perception and domain-situation models for interaction and discourses.

For database applications it is thus a model suite consisting of a conceptual database model (or schema), of a collections of conceptual views for support of business users, and of a collection of commonly accepted domain-situation models with explicit associations to views (see [37]).

The Second Main Scenario for Conceptual Database Models Is the Prescription Scenario. A conceptual model as a prescriptive conceptual model is a coding supporter as an analysed or synthesised, ready-to-apply blueprint because it can be deployed, it is unconditionally accepted, and appraised in a deliberately and precise practice as a tacit tool which provides notion explanations [from descriptive conceptual models].

For database applications it is thus a model suite consisting of a conceptual database model (or schema), of a collection of views for both support of business users and system operating, and of realisation templates (see [37]).

4.3 Models, Languages, and Ontologies

The major goal of an ontology [20] is to determine what exists and what not. It is independent of humans to conceive it and what kinds of existing things there are. It is independent of perception models although it can be shared among humans. Languages might be textual, visual or audio ones. The classical modelling approach often assumes artificial or partially formal languages.

Languages as Enablers for Conceptual Models: Most models are language based. The language is an instrument similar to models. Moreover, the first models that a human develops are preconscious or subconscious, e.g. the model of a 'mother'. Languages are however enablers since the words in languages can be used for denoting concepts. Many conceptual languages integrate several languages, e.g. ER modelling uses the vocabulary from a domain and a graphical language for schema representation.

Conceptual Models Do Not Have to be Based on an Ontology: The notion of ontology is overloaded similar to the notion of model. Ontologies are considered as shared and commonly agreed vocabularies.

A controlled and thus matured ontology must combine a controlled vocabulary, a thesaurus, a dictionary, and a glossary. There is not real need for associating such ontologies with models.

Languages Are Not Necessary Preconditions for Conceptual Models: Social models are often used for teaching human behaviour. They are based on concepts which might also be not explicit or integrated into the deep model. They are thus conceptual models. We observe however that in most cases conceptual normal models use some language.

5 Conclusion

We developed an approach to conceptual modelling with an explicit integration of concepts into the model. This explicit integration is based on a theory of concepts, conceptions, and conceptualisation. Concepts are developed for our understanding of the world we observe. Therefore, perception and domain-situation models become the origins of our conceptual models.

There Are Models that Are Not Conceptual Models: Sciences and engineering use models without explicit integration of concepts. It is often also difficult to use concepts within the model. A model performs a function in a scenario. Explicit conceptualisation would make the model more complex and thus less useful.

What Is Not Yet a Conceptual Model: Middle-range theories are essentially mediator models. They are used for mediation between qualitative theories (e.g. their conceptualisations) and quantitative observations. For instance, sciences such as archeology make use of modern or medieval concepts without having yet an appropriate concept for prehistoric time, e.g. the concept of settlement or a village. Another typical model that might be enhanced by concepts is the graph model for the Königsberg bridge problem that uses pathes within a graph for solving this problem. The topographical model for the bridge problem uses the concepts of islands and bridges and thus allows to explain the solution.

What Will Never Be a Conceptual Model: Most life situations do not need conscious models since we can live with what we have learned. Preconscious, unconscious, and subconscious models guide life, emotions, and intuitions. Conscious models require efforts and thus must have an explicit need. Concept(ion)s do not to be explicated since there might be no necessity in that.

Remark: More detailed information on our research papers can be found on research gate in collections https://www.researchgate.net/profile/Bernhard_Thalheim.

References

1. CRC 1266. Scales of transformation - Human-environmental interaction in prehistoric and archaic societies. Collaborative Research Centre. http://www.sfb1266.uni-kiel.de/en/. Accessed 13 May 2018
2. Annas, J.: An Introduction to Plato's Republic. Clarendon Press, Oxford (1981)
3. Blackmore, S.: The Meme Machine. Oxford University Press, Oxford (1999)
4. Bosco, S., Braucher, L., Wiechec, M.: Encyclopedia Britannica. Ultimate Reference Suite, Merriam-Webster, USA (2015)
5. Brentano, F.: Psychologie vom empirischen Standpunkte. Dunker & Humblot, Leipzig (1874)
6. Dahanayake, A.: An environment to support flexible information modelling. Ph.D. thesis, Delft University of Technology (1997)
7. Dahanayake, A., Thalheim, B.: Co-evolution of (information) system models. In: Bider, I., et al. (eds.) EMMSAD 2010. LNBIP, vol. 50, pp. 314–326. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13051-9_26
8. Deicher, S.: KunstModell in Ancient Egypt. BMBF Project description, University of Applied Sciences, Wismar (2018)
9. Embley, D., Thalheim, B. (eds.): The Handbook of Conceptual Modeling: Its Usage and Its Challenges. Springer, Berlin (2011)
10. Jaakkola, H., Thalheim, B.: Cultures in information systems development. In: Information Modelling and Knowledge Bases XXX, pp. 61–80. IOS Press, Amsterdam (2019)

11. Kangassalo, M.: Changes in children's conceptual models and the development of children's exploration strategies in the PICCO environment. In: Information Modelling and Knowledge Bases XI. Frontiers in Artificial Intelligence and Applications, vol. 61, pp. 251–255. IOS Press, Amsterdam (2000)
12. Kaschek, R.: Konzeptionelle Modellierung. Ph.D. thesis, University Klagenfurt (2003). Habilitationsschrift
13. Kauppi, R.: Einführung in die Theorie der Begriffssysteme. Acta Universitatis Tamperensis, Ser. A, vol. 15, Tampereen yliopisto, Tampere (1967)
14. Knuth, D.E.: Literate programming. Comput. J. **27**(2), 97–111 (1984)
15. Kropp, Y., Thalheim, Y.: Data mining design and systematic modelling. In: Proceedings of the DAMDID/RCDL'17, Moscov, pp. 349–356 (2017). FRC CSC RAS
16. Lafrance, Y.: Pour interpréter Platon: La Ligne en République VI, 509d–511e. Bilan analytique des études, 1804–1984, vol. 114. Les Editions Fides (1986)
17. Lattmann, C.: Vom Dreieck zu Pyramiden - Mathematische Modellierung bei Platon zwischen Thales und Euklid. Habilitation thesis, Kiel University, Kiel (2017)
18. Lebedev, A.V.: The Logos Heraclitus - A reconstruction of thoughts and words; full commented texts of fragments. Nauka, Moskva (2014). (in Russian)
19. Mahr, B.: Information science and the logic of models. Softw. Syst. Model. **8**(3), 365–383 (2009)
20. Mahr, B.: Intentionality and modeling of conception. In: Bab, S., Robering, K. (eds.) Judgements and Propositions - Logical, Linguistic and Cognitive Issues, pp. 61–87. Logos Verlag, Berlin (2010)
21. Mahr, B.: Modelle und ihre Befragbarkeit - Grundlagen einer allgemeinen Modelltheorie. Erwägen-Wissen-Ethik (EWE) **26**(3), 329–342 (2015)
22. Müller, R.: Model history is culture history. From early man to cyberspace. http://www.muellerscience.com/ENGLISH/model.htm (2016). Assessed 29 October 2017
23. Murphy, G.L.: The Big Book of Concepts. MIT Press, Cambridge (2001)
24. Pastor, O.: Conceptual modeling of life: Beyond the homo sapiens (2016). http://er2016.cs.titech.ac.jp/assets/slides/ER2016-keynote2-slides.pdf. Keynote given at ER'2016 (Nov. 15)
25. Popkov, G.P., Popkov, V.K.: A system of distributed data processing. Vestnik Buryatskogo Gosudarstvennogo Universiteta **9**, 174–181 (2013). (in Russian)
26. Robinson, S., Arbez, G., Birta, L.G., Tolk, A., Wagner, G.: Conceptual modeling: definition, purpose and benefits. In: Proceedings of the 2015 Winter Simulation School, pp. 2812–2826. IEEE (2015)
27. Schewe, K.-D., Thalheim, B.: Semantics in data and knowledge bases. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 1–25. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88594-8_1
28. Stachowiak, H.: Allgemeine Modelltheorie. Springer, New York (1973)
29. Tanaka, Y.: Meme Media and Meme Market Architectures: Knowledge Media for Editing, Distributing, and Managing Intellectual Resources. Wiley, Hoboken (2003)
30. Thalheim, B.: Entity-Relationship Modeling - Foundations of Database Technology. Springer, Berlin (2000)
31. Thalheim, B.: The conceptual framework to user-oriented content management. In: Information Modelling and Knowledge Bases. Frontiers in Artificial Intelligence and Applications, vol. XVIII. IOS Press (2007)
32. Thalheim, B.: The conceptual model $\equiv$ an adequate and dependable artifact enhanced by concepts. In: Information Modelling and Knowledge Bases. Frontiers in Artificial Intelligence and Applications, vol. XXV, 260, pp. 241–254. IOS Press (2014)

33. Thalheim, B.: Conceptual model notions - a matter of controversy; conceptual modelling and its lacunas. EMISA Int. J. Conceptual Model. **13**, 9–27 (2018)
34. Thalheim, B.: Normal models and their modelling matrix. In: Models: Concepts, Theory, Logic, Reasoning, and Semantics, Tributes, pp. 44–72. College Publications, London (2018)
35. Thalheim, B.: Conceptual modeling foundations: the notion of a model in conceptual modeling. In: Encyclopedia of Database Systems. Springer, US (2019)
36. Thalheim, B., Nissen, I. (eds.): Wissenschaft und Kunst der Modellierung: Modelle, Modellieren. Modellierung. De Gruyter, Boston (2015)
37. Thalheim, B., Tropmann-Frick, M.: The conception of the conceptual database model. In: ER 2015. LNCS, vol. 9381, pp. 603–611. Springer, Berlin (2015)
38. White, R.T.: Commentary: conceptual and conceptional change. Learn. Instruction **4**, 117–121 (1994)

Building Formal Semantic Domain Model: An Event-B Based Approach

Idir Ait-Sadoune(✉) and Linda Mohand-Oussaid

LRI - CentraleSupelec - Paris-Saclay University, Plateau de Saclay, France
{idir.aitsadoune,linda.mohandoussaid}@centralesupelec.fr

Abstract. Ontologies are structured data models used to describe a set of concepts related to a specific domain, they describe also the semantic properties of these concepts. Formal development process aims to develop a system with respect to properties or constraints. The IMPEX project is interested in involving domain constraints as soon as possible into formal development process, it proposes to integrate the ontologies descriptions in an Event-B development process. This approach assumes to develop a transformation step of the ontologies constructs from their initial description in an ontological language (OWL, OntoML ...) to Event-B. A first version of this transformation approach for OWL ontologies based on a generic and extensible architecture has been developed, it is supported by the OntoEventB tool. An evolution of this approach to include OntoML ontologies and new features is presented in this paper.

Keywords: Ontology · Formal domain model · Measure data types description · Event-B

1 Introduction

When designing a hardware or a software system, the integration of domain or/and environment constraints becomes a determining factor to ensure a great match with the system requirements. This domain or/and environment knowledge is most often described using knowledge models, named ontologies [9], that allow to express the domain or/and environment properties.

In a system design process, several relevant properties can be checked using the formal methods. These properties are expressed according to the semantics associated to the formal technique being used: type checking, proofs theory, logic based reasoning, rewriting, refinement, model checking, trace analysis, simulation, etc. When considering these properties in their domain or/and environment with the associated semantics, these properties may be no longer respected. As a very simple example, take two formally developed systems that are composed to exchange length measures data represented by a float. This system is no longer consistent if one system refers to Meter unit and the other to Inch unit.

This work was supported by grant from the French national research agency ANR (IMPEX ANR-13-INSE-0001: http://impex.loria.fr).

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 140–155, 2019.
https://doi.org/10.1007/978-3-030-32065-2_10

In the IMPEX project[1], we have proposed an approach to integrate domain or/and environment constraints into a system development process based on Event-B formal method [1]. It consists in annotating Event-B models using the ontology concepts [3–5, 10]. We suggest to integrate domain constraints as a part of design models and we propose to model domain concepts using ontologies to annotate systems and to formalise the obtained design models in Event-B. This assumes a formalisation of the domain ontology by using the Event-B formal method. The formalisation of the domain ontology within Event-B models allows to constrain the system under design with the domain or/and environment properties. Therefore, we have proposed an extensible generic transformation approach which develops an Event-B specification based on an ontology described in an ontological language [14]. A Rodin plug-in [2], named *OntoEventB*, was developed to support an automatic generation of Event-B contexts from ontologies descriptions.

In our previous work [14], we have proposed an approach using an architecture composed of three components: Input component, Pivot component and Output component. The most important component is the Pivot Model which is an extensible independent representation of ontologies which summarizes the common pertinent concepts used by different ontology description languages. For example, we have treated all generic concepts describing Web ontologies that we can find in languages like OWL [17].

In this paper, we present the second version of our approach that extends the proposed architecture to integrate new generic concepts used to describe engineering ontologies that we can find in ontologies description languages like OntoML/Plib [15]. This extension impacts the three components of the proposed architecture in the following way: we have the possibility to process OntoML files as input at the level of the Input component, we have extended the Pivot model of the Pivot component to capture all OntoML concepts that are not formalised in the first version, and we have defined new rules to formalise and to integrate the new OntoML concepts in the generated Event-B model at the Output component level.

This paper is organised as follows: Sect. 2 presents the proposed approach for integrating ontologies descriptions into Event-B formal method, Sect. 3 reminds the *OntoEventB* architecture and Sects. 4, 5, 6 and 7 detail the extensions made to different components of the proposed architecture. Finally, Sect. 8 is devoted to the related works.

2 The Proposed Approach

This paper deals with domain constraints integration into a system development process based on Event-B. During an Event-B development process, the system behaviour is modelled by a set of Machine containing a set of variables whose values change within events. The variables are usually typed by using predefined types or built sets declared in a static component, the Context [1].

[1] http://impex.loria.fr.

Regardless the describing language, ontologies [9] share a set of modelling concepts to describe domain constraints. An ontology describes individuals (instances) grouped into collections called classes (concepts). Classes are characterised by typed attributes and linked to other classes using properties. A class can be built by combining other classes using algebraic operators or by constraining a class using a logical predicate established on a property. Classes can be linked by inheritance relationships. Ontology modelling languages use set theory and predicate logic to describe ontologies, they involve deduction mechanisms to infer new assertions and answer queries about ontology components.

In order to integrate domain or/and environment constraints in Event-B development process, we have proposed to formalise the ontology as a system data model within a Context component [3–5]. Thus, the variables of the Event-B models take their values in ontology concepts and inherit domain or/and environment constraints.

The development of a translation approach emerges as a natural choice for the expression of an ontology description in the Event-B formal method. This approach allows to formalise an ontology described with an ontology language into an Event-B specification. It takes as input an ontology description and generates as output the corresponding Event-B model. The translation approach is based on correspondences between the ontology languages constructs and the Event-B formal semantics. The *OntoEventB* plug-in [14] has been developed to automatically support the translation of ontologies description into Event-B formalisation. It takes as input an ontology description file and generates the corresponding Event-B Context component.

3 The OntoEventB Architecture

The *OntoEventB* plug-in is developed according to an architecture composed of three components: Input Models, Pivot Model and Output Models (Fig. 1).

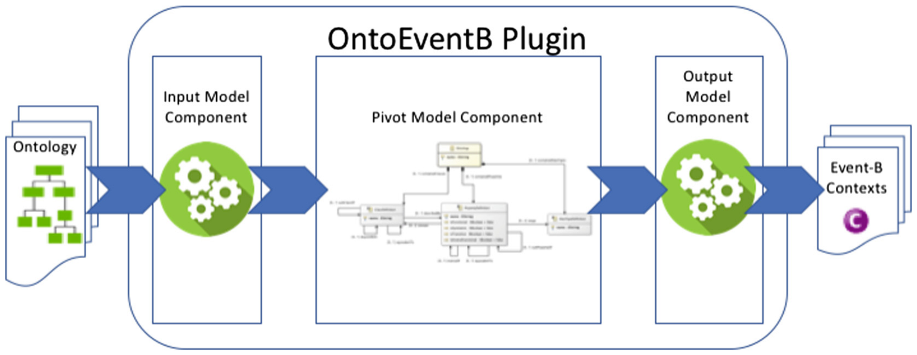


Fig. 1. The OntoEventB architecture.

Input Models Component. This component is devoted to treat ontology files provided as input. It browses the received files as input models in order to extract ontological concepts descriptions (classes, properties, data types, ...) intended to be processed by the Pivot Model component.

Pivot Models Component. This component contains an intermediate model which summarises common pertinent concepts used by a great number of ontology description languages. It defines generic concepts that integrate all specific concepts that can be received from the Input Model component. Following the receipt of different ontological concepts derived from the Input Model component, the Pivot Model component translates them into its generic concepts. After this first translation step, the obtained generic concepts are ready to be treated by the next process handled by the Output Model component.

Output Models Component. This component receives generic concepts computed by the Pivot Model component, and translates them into Event-B Context elements (sets, constants and axioms). This process uses translation rules, proposed in [14], that formalise each ontological concept by an Event-B definition.

The use of this architecture allows us to extend the *OntoEventB* approach by processing new input ontology description languages without updating all defined Event-B formalisation of the Pivot Model concepts. If the added language contains new concepts that are not yet defined in the Pivot Model, the Pivot model will be extended to integrate this new concepts. Indeed, as soon as the new concepts defined by these new languages are added into the Pivot model, they will be directly formalised with Event-B definitions without redefining all transformation rules.

Thus, in this article, we propose to extend the proposed architecture to take into account input ontologies described by OntoML/Plib language [15]. This extension constitute an update into the OntonEventB architecture by adding new ontological concepts into the Pivot Model, and by defining new Event-B formalisation rules for the added concepts. These extensions concern the processing of Currency and Measure data type definitions. They are detailed in the following sections.

4 The Input Model Component Extension

The Input Models component of the first version of the OntoEventB approach was able to treat Web ontologies described using OWL language. In the new version of the OntoEventB architecture, the Input Model component is updated to process Plib ontologies [15] described with OntoML representation [16]. The Input Model component browses the OntoML files, extracts concepts descriptions (classes, properties, data types, ...) and sends them to the Pivot Model component (cf. Fig. 1).

The Listing 1.1 presents an extract of an OntoML file describing the ISO23768 Plib ontology/norm[2]. This Ontology specifies a reference dictionary for all concepts described in the various International Standards relevant to rolling bearings, together with their descriptive properties and domains of values. The Listing 1.1 defines a class named *ball bearing* as subclass of another class named *rolling bearing*. The *ball bearing* class contains a set of properties, one of them is named *material treatment*.

Listing 1.1. An extract of an OntoML/Plib description of the ISO23768 norm.

```
<dictionary type="DICTIONARY_IN_STANDARD_FORMAT_Type">
<contained_classes>
  <class type="ITEM_CLASS_Type"
         id="0112-1---23768__1#01-23768AAA028#001">
   ...
  <preferred_name>ball bearing</preferred_name>
  <its_superclass class_ref="0112-1---23768__1#01-23768AAA007#001"/>
  <described_by>
    <property property_ref="0112-1---23768__1#02-23768BAA018#001"/>
    <property property_ref="0112-1---23768__1#02-23768BAA023#001"/>
    <property property_ref="0112-1---23768__1#02-23768BAA001#001"/>
  </described_by>
   ...
  </class>

  <class type="ITEM_CLASS_Type"
         id="0112-1---23768__1#01-23768AAA007#001">
   ...
  <preferred_name>rolling bearing</preferred_name>
   ...
  </class>
   ...
</contained_classes>
<contained_properties>
  <property type="NON_DEPENDENT_P_DET_Type"
            id="0112-1---23768__1#02-23768BAA018#001">
   ...
  <preferred_name>material treatment</preferred_name>
   ...
  <domain type="NON_QUANTITATIVE_CODE_TYPE_Type">
     ...
    <its_values>
    <dic_value type="STRING_DIC_VALUE_Type">
      <preferred_name>bearing steel</preferred_name>
    </dic_value>
            ...
    </its_values>
  </domain>
  </property>
     ...
</contained_properties>
</dictionary>
```

5 The Pivot Model Component Extension

Following the receipt of different ontological concepts obtained from the Input Model component, the Pivot Model component interprets them as one of its concepts. The Pivot Model component contains a model defining the common

[2] Reference dictionary for rolling bearings https://www.iso.org/standard/41845.html.

concepts used by a great number of ontology description languages. The Fig. 2 presents an UML model formalising the mains concepts that are defined in the proposed Pivot Model component. An *Ontology* is defined by a set of classes definitions, a set of properties definitions, and a set of data types definitions. The detail of the Pivot Model was defined and can be found in our previous contribution [14].

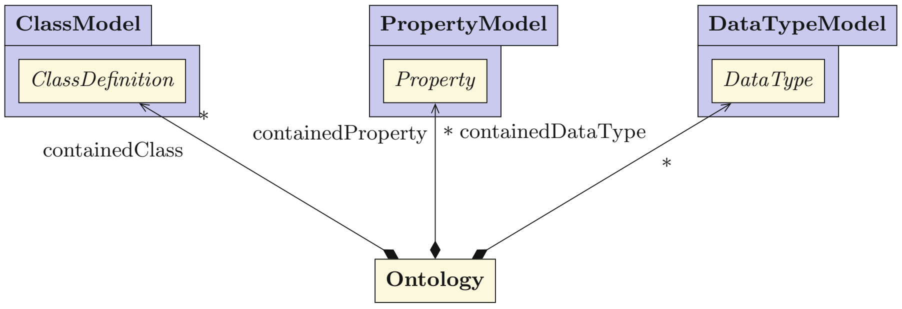


Fig. 2. The main concepts of the Pivot Model.

As indicated in Sect. 3, the Pivot Model can be extended to integrate new generic concepts that can be identified if new ontology description languages are traited by the Input Model component. Therefore, in the new version of the OntoEventB approach, the Pivot Model is extended to handle Plib concepts that are not yet included, essentially Currency and Measure data types description.

The *NumercType* concept, that is a case of a *DataType* definition on the Pivot Model (Fig. 2), is used to formalise the currency and the measure data types. In concrete terms, the *NumercType* class of *Data Type Model* part is extended by two classes: *CurrencyType* class and *MeasureType* class (Fig. 3).

1. The *CurrencyType* class describes the numeric currency types representation. It contains a *currency* attribute that gives the associated code of the described currency expressed according ISO 4217[3] (could be "CHF" for Swiss Francs, "CNY" for Yuan Renminbi (Chinese), "JPY" for Yen (Japanese), "SUR" for SU Rouble, "USD" for US Dollars, "EUR" for Euros etc ...).
2. The *MeasureType* class describes the numeric measure types representation. It contains an association with a *UnitType* class that gives the reference unit associated to the described measure. The *UnitType* class defines two types of units: *NamedUnit* (Millimeter or Pascal are kinds of named unit) and *DerivedUnit* (Newton per square millimeter is a derived unit).
   - The *NamedUnit* class is a unit associated with the word, or group of words, by which the unit is identified. It can represent:

[3] ISO 4217 is the International Standard for currency codes: https://www.iso.org/iso-4217-currency-codes.html.

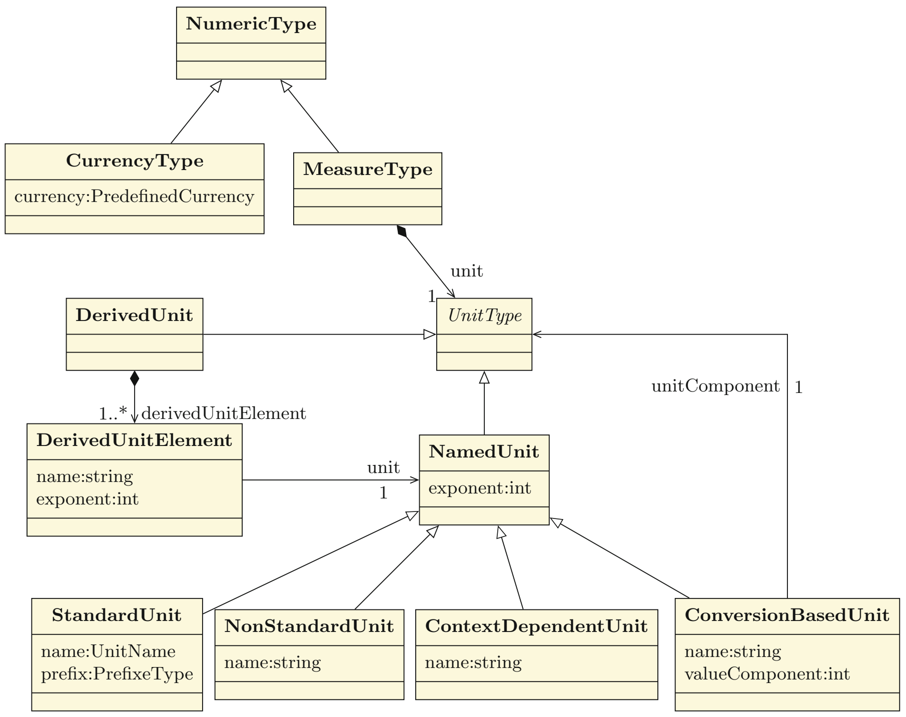


Fig. 3. The NumerciType class definition of the Data Type Model part.

- the fixed quantity used as a standard in terms of which items are measured as defined by ISO 1000[4] (Millimeter is a internationally standardized unit) (*StandardUnit* class),
- units that are not standard units, nor conversion based units, nor length units (*NonStandardUnit* class),
- a unit defined on the base of another unit (inch is a conversion based unit) (*ConversionBasedUnit* class),
- or a unit which is not related to the standard system (The number of parts in an assembly is a physical quantity measured in units that may be called "parts") (*ContextDependentUnit* class).

– The *DerivedUnit* class is a unit that stands for an expression of units. It contains a set of *DerivedUnitElement* which makes up a derived unit (Newton per square millimeter is a derived unit. It would then be represented by two *DerivedUnitElement*: the former for representing the Newton unit, the latter for representing the millimeter unit).

[4] ISO 1000 describes the International System of Units SI: https://www.iso.org/standard/5448.html.

Finally, after translating different ontological concepts obtained from the Input Model component into the Pivot Model component, the next process handled by the Output Model component is ready to treat all the obtained generic concepts.

6 The Output Model Component Extension

The main function of the Output Model component is to formalise using Event-B formal method (contexts, sets, constants and axioms) the generic concepts defined within the Pivot Model component. This process uses transformation rules, proposed in [14], that formalise each ontological concept by an Event-B definition.

The novelty within the new Output Model component is the possibility of generating a PM file (Pivot Model textual representation) that contains all Pivot Model concepts. Moreover, as showed in the previous section, the Pivot Model was extended to handle Plib/OntoML concepts that are not already defined in the first version of the OntoEventB approach, essentially the currency and the measure data types description. The Event-B formalisation of these parts is also one of the main updates of the Output Model component. All these extensions are detailed in the following sections.

6.1 Generating a Pivot Model (PM) File

As announced in the previous paragraph, the Output Model component offers the possibility to generate a PM file that contains the Pivot Model definitions. The purpose of this feature is to provide a better and more readable textual representation of ontologies than XML used by OWL and OntoML languages, and to provide the ability to extend and supplement OWL or OntoML ontologies with new definitions not intended by these languages. For example, we can add measurements definitions to an OWL ontology or define a new class as a union of two OntoML classes.

Listing 1.2. The PM extract of the OntoML ISO23768 norm.

```
Ontology ISO23768{
    containedClasses{
        SimpleClass BallBearing{
            subClassOf(RollingBearing)
            describedBy(materialTreatment ,...)
        },
        SimpleClass RollingBearing{
            describedBy (...)
        },
        ...
    }
    containedProperties{
        SimpleProperty materialTreatment{
            range String
        },
        ...
    }
}
```

We specify that the obtained PM file can be reloaded by the Input Model component to be used by the Pivot Model and the Output Model components in order to generate an Event-B formalisation corresponding to the PM file content. The Listing 1.2 shows an extract of the PM content obtained from the OntoML description of the ISO23768 norm given in Listing 1.1. This extract contains a class named *BallBearing* that is a subclass of another class named *RollingBearing*. The *BallBearing* class is described by a set of properties, one of them is named *materialTreatment*.

6.2 Event-B Modelling of Currency and Measure Data Types

The second main updates of the Output Model component is the definition of an Event-B formalisation of some Plib/OntoML concepts that are not already defined in the first version of the OntoEventB approach, essentially the currency and the measure data types definitions.

Due to the limitation of the pages number, we present only the Event-B formalisation of the *StandardUnit* class, describing measurements standard units, defined within the Pivot Model (Fig. 3). The same principle is used to model all the other concepts defined by the *UnitType* class of the Pivot Model (Fig. 3).

Listing 1.3. Event-B formalisation of the measurements standard units.

```
SETS
    PredefinedPrefix  PredefinedStandardUnit
CONSTANTS
    KILO NONE MILLI ... METRE GRAM SECOND ... PredefinedUnit
AXIOMS
    @axm1 : partition(PredefinedPrefix, {KILO}, {NONE}, {MILLI}, ...)
    @axm2 : partition(PredefinedStandardUnit, {METRE}, {GRAM}, {SECOND}, ...)
    @axm3 : PredefinedUnit = PredefinedPrefix × PredefinedStandardUnit
```

To formalise measurements standard units with Event-B definitions, we have taken over the ISO 1000 specification. More precisely, we have defined two enumerated sets: *PredefinedPrefix* and *PredefinedStandardUnit* (cf. Listing 1.3). The *PredefinedPrefix* set contains all prefixes that may be associated with an internationally standardised unit (axiom *axm*1), and the *PredefinedStandardUnit* contains all names of internationally standardised units (axiom *axm*2). Finally, an internationally standardised unit is described through a prefix and a unit name. It is defined by the *PredefinedUnit* set (axiom *axm*3).

Listing 1.4. Event-B Dimensional definition of a measurements standard units

```
SETS
    Dimension
CONSTANTS
    LENGTH MASS TIME dimensionOfUnit
AXIOMS
    @axm4 : partition(Dimension, {LENGTH}, {MASS}, {TIME}, ...)
    @axm5 : dimensionOfUnit ∈ PredefinedStandardUnit → Dimension
    @axm6 : dimensionOfUnit = {METRE ↦ LENGTH, GRAM ↦ MASS,
                    SECOND ↦ TIME, ...}
```

Each measurements standard units is associated to its dimension. A dimension is defined and formalised by the *Dimension* set containing an enumeration of the most used dimensions (*axm*4 of Listing 1.4). The association between the standard unit and its dimension is defined by the *dimensionOfUnit* function (axioms *axm*5 and *axm*6 of Listing 1.4).

The main concept of the proposed approach is the definition of a numerical measure type that is formalised by the *measurements* set (axiom *axm*7 of Listing 1.5). This type associates (or annotates) an integer value with its dimension and measurement unit definitions. Moreover, we have defined four getter functions that are associated with the *measurements* type to get all information for a measurement: *getValue*, *getPrefix*, *getUnit* and *getDimension* (axioms *axm*8 to *axm*15 of Listing 1.5).

Listing 1.5. Event-B numerical measure type definition

```
CONSTANTS
      measurements  getValue  getPrefix  getUnit  getDimension
AXIOMS
      @axm7 : measurements = ℤ × (Dimension × PredefinedUnit)
      @axm8 : getValue ∈ measurements → ℤ
      @axm9 : ∀v, u·(v ↦ u ∈ measurements) ⇒ getValue(v ↦ u) = v
      @axm10 : getDimension ∈ measurements → Dimension
      @axm11 : ∀v, d, u·(v ↦ (d ↦ u) ∈ measurements) ⇒ getDimension(v ↦ (d ↦ u)) = d
      @axm12 : getPrefix ∈ measurements → PredefinedPrefix
      @axm13 : ∀v, d, p, u·(v ↦ (d ↦ (p ↦ u)) ∈ measurements)
                        ⇒getPrefix(v ↦ (d ↦ (p ↦ u))) = p
      @axm14 : getUnit ∈ measurements → PredefinedStandardUnit
      @axm15 : ∀v, d, p, u·(v ↦ (d ↦ (p ↦ u)) ∈ measurements)
                        ⇒getUnit(v ↦ (d ↦ (p ↦ u))) = u
```

To have a possibility to write expressions/actions that use integer variables or constants annotated as measure, we must override all basic operators/functions that interact with the numerical types, essentially the arithmetic operators (addition, subtraction, multiplication, division, ...). This part of the generated context is not produced from the ontology.

Listing 1.6. Event-B definition of the addition operator

```
CONSTANTS
      addMeasure
AXIOMS
      @axm16 : addMeasure ∈ (measurements × measurements) ⇸ measurements
      @axm17 : ∀m1, m2·(m1 ∈ measurements ∧ m2 ∈ measurements ∧
                        getDimension(m1) = getDimension(m2) ∧
                        getPrefix(m1) = getPrefix(m2) ∧
                        getUnit(m1) = getUnit(m2))
                        ⇔
                        m1 ↦ m2 ∈ dom(addMeasure)
      @axm18 : ∀m1, m2·(m1 ↦ m2 ∈ dom(addMeasure))
                        ⇒ getValue(addMeasure(m1 ↦ m2)) = getValue(m1) + getValue(m2)
```

The Listing 1.6 shows the Event-B definition of the addition operation calculating the sum of two measures (*addMeasure* function defined by the *axm*16 axiom). The axiom *axm*17 gives the well-definedness condition of the *addMeasure* operator and the axiom *axm*18 gives the result computed by this operator. The same formalisation is done to define the others arithmetic operators like subtraction, multiplication and division.

In addition to the proposed measurement type definition, we have provided the possibility to convert a measured quantity to a different unit of measure without changing the relative amount. To accomplish this, we have defined an Event-B function called *convert* (*axm*19 axiom on the Listing 1.7).

Listing 1.7. Event-B definition of the conversion function

```
CONSTANTS
    convert
AXIOMS
    @axm19 : convert ∈ (measurements × (PredefinedUnit)) ⇸ measurements
    @axm20 : ∀m, p, u·(m ∈ measurements ∧ (p ↦ u) ∈ PredefinedUnit ∧
                 getDimension(m) = dimensionOfUnit(u))
                 ⇒ (m ↦ (p ↦ u)) ∈ dom(convert)
    @axm21 : ∀m, p, u·((m ↦ (p ↦ u)) ∈ dom(convert)) ⇒
                 getPrefix(convert(m ↦ (p ↦ u))) = p ∧
                 getUnit(convert(m ↦ (p ↦ u))) = u ∧
                 getDimension(convert(m ↦ (p ↦ u))) = getDimension(m)
```

In addition, this definition is strengthened by two axioms: the *axm*20 axiom gives the well-definedness conditions of the *convert* function, and the *axm*21 axiom states that the obtained measure has the same dimension as the converted measure and it is annotated with the measurement unit given as input of the convert function.

6.3 Example of Using Event-B Definition of Measurements Units

As a very simple example, we take two systems that are composed to exchange length measures data represented by integer. The Listing 1.8 shows the Event-B model applying the measurements formalisation proposed by our approach. It contains three integer variables named *length_1*, *length_2* and *length_3* annotated with the *LENGTH* dimension. The *length_1* and *length_2* variables are expressed with the *MILLI METRE* unit, and the *length_3* variable is expressed with the *KILO METRE* unit (*inv*1, *inv*2 and *inv*3 invariants).

Listing 1.8. Event-B model of systems composed to exchange length measures

```
VARIABLES
    length_1 length_2 length_3 value_1 value_2
INVARIANTS
    @inv1 : length_1 ∈ ℤ × ({LENGTH} × ({MILLI} × {METRE}))
    @inv2 : length_2 ∈ ℤ × ({LENGTH} × ({MILLI} × {METRE}))
    @inv3 : length_3 ∈ ℤ × ({LENGTH} × ({KILO} × {METRE}))
    @inv4 : value_1 ∈ ℤ
    @inv5 : value_2 ∈ ℤ
EVENTS
    correct_composition ≙
        Begin
        @act : value_1 := getValue(addMeasure(length_1 ↦ length_2))
        End
    wrong_composition ≙
        Begin
        @act : value_2 := getValue(addMeasure(length_1 ↦ length_3))
        End
```

The Listing 1.8 contains two events: the *correct_composition* event using the *addMeasure* operator to compose (by addition) two length measures contained into the *length_1* and the *length_2* variables, and the *wrong_composition* event using the same operator to compose the *length_1* variable with the length contained into the *length_3* variable.

Listing 1.9. The PO of the *addMeasure* operator used by the *wrong_composition* event

```
$length\_1 \in \mathbb{Z} \times (\{LENGTH\} \times (\{MILLI\} \times \{METRE\}))$
$length\_3 \in \mathbb{Z} \times (\{LENGTH\} \times (\{KILO\} \times \{METRE\}))$
$\forall m1, m2 \cdot (m1 \in measurements \land m2 \in measurements \land$
                              $getDimension(m1) = getDimension(m2) \land$
                              $getPrefix(m1) = getPrefix(m2) \land$
                              $getUnit(m1) = getUnit(m2))$
                              $\Leftrightarrow m1 \mapsto m2 \in dom(addMeasure)$
$\Rightarrow length\_1 \mapsto length\_3 \in dom(addMeasure)$
```

By launching the *AtelierB* prover of the Rodin platform [2], we obtain a set of proof obligations (PO) that are proved except the well-definedness proof obligation (WD PO) associated to the *addMeasure* operator used by the *wrong_composition* event (Listing 1.9). This means that the *axm*17 axiom of the Listing 1.6 is not established by the *wrong_composition* event. This system is not consistent because we compose one variable that refers to the Milli-Meter unit and other to the Kilo-Meter unit.

Listing 1.10. The correct version of composing two measures with different units.

```
corrected_wrong_composition ≙
Begin
@a : $value\_2 := getValue(addMeasure(length\_1 \mapsto convert(length\_3 \mapsto (MILLI \mapsto METRE))))$
End
```

On the other hand, we give on the Listing 1.10 how we can compose the two length measures by converting the *length_3* variable into the same measure unit as the *length_1* variable by using *convert* function defined on the Listing 1.7. After this correction, the defined system becomes consistent.

7 The OntoEventB Plug-In

The *OntoEventB* plugin implements the proposed approach (Fig. 1) and it has been developed to automatically support the formalisation of ontologies using the Event-B method. The *OntoEventB* tool is developed as an Eclipse plug-in to be integrated into the Rodin platform [2] which is an Eclipse IDE (Integrated Development Environment) supporting the Event-B developments.

Unlike the first version of the plugin [14] (it proposes only the **OWL** $\mapsto$ **Event-B** feature), this second version proposes five features:

1. the **OntoML** $\mapsto$ **Event-B** feature that generates an Event-B context from an OntoML ontology,
2. the **OntoML** $\mapsto$ **PM** feature that generates Pivot Model textual file from an OntoML ontology,
3. the **OWL** $\mapsto$ **Event-B** feature that generates an Event-B context from an OWL ontology,
4. the **OWL** $\mapsto$ **PM** feature that generates Pivot Model textual file from an OWL ontology, and
5. the **PM** $\mapsto$ **Event-B** feature that generates an Event-B context from a Pivot Model file.

7.1 Installing the OntoEventB Plug-In

To use *OntoEventB* plug-in in your Rodin platform instance, you must, first install xText plugin[5], and then OntoEventB plugin[6]. After installing these plugins in your Rodin platform instance, the *OntoEventB* sub-menu becomes available by right clicking on a file with a *.owl, .xml, or .pm* extensions in the project explorer.

7.2 Example of Using the OntoEventB Plug-In

The presented approach was applied to the ISO23768 norm. The Listing 1.11 presents an OntoML extract of this norm containing an example of a property definition named *outside diameter*. The *outside diameter* property has as domain the *Millimeter Standard Unit Type*.

Listing 1.11. The OntoML definition of the *outside diameter* property.

```
<property ...>
    <preferred_name>outside diameter</preferred_name>
    <domain type="REAL_MEASURE_TYPE_Type">
        <unit>
            <structured_representation type="SI_UNIT_Type">
                <prefix>MILLI</prefix><name>METRE</name>
            </structured_representation>
        </unit>
    </domain>
</property>
```

By using the *OntoML* $\mapsto$ *PM* feature of the *OntoEventB* plugin, we obtain the Listing list12 containing an example of a property definition with the case of measure data type definition. This property is named *pOutsideDiameter* and it has as range the *millimeter unit* type (*MILLI_METRE_Measure* data type). This definition corresponds to the Pivot Model formalisation of the MeasureType proposed in the Sect. 2.

By using the *PM* $\mapsto$ *Event-B* or the *OntoML* $\mapsto$ *Event-B* feature, and the Event-B formalisation of currency and measure data type defined in the Listings 1.3, 1.4 and 1.5, the *pOutsideDiameter* property is defined on the Listing 1.13 by the *axm*11 axiom and its range the *MILLI_METRE_Measure* data type is defined by the *axm*8 axiom.

Listing 1.12. The PM definition of the *outside diameter* property.

```
containedProperties{
    ...
    SimpleProperty pOutsideDiameter {
        range MILLI_METRE_Measure
    }
}
containedDataTypes {
    ...
    MeasureType MILLI_METRE_Measure {
```

[5] xText update site: http://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/.
[6] OntoEventB update site: http://wdi.supelec.fr/OntoEventB-update-site/.

```
        basedOn INTEGER
        unit StandardUnit {
            prefix MILLI
            name METRE
        }
    }
}
```

Listing 1.13. The Event-B definition of *pOutsideDiameter* property

```
CONSTANTS
    ... MILLI_METRE_Measure ... pOutsideDiameter ...
AXIOMS
    @...
    @axm8 $MILLI\_METRE\_Measure \subseteq (\mathbb{Z} \times \{LENGTH \mapsto \{MILLI \mapsto METRE\}\})$
    @...
    @axm11 $pOutsideDiameter \in Thing \leftrightarrow MILLI\_METRE\_Measure$
```

8 Related Work

Integration of domain constraints into design processes has attracted lately great interest in the software engineering community. Many proposed approaches uses ontology descriptions as design models for these domain constraints. We focus in this overview on integration approaches devoted to formal development processes. In [18], an ontology axioms transformation into Z notation is proposed to express application domain rules. In [12], an integration of domain constraints is proposed for cyber-physical systems, it is performed by interpretation of computing platform components on real-world types to derive properties specification and validation. In [8], authors propose to define real-world systems semantics using domain ontologies. A domain specific description of the system is coupled to a domain ontology, the two are formalized into logic theories and conformity validation is conducted using the Alloy formal method. In [13], an Event-B specification of an OWL domain ontology is integrated to goal-based model during requirements engineering phase, this approach aims to construct a data structure for typing. In [11], domain knowledge is integrated to a development process by annotation using two approaches: an MDE approach mixing ontology and OCL constraints and a theorem proving approach based on Event-B specifications. The proposed Event-B modelling approach for ontologies is based on an own definition of ontological constructs in an ontology model context combined to instantiation mechanisms. In [6], a derivation approach is proposed to generate Event-B models from OWL ontologies through the ACE controlled natural language.

In comparison with the approaches cited above, our approach is distinguished, on the one hand, by its genericity assured by the Pivot Model which federates several ontological description languages (OWL and OntoML/Plib) and ensures the integration of new languages, on the other hand, by the broad coverage of the ontological languages primitives allowing a total ontology support. Finally, our approach is tooled in order to automatically support the transformation of ontologies in Event-B.

9 Conclusion

Our results show that it is possible to handle formally domain knowledge in formal system developments with Event-B and the Rodin platform. Ontologies have been formalised within Event-B as contexts and a Rodin plugin has been developed for this purpose. The proposed approach consists in defining models allowing to handle formal verification techniques and make it possible to handle explicit domain knowledge in such formal models.

The presented approach gave rise to several extensions. We are currently working on:

1. Extending the Pivot Model for providing the possibility to express properties by using predicates in all supported concepts (classes, properties, data defintions, ...). This point is usually omitted by the languages used for describing ontologies. This extension can be exploited to generate automatically invariants properties expressed at the ontological level.
2. Extending the OntoEventB approach to emphasis other formal methods at the level of the Output Model. Actually we are developing an Isabelle/HOL framework to support the integration of ontologies [7].

References

1. Abrial, J.: Modeling in Event-B - System and Software Engineering. Cambridge University Press, Cambridge (2010)
2. Abrial, J., Butler, M.J., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. STTT **12**(6), 447–466 (2010). https://doi.org/10.1007/s10009-010-0145-y
3. Aït-Ameur, Y., et al.: On the importance of explicit domain modelling in refinement-based modelling design. Experiments with Event-B. In: 6th International Conference, ABZ 2018, Southampton, UK, 5–8 June 2018, Proceedings, pp. 425–430 (2018). https://doi.org/10.1007/978-3-319-91271-4_35
4. Aït-Ameur, Y., Aït-Sadoune, I., Hacid, K., Mohand-Oussaïd, L.: Formal modelling of ontologies: an Event-B based approach using the rodin platform. In: Proceedings Joint Workshop IMPEX and FM&MDD, Xi'An, China, 16th November 2017, pp. 24–33 (2017). https://doi.org/10.4204/EPTCS.271.2
5. Aït-Ameur, Y., Méry, D.: Making explicit domain knowledge in formal system development. Sci. Comput. Program. **121**, 100–127 (2016). https://doi.org/10.1016/j.scico.2015.12.004
6. Alkhammash, E.H.: Derivation of event-b models from owl ontologies. In: MATEC Web of Conferences, vol. 76, p. 04008. EDP Sciences (2016)
7. Brucker, A.D., Aït-Sadoune, I., Crisafulli, P., Wolff, B.: Using the isabelle ontology framework - linking the formal with the informal. In: 11th International Conference, CICM 2018, Hagenberg, Austria, 13–17 August 2018, Proceedings, pp. 23–38 (2018). https://doi.org/10.1007/978-3-319-96812-4_3
8. de Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using reference domain ontologies to define the real-world semantics of domain-specific languages. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 488–502. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_33

9. Gruber, T.: Ontology. In: Encyclopedia of Database Systems, 2nd edn. (2018). https://doi.org/10.1007/978-1-4614-8265-9_1318
10. Hacid, K., Aït-Ameur, Y.: Annotation of engineering models by references to domain ontologies. In: 6th International Conference, MEDI 2016, Almería, Spain, 21–23 September 2016, Proceedings, pp. 234–244 (2016). https://doi.org/10.1007/978-3-319-45547-1_19
11. Hacid, K., Aït-Ameur, Y.: Strengthening MDE and formal design models by references to domain ontologies. A model annotation based approach. In: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, 10–14 October 2016, Proceedings, Part I, pp. 340–357 (2016). https://doi.org/10.1007/978-3-319-47166-2_24
12. Knight, J., Xiang, J., Sullivan, K.: A rigorous definition of cyber-physical systems. In: Trustworthy Cyber-Physical Systems Engineering, p. 47 (2016)
13. Mammar, A., Laleau, R.: On the use of domain and system knowledge modeling in goal-based event-B specifications. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9952, pp. 325–339. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47166-2_23
14. Mohand-Oussaïd, L., Aït-Sadoune, I.: Formal modelling of domain constraints in Event-B. In: 7th International Conference, MEDI 2017, Barcelona, Spain, 4–6 October 2017, Proceedings, pp. 153–166 (2017). https://doi.org/10.1007/978-3-319-66854-3_12
15. Pierra, G.: Context-explication in conceptual ontologies: the PLIB approach. In: Proceedings of the 10th ISPE International Conference on Concurrent Engineering (ISPE CE 2003), Madeira, Portugal, 26–30 July 2003, pp. 243–253 (2003)
16. Pierra, G., Sardet, E.: Proposal for an XML representation of the PLIB ontology model: OntoML. Technical report, LIAS Laboratory (2007)
17. Sengupta, K., Hitzler, P.: Web ontology language (OWL). In: Alhajj, R., Rokne, J. (eds.) Encyclopedia of Social Network Analysis and Mining, pp. 2374–2378. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-6170-8_113
18. Vasilecas, O., Kalibatiene, D., Guizzardi, G.: Towards a formal method for the transformation of ontology axioms to application domain rules. Information Technology and Control **38**(4) (2015)

Social-Based Collaborative Recommendation: Bees Swarm Optimization Based Clustering Approach

Lamia Berkani

Laboratory for Research in Artificial Intelligence (LRIA), Department of Computer Science, USTHB University, 16111 Bab Ezzouar, Algiers, Algeria
lberkani@usthb.dz

Abstract. This paper focuses on the recommendation of items in social networks, through which the social information is formalized and combined with the collaborative filtering algorithm using an optimized clustering method. In this approach, users are clustered from the views of both user similarity and trust relationships. A Bees Swarm optimization algorithm is designed to optimize the clustering process and therefore recommend the most appropriate items to a given user. Extensive experiments have been conducted, using the well-known Epinions dataset, to demonstrate the effectiveness of the proposed approach compared to the traditional recommendation algorithms.

Keywords: Social recommendation · Collaborative filtering · Clustering · Optimization · Bees Swarm Optimization algorithm · BSO

1 Introduction

Collaborative Filtering (CF) is a widely-exploited technique in recommender systems to provide users with items that well suit their preferences [1]. The basic idea is that a prediction for a given item can be generated by aggregating the ratings of users with similar interest.

However, although the popularity of the CF, the time-consuming process of searching for similar users is considered as a big challenge when facing large-scale data sets, which characterizes the Web 2.0 and the social media contexts in particular. Moreover, one of the shortcomings of memory-based methods is the sparsity and cold start problems due to insufficient rating [2]. In contrast, model-based methods can address these issues by training a prediction model offline using all the rating data.

In order to boost the CF algorithm, other information is often combined with it, such as the social information in the context of social networks. The study of social network-based recommender systems has become an active research topic. On the other hand, recommender systems using clustering-based approaches offer an alternative to model-based methods [3]. Instead of decomposing the rating matrix into matrices with small ranks, these approaches reduce the search space by clustering similar users or items together. Most previous works focus on clustering users or items from the view of similarity [4]. However, the state of the art shows that these approaches suffer from low accuracy and coverage. To address these issues, Guo et al. [4] developed a multi-view clustering method through which users are iteratively clustered from the views of both rating patterns (user similarity) and social trust relationships (social similarity).

We focus in this paper by this same issue for the social recommendation of items. The proposed algorithm considered an optimized clustering, where users are clustered from the views of both user similarity and trust relationships. A Bees Swarm optimization algorithm is designed to optimize the clustering and therefore recommend the most appropriate items to a given user.

The remainder of this paper is organized as follows: Sect. 2 presents some related work on trust-based and clustering-based recommender systems. Section 3 proposes a novel social recommender system of items using a Bees Swarm optimization-based clustering approach (BSO-C). Then in Sect. 4, experiments are conducted using the well-known Epinions[1] dataset. Finally, Sect. 5 highlights the most important results and draws some future perspectives.

2 Related Work

2.1 Trust-Based Recommendation

Trust-based recommendations can improve the performance of traditional recommender systems as people trusting each other usually share similar preferences [5]. Trust information has been widely used as an additional dimension to support model user preference in recommender systems. It has been combined in both types of recommendation: (1) in memory-based methods [6, 7]; and (2) in model-based methods [8, 9].

Due to data sparsity of the input ratings matrix, Massa and Avesani [10] replaced the step of finding similar users with the use of a trust metric. They proposed an algorithm that propagate trust over the trust network and estimate a trust weight that can replace the similarity weight. Guo et al. [11] incorporated trusted neighbors into the CF, by merging the ratings of trusted neighbors. The objective is to form a more complete rating profile for the active users in order to resolve the cold start and data sparsity problems. On the other hand, Ma et al. [12] developed two methods: a matrix factorization framework with social regularization and a factor analysis approach based on probabilistic matrix factorization that exploits users' social network. Based on the two methods proposed in [12], Wang and Huang [13] included the friendships as the regularization term to enhance the prediction accuracy.

2.2 Clustering-Based Recommendation

Clustering-based approaches are being demonstrated to be efficient and scalable to large-scale data sets. As a dimension-reduction method, they are capable of alleviating the sparsity of rating data [14]. Recent works report that by applying more advanced clustering method, the accuracy can be further improved and even outperform the other

[1] Epinions.com.

CF approaches [15]. Salah et al. [16] proposed a novel incremental collaborative filtering system, based on a weighted clustering approach, to provide a high quality of recommendations with a very low computation cost.

Few works have tried to integrate social relationships into clustering-based methods with the aim of better performance of CF. Sun et al. [17] proposed a social regularization approach that incorporates social network information, namely the users' friendships and rating records (tags) for the prediction of the recommendations. They used a bi-clustering algorithm to identify the most suitable group of friends for generating different final recommendations. DuBois et al. [18] combined a correlation clustering algorithm and trust models together to derive trust from the connection distance in a trust network. However, only limited improvement is observed.

According to Guo et al. [4] previous clustering-based approaches suffer from relatively low accuracy and, especially, coverage. To alleviate these issues, they developed a multi-view clustering method through which users are iteratively clustered on the basis of rating patterns in one view and social trust relationships in the other [4]. Sheugh and Alizadeh [19] proposed a multi-view clustering based on Euclidean distance, merging similarity and trust relationships including explicit and implicit trusts. He et al. [20] proposed a web items recommendation system based on a multi-content clustering CF model. According to the authors, a multi-view clustering can be applied to the mining of the similarity and relevance of items to improve the classic CF. Different views such as user ratings and user comments have been considered and users' preferences were analyzed by their historical interaction features and additional behaviour features for an appropriate recommendation.

On the other hand, Selvi and Sivasankar [21] considered a model-based CF based on a fuzzy c-means clustering approach. Because model-based CF suffers by higher error rate and takes more iterations for convergence, the authors proposed a modified cuckoo search algorithm to optimize the data points in each cluster in order to provide an effective recommendation.

2.3 Discussion

Our study of related work confirmed the contribution of integrating social information with CF to enhance the recommendation accuracy. On the other hand, the multi-view clustering has been adopted in some works, as clustering-based approaches suffer from relatively low accuracy and, especially, coverage. Finally, to our best knowledge no work has yet applied multi-view clustering with optimization. Indeed, it is not always easy to have the best partitioning for a given clustering. Therefore, we are interested by this issue and we try to solve it by proposing a new approach, based on optimized clustering. In the following section we present our clustering solution based on the use of Bees Swarm Optimization algorithm.

3 BSOC–Based Recommendation

Several algorithms have been implemented for CF and social filtering (SocF): (1) Cf/SocF; (2) CF/SocF based on clustering algorithms; and (3) CF/SocF based on optimized clustering algorithm. Moreover, some hybridizations have been proposed.

3.1 Collaborative and Social Recommendation Algorithms

Collaborative Filtering. We chose a memory-based CF approach and used the user-user based recommendation. In this approach, the system offers the possibility of identifying the best neighbors for a given user, using the ratings of users on items. We adopted the Pearson Correlation coefficient [1] to compute the similarity between users, i.e. $Sim_{Pearson}(u_1, u_2)$.

Social Filtering. The SocF is based on social community which is identified using the social similarity between users. This similarity, $Sim_{Soc}(u_i, u_j)$, computes the Jaccard similarity weight between two users u_i and u_j, based on their social relationships, which is defined as the size of the intersection divide by the size of the union of friend:

$$Sim_{soc}(u_1, u_2) = \frac{F(u_1) \cap F(u_2)}{F(u_1) \cup F(u_2)} \tag{1}$$

where: $F(\mathrm{u}_1)$, respectively$F(\mathrm{u}_2)$: is the number of friends of u_1, respectively u_2.

CF-SocF Hybrid Algorithms. The hybridization of the CF and SocF combines the interests based similarity of users with their social similarity weight to compute the overall similarity between two users as follows:

$$Sim_{Hyb}(u_1, u_2) = \alpha\, Sim_{Pearson}(u_1, u_2) + \beta\, Sim_{Soc}(u_1, u_2) \tag{2}$$

where: α and β are weights that express a priority level, with $\alpha + \beta = 1$.

The prediction will be calculated on the basis of the community which is identified using the hybrid similarity between users.

Another way to generate hybrid recommendations is to consider α' items generated by the CF and β' items generated from the SocF, where $\alpha'$and β' are weights that express an importance degree, with $\alpha' + \beta' = 1$.

3.2 Clustering-Based Collaborative and Social Recommendation Algorithms

Clustering–Based CF. We used two classification techniques: an unsupervised technique that uses the K-means and the K-medoids algorithms; and a supervised technique that uses the K-Nearest Neighbors algorithm (KNN). The KNN-based CF algorithm allows predictions using the ratings of the K nearest neighbors of the active user. On the other hand, the K-means-based CF algorithm, allows predictions based on clusters that are generated by the application of the K-means algorithm. The K-means partitioning algorithm is applied to build clusters of users with similar interests. The algorithm randomly selects k initial cluster centroids and repeats the following instructions until reaching the stability of the centroids: (1) Assign each user to the cluster whose centroid is the closest according to a collaborative distance measure (the Pearson correlation function is used as a similarity function); and (2) update the centroids of the k clusters. The following is the K-means-based CF algorithm:

```
Algorithm 1: K-means-based CF ()
Input: usage matrix, number of K clusters, nbIteration
Output: sets of K cluster centroids.
Begin
Initialize the K Cluster Centroids: Randomly designate K
users as Cluster Centroids;
While nbIteration> 0 Do
Find the nearest cluster for each user in the matrix us-
ing the Pearson correlation;
Updated cluster centroids;
Nb Iteration --;
Done
End.
```

Finally, the K-medoids-based CF allows predictions based on clusters that are generated by the application of the Partitioning Around Medoïd (PAM) algorithm (the most common realization of k-medoid clustering). The K-medoids algorithm is more robust to noise and outliers as compared to K-means because it minimizes a sum of pairwise dissimilarities instead of generating a centroid by calculating the average of all the values of each attribute of the cluster. PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search [22]. The following is the PAM algorithm:

```
Algorithm 2: K-medoïds PAM Algorithm ()
Input: usage matrix, number of K clusters
Output: sets of K cluster medoids.
Begin
Initialize K cluster medoids: Randomly designate K users;
Calculate the initial cost;
For each medoid M Do
 For each non medoid O Do
```

```
    Swap (M, O);
    Distribution ();
    Calculate-Cost();
     If cost current iteration> cost previous iteration Then
       Undo (M, O)
     End If
     If no change Then exit algorithm;
     End If
    Done
  Done
End.
```

where:

- *Swap (M, O):* replaces the medoid *M* by the user *O*;
- *Undo (M, O):* cancel the swap of *M* by *O*;
- *Distribution:* Find the closest cluster for each user in the matrix by calculating the Pearson correlation between the user and the K Medoids;
- *Cost:* is an objective function that is intended to be minimized.

$$Cost = min \sum\nolimits_{c \in C} \sum\nolimits_{u,v \in C} d(u, v) \tag{3}$$

where: C is the set of clusters (partitions) resulting from the K-medoïds algorithm; u, v are two users, where v belongs to the cluster and u is the medoid of the latter.

Clustering–Based SocF. In the CF algorithm, partitioning was based on assessments of users with similar interests. However, the social relationship between users can also be used to create partitions, based on friendships. A classification algorithm has been used to create clusters that represent groups of users who are friends or have friendships in common. The prediction will be based on the vote of the neighboring users who are in the same cluster. The following is the clustering algorithm:

```
Algorithm 3: K-means-based SocF ()
Input: Set of users and their friends, number of K clus-
ters, nbIteration
Output: Sets of K cluster centroids.
Begin
Initialize the K cluster centroids: randomly designate K
users as cluster centroids;
While nbIteration> 0 Do
Find the closest cluster for each user by calculating
Jaccard's similarity with the centroid;
Updated cluster centers;
Nb Iteration --;
Done
End.
```

Clustering–Based Hybrid Algorithm. We developed the hybrid algorithm that combines the KNN-based CF and the SocF algorithms. The principle is to find among all the friends of a given user, whose who are similar to him (k more similar) in terms of interests. To illustrate this classification, let's consider an active user *u* with ten friends and K equal to 4. The following Fig. 1 shows the classification that distinguishes the k friends (the most similar in terms of interests to the user u) among the ten friends:

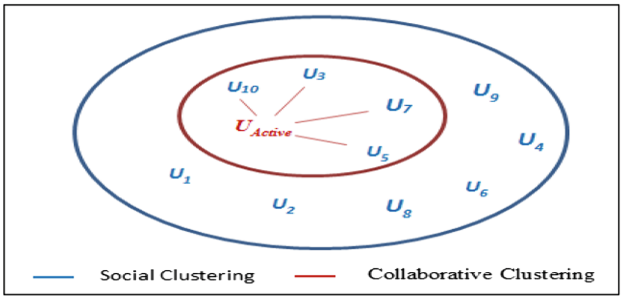


Fig. 1. Clustering-based hybrid algorithm

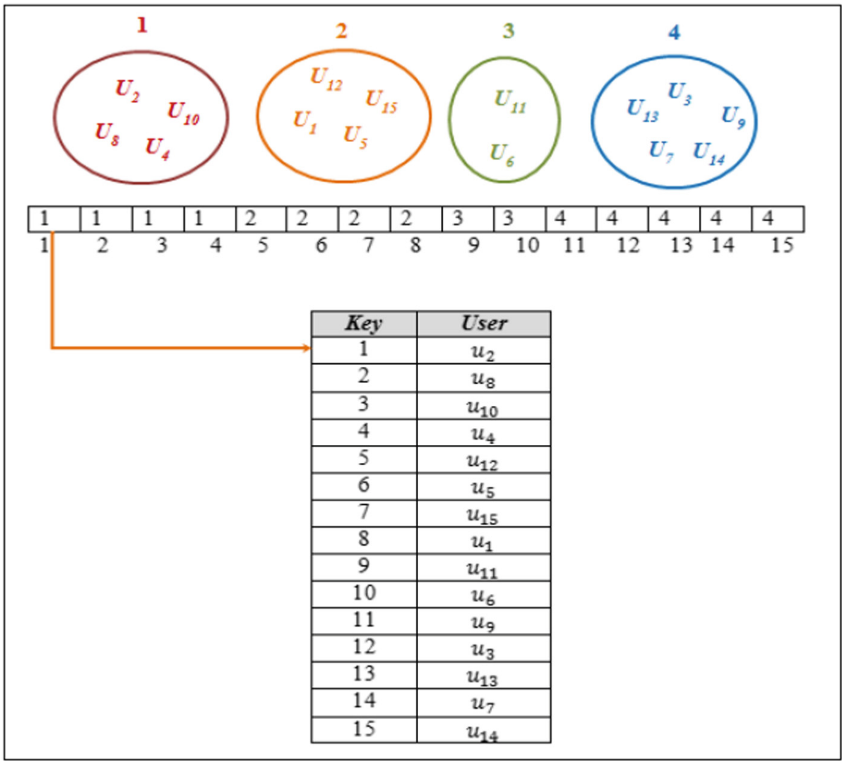
3.3 BSOC-Based Collaborative and Social Recommendation Algorithms

BSOC-Based CF. The Bee colony optimization manipulates a set of bees where each bee corresponds to a feasible solution to a given problem. In order to make the best use of the meta-heuristic and prove its effectiveness, it would be necessary to consider a codification that allows modeling the problem. The purpose of applying this meta-heuristic is to improve the clustering quality resulting from the classification algorithm (k-means/k-medoids) and then to ensure a better recommendation.

Coding and Initialization of the Solution: Each possible partitioning represents a solution, where each bee corresponds to a feasible partitioning. The solution can be represented by a vector containing the different existing clusters. The indices of the vector represent the keys of each user in a hash table. Each box of the vector contains the identifier of the cluster to which the user belongs, where the user having as key the index of this box. The algorithm starts with an initial solution which is a solution vector built from the clusters, resulting after the application of the K-means or K-medoid algorithm, which represents a partitioning of the users into k clusters.

Example: Let's consider 15 users with K = 4. Figure 2 represents the partitioning (obtained after the application of K-Means or K-medoids algorithm), and the representation of the initial solution:

As illustrated in Fig. 2, the users who belong, for example, to cluster 1 are u_2; u_8; u_{10} and u_4 will respectively have as keys the indices 1; 2; 3 and 4 of the table. Similarly, the users of the other three clusters will have keys corresponding to the indices of the linked boxes, given that the users will be classified in order class by class.


1
U_2 U_{10} U_8 U_4
2
U_{12} U_{15} U_1 U_5
3
U_{11} U_6
4
U_3 U_{13} U_9 U_7 U_{14}
1 1 1 1 2 2 2 2 3 3 4 4 4 4 4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Key	User
1	u_2
2	u_8
3	u_{10}
4	u_4
5	u_{12}
6	u_5
7	u_{15}
8	u_1
9	u_{11}
10	u_6
11	u_9
12	u_3
13	u_{13}
14	u_7
15	u_{14}

Fig. 2. Example of BSO clustering.

The BSOC-based CF algorithm describes the application of the BSO algorithm adapted to our problem. Note that whenever a reference solution is added, it must be verified that it did not already exist in the taboo list.

```
Algorithm 4: BSOC-based CF Algorithm ()
Input: Initial Vector V, nbIteration, Flip, maxChance
Output: optimized vector
Begin
Sref = V;
While (nbIteration > 0 || optimal) Do
Insert Sref in the Tabu list;
Determine-Search-Space (Sref, Flip);
For each Bee in Search-Space Do
Perform a local search;
Save Search-Results (dance);
End For
Bee communication to assign the best solution to Sref;
Done
End.
```

Search Space. The main idea is to change n/Flip value of the vector representing the solution and keep the rest of the vector values unchanged, starting from the beginning. Note that n is the size of the vector and Flip (used to determine the different search areas), is a parameter that determines the number of variables to change n/Flip of Sref. It allows diversification from a feasible solution. Too little value given to the flip implies that Sref is close to the region already exploited, i.e. the diversification. A very large value can guide the swarm to very remote research areas.

Fitness Function and Dance Quality. To evaluate the quality of the solution's vector we need to evaluate the constructed clustering. The objective is to evaluate how the elements of the same cluster are homogeneous with each other. The intra-class inertia has been used for this evaluation. A good classification must have a low index of intra-class inertia. The quality of the dance will be represented by this index which is a function which is supposed to be minimized.

Example: Let's consider a solution of n = 15 values (Fig. 3).

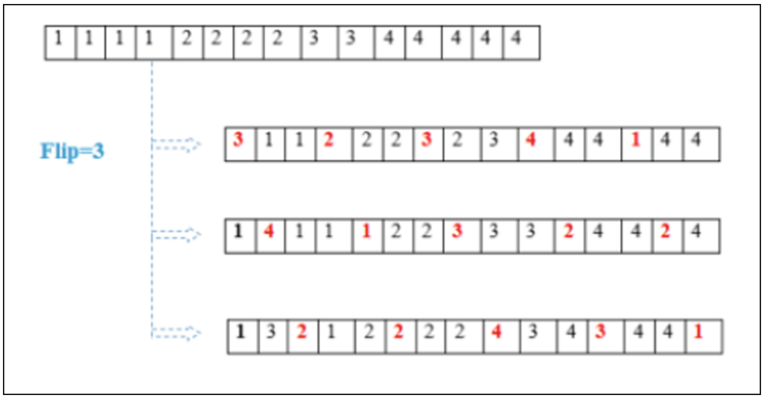


Fig. 3. Example of a solution.

After generating the search areas, each bee conducts a local search and communicates its local optimum to the other bees, in order to select the Sref of the next iteration. At the end of the process, the final solution will have a small fitness value, function close to or equal to zero (optimal = true).

```
Algorithm 5: Bee communication BSO
Begin
Best = best local optimum at time t;
Sref = best solution at time (t-1);
Δf = Inertia (best) - Inertia (Sref);
If Δf >0 Then
Sref=Best;
Nb_chance=Max_chance;
```

```
Else
If Nb_chances > 0 Then
Nb_chance=NB_chance-1;
Sref=Best;
Else
Sref=new solution (generated by K-means / Kmedoids);
Nb_chance=Max_chance;
End if
End If
End
```

where: Max-chance: is an empirical parameter that designates the maximum number of chances given to find a reference solution Sref when there is stagnation.

BSOC-Based SocF. The principle is similar of that of the CF algorithm. The changes are related to the dance and its quality.

Quality of social clustering (the quality of a bee's dance): to evaluate the social clustering, we used the following formula:

$$Dance = \sum\nolimits_{c \in C} \frac{1}{n} \sum\nolimits_{v \in C} \frac{1}{2m} sim(C, V) \tag{4}$$

with:

n: is the number of users and m is the number of the user in a cluster, this is the center of the cluster,
v: a user of the latter.

This formula is supposed to be maximized (the bigger it is, the more the users of the cluster are socially linked).

BSOC-Based CF-SocF Hybrid Recommendation. The main idea is to apply a multiview clustering, which should be optimized in parallel using the BSO meta-heuristics. Multiview clustering has been adapted from (Guo et al. 2015). This method considers a clustering according to both views: interests (ratings) and friendships.

4 Experiments

We conducted empirical experiments in order to evaluate the proposed optimized recommendation approach. Two main research questions have been studied: (1) show the contribution of combining social information with the user-based CF using clustering techniques; and (2) demonstrate the added value of using the BSO optimization on the clustering process and then on the recommendations.

4.1 Dataset

The experiments have been done using the Epinions dataset. The usage matrix was constructed using the Review table. The trust matrix was built using the trust table by taking only bidirectional relationships. A sample of 1052 users was considered for the evaluations.

4.2 Metrics

We used the Mean absolute error (MAE) metric as it is the most popular predictive metric to measure the closeness of predictions relative to real scores.

$$MAE = \frac{\sum_{u,i \in \Omega} \left| r_{u,i} - p_{u,i} \right|}{|\Omega|} \tag{5}$$

where:

Ω: set of test assessments and $|\Omega|$ indicates the cardinality of the set Ω;
$r_{u,i}$: is the rating given by the user u on the item i; and
$p_{u,i}$: is the rating prediction of the user u on the item i.

4.3 Evaluation Results

Contribution of Social Information and Clustering on CF. A preliminary assessment was performed on the CF-based user-user to set the best value of the similarity threshold. The best value of MAE was obtained with a threshold equal to 0.4 (Table 1).

Then, we varied the number of clusters for the CF and SocF algorithms, for the different clustering algorithms. We present the results obtained with the K-means algorithm. The best value of MAE was obtained with k = 30 for CF and K = 60 for SocF:

Table 1. K-means-based CF/SocF evaluation

K	10	20	30	40	50	60
CF	0,890	0,892	**0,870**	0,888	0,897	0,884
SocF	1,261	1,260	1,090	1,098	1,094	**1,093**

Finally, we evaluated the weighted hybrid algorithm that combines the CF and SocF algorithms, and which are based in this evaluation on the application of the K-means classification. We have varied the value of the weight α (which is the degree of importance of the CF) in order to find the value that gives the best recommendation. The evaluation of the hybrid algorithm with a variation of α shows that we obtained the best value of MAE with $\alpha = 0.1$ (see Table 2).

Table 2. Evaluation of the weighted hybrid algorithm

α	MAE
0.1	**0.877**
0.2	0.891
0.3	0.907
0.4	0.911
0.5	0.926
0.6	0.926
0.7	0.926
0.8	0.950
0.9	0.926

Contribution of BSOC on CF. We present the evaluation related to the application of the BSOC-based CF, using the K-means/K-medoids classification algorithms in order to show the contribution of the meta-heuristic on the performances of the CF. The BSO meta-heuristics has several empirical parameters, namely: the Flip, the number of iterations (nbIter) as well as the maximum number of chances. It is important to vary these parameters to find the best values to consider in our assessments.

1. *The BSO-Kmeans-based CF evaluation*

- ***Variation of the number of iterations:*** The evaluation of BSO-Kmeans-based CF with a variation of the number of iterations shows that we obtained the best value of MAE with nbIter = 30 (Table 3).

Table 3. Variation of nbIter-BSO-Kmeans based CF

nbIter	10	20	30	40	50
MAE	0.832	0.810	**0.761**	0.813	0.896

- ***Variation of the Flip:*** The evaluation of BSO-Kmeans-based CF with a variation of the Flip shows that we obtained the best value of MAE with Flip = 3 (Table 4).

Table 4. Variation of the Flip-BSO-Kmeans based CF

Flip	**2**	**3**	**4**	**5**
MAE	0.802	**0.766**	0.757	0.797

- ***Variation of chance number:*** This parameter represents the number of chances given to a reference solution to avoid stagnation. The evaluation of BSO-Kmeans-based CF with a variation of the chance number shows that we obtained the best value of MAE with maxChance = 3 (Table 5).

Table 5. Variation of maxChance-BSO-Kmeans based CF

maxChance	**1**	**3**	**5**	**7**
MAE	0.797	**0.779**	0.789	0.801

2. *The BSO-Kmedoids-based CF evaluation*

The same previous evaluations were performed with the BSO-Kmedoids-based CF. The results obtained are presented below:

- ***Variation of the number of iterations:*** The evaluation of BSO-Kmedoids-based CF with a variation of the number of iterations shows that we obtained the best value of MAE with nbIter = 30 (Table 6).

Table 6. Variation of nbIter-BSO-Kmedoids based CF

nbIter	10	20	30	40
MAE	0.723	0.710	**0.681**	0.712

- ***Variation of the Flip:*** The evaluation of BSO-Kmedoids-based CF with a variation of the Flip shows that we obtained the best value of MAE with Flip = 4 (Table 7).

Table 7. Variation of the Flip-BSO-Kmedoids based CF

Flip	2	3	4	5	6
MAE	0.726	0.710	**0.686**	0.729	0.749

- ***Variation of chance number:*** The evaluation of BSO-Kmedoids-based CF with a variation of the chance number shows that we obtained the best value of MAE with maxChance = 3 (Table 8).

Table 8. Variation of maxChance-BSO-Kmedoids based CF

maxChance	1	3	5	7
MAE	0.721	**0.691**	0.733	0.735

Results Summary for the CF Algorithm. To demonstrate the contribution of the optimized clustering on the CF recommendation, we presented the results by the following table, where the value of MAE represents the best value obtained in the previous preliminary assessments. For the KNN-based CF, the best value of MAE was obtained with K = 70. On the other hand with the unsupervised algorithms, the best value of MAE was obtained with K = 30 for K-means and K-medoids (Table 9).

Table 9. Summary of the results obtained for the CF.

Algorithm	MAE
KNN-based CF	0.731
Kmeans-based CF	0.870
Kmedoids-based CF	0.733
BSO-Kmeans-based CF	0.761
BSO-Kmedoids-based CF	**0.681**

As illustrated in Fig. 4, we can easily see that the supervised classification algorithm was more efficient, in terms of recommendation accuracy compared to the two unsupervised algorithms. On the other hand the integration of the optimization has surpassed all the algorithms of recommendation, where we can notice a clear improvement of the value of MAE with BSO-Kmeans-based CF compared to Kmeans-based CF and also with BSO-Kmedoids-based CF compared to Kmedoids-based CF.

In this evaluation, the K-medoids algorithm performed better than K-means, but in order to confirm this result, it is necessary to carry out other more in-depth evaluations on larger databases, in order to better see the contribution of optimization on the different algorithms.

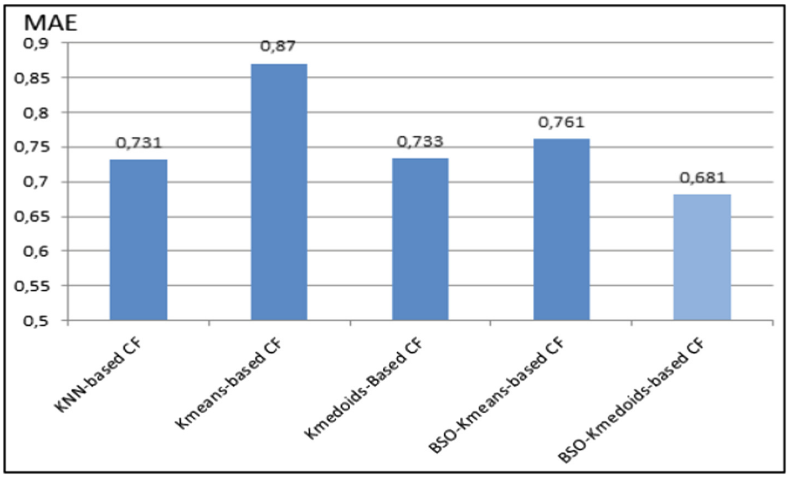


Fig. 4. Contribution of the BSO meta-heuristic on CF.

BSOC-Based Hybrid Recommendation. We evaluated the different hybrid recommendation algorithms (see Fig. 5): (1) the clustering-based hybrid algorithm (the weighted hybrid algorithm using the k-means or K-medoids algorithms and the multi-view clustering algorithm); and (2) the BSOC-based Hybrid algorithm.

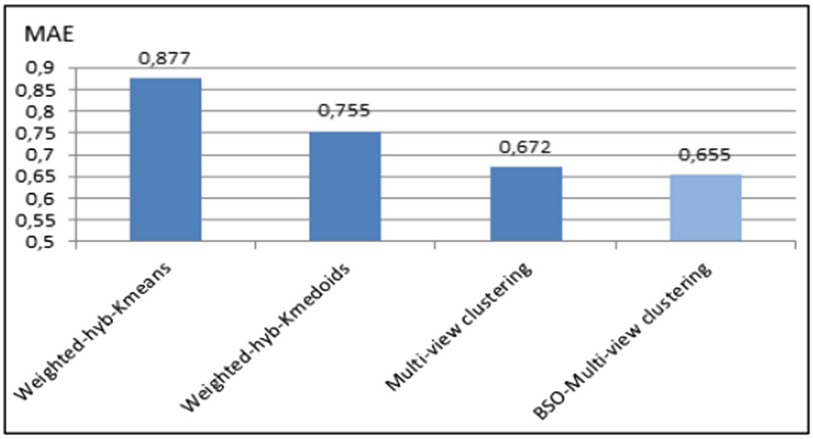


Fig. 5. Contribution of the BSO meta-heuristic on the hybrid recommendation algorithm.

The results we have obtained show the contribution of integrating the social information with the CF, the contribution of the multi-view clustering algorithm compared to the weighted hybrid algorithm, and finally the contribution of the optimized multi-view clustering algorithm compared to multi-view clustering algorithm that outperformed all other results by providing the best accuracy in terms of MAE.

5 Conclusions

This paper proposes a novel recommendation approach which applies an optimized clustering method for the recommendation of items. First the CF has been integrated with social information, using supervised and unsupervised classification algorithms. Then in order to optimize the recommendation accuracy, the Bees Swarm Optimization algorithm was applied in the classification process. The results of the evaluations we carried out using the Epinions dataset show the effectiveness of our system compared to the CF in terms of accuracy using the MAE metric.

In our future work, we plan to perform additional evaluations with larger databases, use other optimization and classification algorithms especially those non-k dependent, and consider other aspects of the social information (e.g. influence, implicit trust).

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. (TKDE) **17**, 734–749 (2015)
2. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10

3. Sarwar, B., Karypis, G., Konstan, R.J.: Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering. In: Proceedings of the 5th International Conference on Computer and Information Technology, pp. 158–167 (2002)
4. Guo, G., Zhang, J., Yorke-Smith, N.: Leveraging multi-views of trust and similarity to enhance clustering-based recommender systems. Knowl.-Based Syst. **74**, 14–27 (2015)
5. Singla, P., Richardson, M.: Yes, there is a correlation: from social networks to personal behavior on the web. In: Proceedings of the 17th International Conference on World Wide Web (WWW), pp. 655–664 (2008)
6. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys), pp. 17–24 (2007)
7. Guo, G., Zhang, J., Thalmann, D.: Merging trust in collaborative filtering to alleviate data sparsity and cold start. Knowl.-Based Syst. **57**, 57–68 (2014)
8. Ma, H., King, I., Lyu, M.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), ACM, pp. 203–210 (2009)
9. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of the 4th ACM Conference on Recommender Systems (RecSys), pp. 135–142 (2010)
10. Massa, P., Avesani, P.: Trust-aware collaborative filtering for recommender systems. In: Federated International Conference on the Move to Meaningful Internet (2004)
11. Guo, G., Zhang, J., Thalmann, D.: Merging trust in collaborative filtering to alleviate data sparsity and cold start. Knowl.-Based Syst. (KBS) **57**, 57–68 (2014)
12. Ma, H., Zhou, D., Liu, C., Lyu, M., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 287–296. ACM, New York (2011)
13. Wang, X., Huang, W.: Research on social regularization-based recommender algorithm. Math. Comput. Modell. **1**, 77–80 (2014)
14. Pham, M., Cao, Y., Klamma, R., Jarke, M.: A clustering approach for collaborative filtering recommendation using social network analysis. J. Univ. Comput. Sci. **17**, 583–604 (2011)
15. Bellogín, A., Parapar, J.: Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In: Proceedings of the 6th ACM Conference on Recommender Systems (RecSys), pp. 213–216 (2012)
16. Salah, A., Rogovschi, N., Nadif, M.: A dynamic collaborative filtering system via a weighted clustering approach. Neuro Comput. **175**, 206–215 (2016)
17. Sun, Z., et al.: Recommender systems based on social networks. J. Syst. Softw. **99**, 109–119 (2015)
18. DuBois, T., Golbeck, J., Kleint, J., Srinivasan, A.: Improving recommendation accuracy by clustering social networks with trust. Recommender Systems Social Web, pp. 1–8 (2009)
19. Sheugh, L., Alizadeh, S.H.: Merging similarity and trust based social networks to enhance the accuracy of trust-aware recommender systems. J. Comput. Robot. **8**(2), 43–51 (2015)
20. He, X., Kan, M.Y., Xie, P., Chen, X.: Comment-based multi-view clustering of web 2.0 items. In: International World Wide Web Conference WWW 2014, Seoul, Korea, 7–11 April 2014, pp. 771–781. ACM (2014)
21. Selvi, C., Sivasankar, E.: A novel optimization algorithm for recommender system using modified fuzzy C-means clustering approach. Soft Comput. **23**(6), 1–16 (2017)
22. Kaufman, L., Rousseeuw, P.J.: Clustering by means of Medoids. In: Dodge, Y. (ed.) Statistical Data Analysis Based on the Norm and Related Methods, pp. 405–416. North-Holland, The Netherlands (1987)

Hyperledger Fabric Blockchain as a Service for the IoT: Proof of Concept

Saša Pešić[1](✉), Miloš Radovanović[1], Mirjana Ivanović[1], Milenko Tošić[2], Ognjen Iković[2], and Dragan Bošković[2]

[1] Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia
{sasa.pesic,radacha,mira}@dmi.uns.ac.rs
[2] VizLore Labs Foundation, Novi Sad, Serbia
{milenko.tosic,ognjen.ikovic,dragan.boskovic}@vizlore.com

Abstract. Blockchain as a Service for the Internet of Things is an emerging topic in the blockchain research and industrial community, especially relating to increased system consistency, security, and privacy. Blockchains represent highly distributed and autonomous decision-making systems with distributed data and process management. Internet of Things systems share these characteristics, while also bringing the cyber-physical dimension and machine-to-machine interaction concept to the ecosystem of devices and users. Blockchain infrastructure setup and smart contract prototyping are cumbersome tasks. Onboarding an existing system to a blockchain network takes a significant amount of time and manual effort, and incorporating business logic requires a series of complex steps. For IoT systems, these task needs to be carried out having in mind the typical characteristics of such systems: low hardware, storage, and networking capabilities and high node churn and transaction volume. Moreover, these tasks need to be semi to fully automated in terms of workflows that support easy-to-use integration mechanisms for onboarding of diverse IoT infrastructures and on-demand business logic generation. In this paper, we present a Hyperledger Fabric-based Blockchain as a Service for addressing the identified challenges. Specifically, the framework is tailored to answer to specific requirements of IoT systems through three major services: Hyperledger Fabric Infrastructure Configuration Generator, Hyperledger Fabric Chaincode Builder and Hyperledger Fabric Operator Modules.

Keywords: Blockchain · Internet of Things · Distributed systems

1 Introduction

Driven by artificial intelligence, cognitive computing and new solutions for device-to-device connectivity as well as rising technologies concerning big data and data analytics, the adoption of the Internet of Things (IoT) is accelerating rapidly. IoT applications are distributed by nature. Hence, distributed ledger technology (DLT) such as blockchain has the potential to play a big role in how devices

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 172–183, 2019.
https://doi.org/10.1007/978-3-030-32065-2_12

communicate in IoT. Since blockchains are designed as systems that provide the basis for transaction processing or semantically richer smart contracts, they can be leveraged to model IoT processes. This way, blockchains can improve not just data integrity, confidentiality, and compliance in the IoT, but also enhance IoT features and cost-efficiency.

Integrating a full blockchain framework into an existing industrial IoT system is a cumbersome task. It requires setting up an entirely separate infrastructure, that also often requires specific technologies, protocols, but also hardware requirements in terms of storage space, CPU utilization and RAM consumption. For applications in IoT, systems, frameworks, mechanisms, and technologies that leave a small CPU/RAM footprint are preferred, but are also often an imperative. Blockchains typically requires larger processing capabilities than that of a typical node in an IoT ecosystem (e.g. smartwatch, embedded device, Raspberry Pi). For the integration between IoT and blockchains to be successful, Blockchain as a Service (BCaaS) should enable most of IoT devices to participate in the blockchain ecosystem, through tailored communication modules. Automation of existing tasks in blockchain deployment and operation presents a set of challenging tasks for BCaaS since blockchains represent highly distributed systems which are hard to deploy, manage, and operate on. Lastly, another relevant issue connected specifically to IoT devices is the lack of support for 64-bit operating systems (OS), as nodes in most blockchain frameworks require 64-bit OS.

BCaaS for IoT should allow multiple classes of heterogeneous devices to seamlessly communicate in a homogeneous blockchain environment. A BCaaS framework should be able to support integrating with heterogeneous environments – systems running on multiple physical machines with different technological footprints. That is why BCaaS aimed at IoT should have designated mechanisms to support deployment even to most resource-constrained devices (embedded devices, sensors, etc.). Creating and deploying a blockchain network is not a simple task – thus BCaaS should handle creating network infrastructure and later deployment in an easy, user-friendly manner through specific interfaces. Implementation of business logic should be enabled with tools and processes aimed at helping users, devices, or processes in generating and deploying the smart contract. Furthermore, efforts in BCaaS must be also directed towards monitoring - specifically blockchain exploration (transactions, blocks, users), device's health inspection (cpu/ram stats, tps, etc.) and topology changes capture, resource sharing and computational offloading and managing complex privacy models, but these concepts will not be the focus of this paper.

In this paper, we plan to address the above-mentioned challenges through showing that semi-automated blockchain deployment, business logic prototyping and integration with existing IoT systems can be easily managed through specifically tailored deployment workflows and mechanisms with Hyperledger Fabric (HF) blockchain technology. Specifically, our contributions rest in the following: Building an automated blockchain infrastructure specification and deployment component with Docker containerization (Hyperledger Fabric Infrastructure Configuration Generator - HFICG); Building an automated smart contract prototyping component with deployment mechanisms in-place (Hyperledger Fabric Chaincode Builder - HFCB); and Building a Hyperledger Fabric Operator Module (HFOM) to offer protected REST API-based communication with the underlying blockchain network (with a version of the module for resource-constrained devices). These components are implemented as parts of a commercial product [15].

The rest of the paper is structured as follows: In Sect. 2 we will discuss related work; in Sect. 3 we explain Hyperledger Fabric infrastructure and vital components; in Sect. 4 we elaborate on our proof-of-concept implementation; Sect. 5 offers a discussion over the solutions presented in Sect. 4; finally, we conclude the paper with Sect. 6.

2 Related Work

Integration of blockchain and IoT ecosystems has become an overarching industrial and academic topic covering many use-cases: IoT access control [8], secure firmware updates [5], energy systems [6], and so on. Concepts of collaboration frameworks for blockchain and IoT have been of research interest in terms of digital asset management [11], management and configuration of IoT devices [4], secure information distribution [9], and so on.

Concerning HF, however, much of the actual research is focused on performance benchmarking models [10,14]. While it is important to maximize the performance of the blockchain, enabling automated frameworks for generating configurations and deploying nodes is necessary, since HF brings a lot of complexity in configuration [3,17]. Use cases like industrial IoT deployments (microcontrollers, sensors, etc.) benefit from automated processes since they rely on machine-to-machine communication with little to none human interaction.

Although there are an abundance of commercial BCaaS tools (i.e by Google, Amazon, Oracle, IBM and others SMBs [2,16], etc.), none of them is equipped to tackle the challenges in using private blockchains in IoT systems through creating mechanisms for easy deployment, business-logic maintenance and high node churn resilience. To the best of our knowledge, this is a novel approach to addressing proof-of-concept tools, implementations, and discussion on the topic of autonomous onboarding of distributed systems to private permissioned blockchains, and automated business logic prototyping using Hyperledger Fabric, for IoT.

3 Hyperledger Fabric

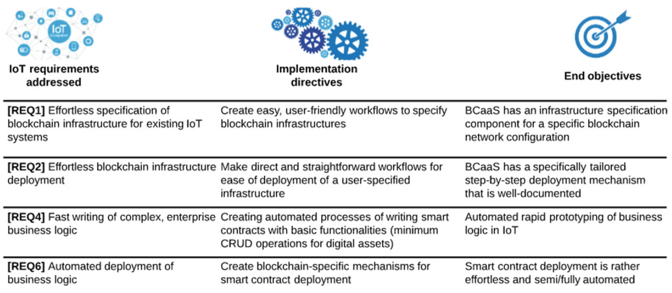
Hyperledger Fabric is an enterprise-grade distributed ledger based on blockchain that leverages smart contracts to enforce trust between parties. It is a platform for distributed ledger solutions, underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility and scalability [7].

Most important components of a HF blockchain are the domain, organizations, peers, orderers and certificate authorities. Domain represents a top-level network and project name. The organizations are the containers for the peers and respective certificate authorities (CA). Organizations are used for physical separation of the blockchain network to participating entities. The peers are nodes which are connected to clients and are responsible for committing transactions to the world state. Each peer has its copy of all transactions, and an organization can have multiple peers (i.e. for redundancy, physical separation). Ordering service provides a shared communication channel to clients and peers, offering a broadcast service for messages containing transactions. CA is responsible for creating HF certificates. It is used for verifying ownership in the network. Each certificate authority is tied to an organization. Other relevant concepts include channels and chaincode. Channels represent separate instances of ledgers and they are independent. A channel has a list of peers joined to it and instantiated chaincodes. Chaincode is the HF term for a smart contract. It represents the business logic behind a blockchain application written in programming code that can read from the ledger and write to the ledger. Chaincode is the only component that can manipulate the ledger. When the chaincode is created an endorsement policy must be provided. The policy can be any boolean logic (all peers, one peer from each organization, etc.) explaining who has to endorse a transaction. Policies are created per chaincode and are checked by the orderer service.

3.1 Addressing IoT BCaaS Requirements with Hyperledger Fabric

In our previous paper, we have identified 12 important BCaaS requirements for integration with IoT systems, categorizing them to 5 groups: Easy Network Specification and Deployment, Rapid Business Logic Prototyping, Resource Sharing, Complex Privacy Management Support, External Interoperability and Data Federation [13]. This paper focuses on the first two categories and four requirements (see Fig. 1), directly connecting these requirements with the Hyperledger Fabric DLT technology and highlight implementation directives for BCaaS. A proof-of-concept design and implementation for these requirements will be presented in Sect. 3, focusing on HFICG, HFCB, and HFOM. The tools addressed in this paper are implemented independently (and can be tested online) as part of a commercial product called ChainRider [15] (code not available publicly).

HF is suitable for BCaaS-IoT framework due to its in-built modularity of components, allowing them to be physically and logically distributed through the entire deployment ecosystem (IoT platform). The cost of running HF components (ordering service, peers, chaincodes) are rather small in comparison with Ethereum or Quorum. Specific tools can be built around automated HF infrastructure specification, generation, and deployment. Especially, leveraging HF binaries for cryptographic material generation can be automated, and complex networks can be built in a matter of seconds, provided that the tool has input about: blockchain network participants (organizations and peers), ordering service and certificate authority specifications, channels and chaincodes. Fast and easy writing and deployment of smart contracts can be enabled through specific software modules enabling input-based creation of Node.js, Golang or Java smart contracts with a basic interface and CRUD (and other) functions - similarly how you can inherit the basic token smart contract in Ethereum and build upon it. Deployment mechanisms, although not complex, can be automated by specifying channels and specific endorsement policies per chaincode.



IoT requirements addressed	Implementation directives	End objectives
[REQ1] Effortless specification of blockchain infrastructure for existing IoT systems	Create easy, user-friendly workflows to specify blockchain infrastructures	BCaaS has an infrastructure specification component for a specific blockchain network configuration
[REQ2] Effortless blockchain infrastructure deployment	Make direct and straightforward workflows for ease of deployment of a user-specified infrastructure	BCaaS has a specifically tailored step-by-step deployment mechanism that is well-documented
[REQ4] Fast writing of complex, enterprise business logic	Creating automated processes of writing smart contracts with basic functionalities (minimum CRUD operations for digital assets)	Automated rapid prototyping of business logic in IoT
[REQ6] Automated deployment of business logic	Create blockchain-specific mechanisms for smart contract deployment	Smart contract deployment is rather effortless and semi/fully automated

Fig. 1. Hyperledger Fabric BCaaS IoT requirements addressed in this paper

4 Proof-of-Concept BCaaS Design and Implementation

This section covers implementation details of HFOM, HFICG, and HFCB. In this Section, we address **REQ1**, **REQ2**, **REQ4** and **REQ6** from our previous paper [13] (see Fig. 1). **REQ1** and **REQ2** belong in the Easy Network Specification and Deployment category. Their combined aim is the effortless specification of HF blockchain infrastructure and automated deployment. **REQ4** and **REQ6** belong in the Rapid Business Logic Prototyping category, and their objective is to enable fast writing of complex enterprise business logic through automated workflows with instant-deployment enabled. The rest of the requirements will be addressed with implementation as part of our future work, and we do not provide implementation details nor discussion at this point. All of these requirements address key IoT challenges: complex and distributed IoT topologies with heterogeneous devices having multiple technological footprints should have access to create and join a HF blockchain in a straightforward, guided workflow; IoT devices can create on-demand business logic components leading to rapid business functions modeling.

4.1 Hyperledger Fabric Operation Module

The Hyperledger Fabric Operation Module (HFOM) is a core component that handles secure communication between users, applications and the underlying blockchain infrastructure. The HFOM is a Node.js REST application that is designed to operate on any operating system with minimal CPU/RAM footprint. It is a dockerized service whose resources are accessible through a set of asynchronous token-based REST APIs. The design of the components and its services is completely asynchronous - as the underlying blockchain system naturally processes transactions asynchronously. To address the low resource requirements for IoT, HFOM offers three distinct deployment types:

1. Full HFOM with HF components running locally (on the same machine) - this HFOM requires HF peer node(s) to be deployed on the device running the HFOM. This deployment option offers fastest ledger Read/Write operations capabilities since it directly communicates with a locally running peer node. HFOM has all cryptographic materials of the network locally stored. It is implemented in Node.js. It is suitable for devices capable of addressing at least 1 Gb of RAM memory and a 1.3 GHz CPU, such as the Raspberry Pi 3B. The capabilities also depend on the HF components running locally.
2. Full HFOM without HF components running locally - HFOM is connecting to a specific set of functioning peer nodes in the local IoT system. HFOM has all cryptographic materials of the network locally stored. It is implemented in Node.js. It is suitable for all devices with a minimum of 256 Mb of RAM (such as Raspberry Pi 0, or Orange Pi 0).
3. Soft HFOM - connects to the closest full HFOM (1 or 2) and delivers its request using curl and HTTP or HTTPS. Its size is insignificant and it leaves the minimal CPU/RAM footprint and is suitable for most resource-constrained devices. It is implemented in Node.JS, Python, and Bash. It can run as a dockerized service, or as standalone service. It is suitable for devices that have under 128 Mb of RAM (such as Onion Omega 2).

To run properly the HFOM needs a blockchain network specification in *yaml* format. The specification is generated as part of the HFICG (Fig. 2). HFOM on all levels can perform the following actions: enroll blockchain users, create channels, join peers to channels, install chaincodes, instantiate chaincodes, invoke chaincode functions, query ledger data, fetch specific blocks, fetch specific transactions, query installed/instantiated chaincodes on a specific channel and fetch channel information. HFOM on all levels have access to the HFICG and HFCB, as well as ledger data. Applications never directly interact with the core components (Peer, Orderer). All communication is proxied through the HFOM, building another security layer on top of already existing blockchain layer. Lastly, HFOM takes care of identity management on different levels of blockchain operation. By interacting with a Certificate Authority tied to the specific part of the network (usually organization), HFOM deals with certificate generation and revocation and transacting identities creation and removal on the blockchain. HFOM stores app certificates at any point in time until the access token tied to the certificate is expired, or the app's certificate has been revoked by the Certificate Authority. Access tokens are refreshed on a 10-minute basis, to minimize token hijacking probabilities.

4.2 Hyperledger Fabric Infrastructure Configuration Generator

HFICG is used for the creation of all necessary configuration and cryptographic material for an IoT system that is ready to join a HF blockchain network. HFICG component is designed to incorporate effortless creation of complex HF blockchain networks, with multiple organizations, channels, and various peer-channel connections, as well as provisioning all of these components to any desired number of physical machines. For any given system description that is provided to the HFICG, the component will generate a set of files and folders necessary for instant deployment and starting of the blockchain network, including HFOM deployment options of choice per physical machine. The workflow of HFICG is displayed in Fig. 2.

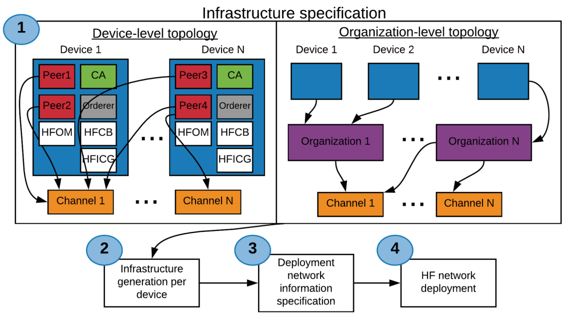


Fig. 2. Hyperledger Fabric Infrastructure Configuration Generator

Step 1 of Fig. 2 indicates how infrastructure is specified in HFICG. The HF network infrastructure to be generated is described with two graphs, one being the device-level topology graph and the other organization-level topology graph. Through a simple user interface, these two graphs are made using drag-and-drop for adding HF and BCaaS components. In the same step, the graphs are transformed into an adequate JSON representation and passed to step 2. If an IoT system is to be a part of the Hyperledger IoT fabric network, a set of configuration files is required to be deployed together with the core Hyperledger Fabric components on each of the devices of an IoT system (step 2). All configuration and cryptographic material must be created on one machine, and later distributed to all parts of the system. The generated HF blockchain network deployment materials include: docker-compose file for the Ordering service, cryptographic material, genesis block, and a Bash startup script to bring up the Ordering service containers; docker-compose file (including: peers, state database and Certificate Authority - CA containers), cryptographic materials, channel creation and anchor peer configuration files, and a Bash startup script to bring up peers, state database and CA server configuration (including: channel creation commands, anchor peer update commands, channel join command per peer) per machine; HFOM/HFCB docker-compose files per machine and a Bash startup script to bring up the HFOM/HFCB services. To answer to the requirement of IoT systems having significant node churn, HFICG offers an easy extension of existing networks in terms of adding new organizations, devices, channels, and HF and BCaaS components, through the same mechanism displayed in Fig. 2.

4.3 Hyperledger Fabric Chaincode Builder

Business logic in HF is specified through smart contracts or chaincodes in HF terminology. Chaincodes can be written in multiple programming languages: Golang, NodeJS, and Java. HFCB is designed as a fully-automated chaincode generation and deployment component. It handles all steps from describing the smart contact in a predefined format, to having the chaincode up and running on the blockchain network. This means that if there is an application interacting with the blockchain (case 1), and it is in need for a new smart contract that handles a new data stream, it can autonomously ask for a chaincode to be created through HFCB. Once the chaincode it is created by the HFCB the application is notified, and it can start interacting with the new chaincode. If there is a system user (human) interacting with the blockchain (case 2), this procedure is semi-automated with additional steps to cover on-the-fly smart contract editing. Both HFCB workflow cases are displayed on Fig. 3.

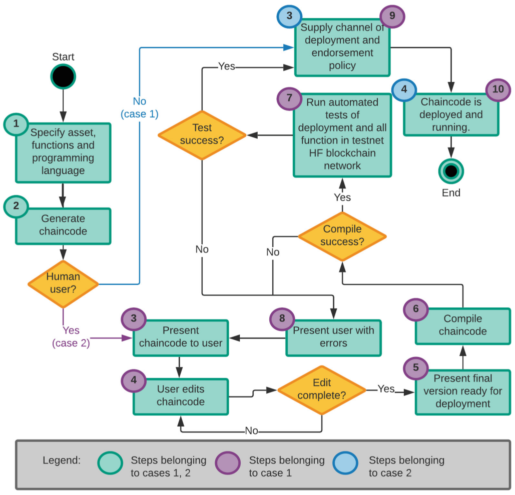

Start
1 Specify asset, functions and programming language
2 Generate chaincode
Human user?
No (case 1)
Yes (case 2)
3 9 Supply channel of deployment and endorsement policy
Yes
Test success?
7 Run automated tests of deployment and all function in testnet HF blockchain network
4 10 Chaincode is deployed and running.
End
Yes
Compile success?
No
No
3 Present chaincode to user
8 Present user with errors
6 Compile chaincode
4 User edits chaincode
Edit complete?
Yes
5 Present final version ready for deployment
No
Legend:
Steps belonging to cases 1, 2
Steps belonging to case 1
Steps belonging to case 2

Fig. 3. Hyperledger Fabric Chaincode Builder

Both workflow cases start with the description of the digital asset/record, central to the chaincode, available functions, and the programing language (step 1). A ledger record is usually specified as an arbitrary Javascript (JSON) object. Essentially, when generating a smart contract, we are specifying two things: (1) how this record object will look like (its attributes and their types) and (2) what functions are going to be performed with it. For illustration, let us consider a simple IoT use case: we have a smart home environment with temperature sensors collecting data from different rooms in the house. To record these sensory data on the HF blockchain we need to have a chaincode that will represent temperature reading objects and allow for sensory data to be written on the blockchain. Every temperature reading that is going to be written to the blockchain will contain the room where it was recorded, the value that was recorded, and a unique key. Room will be a string such as a kitchen or living room, value is the temperature value. HFCB offers to include following functions on the given asset: insert asset, update asset, update asset by attribute value, associate asset with a different key, remove asset from the state, query asset by key, query asset by key range, query asset by CouchDB query string. An application can use a subset of the functions. Programing languages of choice are Node.js, Golang, and Java. Both workflows continue to the generation of code and dependency files (step 2).

Once the chaincode files have been generated, the application/device is asked to supply the deployment channel and the endorsement policy (step 3, case 2). After that (step 4, case 2) the chaincode is up and running, and ready to serve requests.

The case 1 workflow continues by presenting the generated chaincode to the user (step 3, case 1). On the chaincode, a user can perform many edits (step 4, case 1) to add new functions, change existing ones, etc. When editing is complete user is presented with the final version (step 5, case 1). This final version is then compiled (step 6, case 1) and if the compilation succeeds the chaincode is deployed and tested in a testnet blockchain (step 7, case 1). If the compilation fails, the error messages are presented to the user (step 8, case 1), and the user is back at step 3 of case 1. Outputs from step 7 of case 1 are presented to the user, and if not satisfied, the user can go back to the editing step (step 4, case 1). While step 6 eliminates all possible syntax errors, step 7 makes sure that there are no chaincode runtime errors. If deployment tests finished without errors, the user is asked to select a channel where the chaincode will be deployed and the endorsement policy (step 9, case 1). Once that step is completed the chaincode is up and running and begins accepting transactions (step 10, case 1).

5 Discussion

The implemented components are used and tested in three real-world IoT deployments: Agile IoT platform (see Acknowledgement Section), the Arizona State University Blockchain Research Laboratory Carbon trading IoT platform [1] and our Asset Tracking fog computing solution [12]. During development and testing, we have generated 50 HF infrastructure configurations with varying complexity, and 150 chaincodes for deployed IoT solutions mentioned in the previous sentence.

The most complex example we have tested with HFICG is the following infrastructure specification: 30 organisations with 40 channels (1 channels per organisation, and 10 channels shared between multiple organisations); Kafka

Ordering service (7 Kafka nodes, 5 Zookeeper nodes); a CouchDB state database Docker container per machine (omitted for low machine CPU/RAM capabilities); a HFOM per machine (with the suitable deployment option (see Sect. 4.3) depending on the machine CPU/RAM capabilities); a Certificate Authority container per organisation; Each organisation runs 5 machines, making a total of 150 machines for this deployment; Each machine runs 0–2 endorsing peers (depending on the machine CPU/RAM capabilities), making a total of 300 peers. This HF configuration is generated with HFICG within 7 min. With network information supplied for all participating machines, we have measured the amount of time necessary to: 1. deploy configuration files; 2. start HF components in Docker containers; 3. Perform channel creation, peer channel joins, and anchor peer updates. For example, on a Raspberry Pi 3 B+ device, steps (1), (2) and (3) are done within 3 min. Note that this is the time in an ideal scenario. Due to network issues and deployment retry-policies, the deployment time varies.

We have simulated the above-mentioned deployment on the Azure Cloud Platform. Virtual machine specifications regarding CPU and RAM were set to lowest to mimic typical IoT devices – two machines types were randomly assigned to each device: first having 1 CPU and 0.5 Gb RAM, second having 1 CPU and 1 Gb of RAM. The machines with lower settings run 0–1 peers (assigned randomly), HFOM with deployment option 3 (see Sect. 4.1), and no CouchDB container. The Ordering service node was deployed to a stronger machine (8 CPUs, 32 Gb RAM). Since deployment is done in parallel for all machines, the entire blockchain network was deployed, set up and running in under 15 min. Note, that most of the time is spent on machines downloading necessary Docker images to run HF components and setting up HFOM (approx. 70%).

The most complex example we have tested with HFCB is a chaincode for a digital asset with 30 attributes, having all functions (CRUD and three different query functions) with a CouchDB index created for 15 attributes when performing inserts. The chaincode is generated within 3 seconds, and deployed (through a HFOM setup on a Raspberry Pi 3 B+), up and running within another 100 seconds, making a total execution time under 2 min.

In conclusion, one can have a complex, fully functional HF network with on-demand business logic up and running in under 25 min with HFICG, HFCB and HFOM. Furthermore, with HFICG and HFCB the time required to deploy infrastructure and business logic is significantly diminished over manual writing of configuration files, moving files to dedicated machines and starting all HF components.

Lastly, we argue that security and privacy of IoT systems can be strengthened through using a blockchain infrastructure such as HF. Transparency levels for data can be adjusted according to the desired use case through channels modeling or private data specification. Since channels represent physically separate data ledgers, data privacy and security can be managed within a certain set of organisations, even peers. User/Device level attributes can be tied to user/device certificates and access policies for data can be enforced through chaincodes at runtime. To ensure proper transaction validation, chaincode endorsement policies can be set to have varying endorsement requirements. Data access is secured via strict checking of signatures/certificates at all steps: HFOM/HFICG/HFCB, peer level and chaincode level. Unauthenticated reads and writes are thus automatically discarded. Having a supportive HF infrastructure makes on-demand traceability and auditability of actions simpler. Malicious actors and actions can be detected, and machines/users blacklisted.

6 Conclusion

In this paper we have built a proof-of-concept BCaaS system using Hyperledger Fabric blockchain technology. This paper showcases implementation details from a set of BCaaS requirements for IoT systems defined in our previous paper on the topic [13].

We have discussed BCaaS in IoT, and provided an implementation of several functionalities, making this PoC the first, relevant BCaaS PoC that considers: automated blockchain network infrastructure generation and deployment in a physically distributed system environment and rapid business logic prototyping through hastened smart contract writing and deployment. With HFICG configuration specification, infrastructure setup and deployment time are significantly diminished. HFCB makes a large impact where fully-automated on-demand chaincode generation and deployment is necessary. Lastly, HFOM makes a move towards providing a tool for fast and secure communication with the underlying blockchain infrastructure. This framework of tools has been already validated through integration with Agile IoT platform. Furthermore, our framework has been used by Arizona State University Blockchain Research Laboratory in context of their Carbon trading IoT platform development and is providing a HF blockchain network for our BLEMAT Asset Tracking IoT solution. The tools mentioned in the paper are offered as parts of a commercial solution [15].

As part of the future work our focus will be complementing the BCaaS framework until the complete set of BCaaS services from our previous paper is implemented. We will aim for integration with other IoT platforms, while also carefully reviewing new BCaaS requirements, implementation recommendations and end objectives. Lastly, IoT platforms are always subject to a degree of uncertainties, particularly as regards as their number of components and the frequency that they are running. As part of future work we will work on larger experimental studies to see how the BCaaS could work in an environment when new components are frequently added or removed over time.

Acknowledgement. The research and outcomes behind this paper are a part of the open call research project for Agile IoT, H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Project ID: 688088. The authors also thank the Ministry of Education, Science and Technological Development of the Republic of Serbia for support through project no. OI174023 – "*Intelligent techniques and their integration into wide-spectrum decision support*".

References

1. Arizona State University Blockchain Research Laboratory: Carbon trading on a blockchain (2018). https://blockchain.asu.edu/carbon-trading-on-a-blockchain. Accessed 18 May 2019
2. ChainStack: Managed blockchain infrastructure. https://chainstack.com/. Accessed 10 August 2019
3. Gaur, N., Desrosiers, L., Ramakrishna, V., Novotny, P., Baset, S.A., O'Dowd, A.: Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer. Packt Publishing Ltd., UK (2018)
4. Huh, S., Cho, S., Kim, S.: Managing IoT devices using blockchain platform. In: 2017 19th International Conference on Advanced Communication Technology (ICACT), pp. 464–467. IEEE (2017)
5. Lee, B., Lee, J.H.: Blockchain-based secure firmware update for embedded devices in an internet of things environment. J. Supercomputing **73**(3), 1152–1167 (2017)
6. Li, Z., Kang, J., Yu, R., Ye, D., Deng, Q., Zhang, Y.: Consortium blockchain for secure energy trading in industrial internet of things. IEEE Trans. Industr. Inf. **14**(8), 3690–3700 (2018)
7. Linux Foundation: Hyperledger fabric blockchain (2019). https://hyperledger-fabric.readthedocs.io. Accessed 18 May 2019
8. Ouaddah, A., Abou Elkalam, A., Ait Ouahman, A.: Fairaccess: a new blockchain-based access control framework for the internet of things. Secur. Commun. Networks **9**(18), 5943–5964 (2016)
9. Polyzos, G.C., Fotiou, N.: Blockchain-assisted information distribution for the internet of things. In: 2017 IEEE International Conference on Information Reuse and Integration (IRI), pp. 75–78. IEEE (2017)
10. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. IEEE (2017)
11. Samaniego, M., Deters, R.: Blockchain as a service for IoT. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 433–436. IEEE (2016)
12. Pešić, S., et al.: Blemat-context modeling and machine learning for indoor positioning systems (2019), submitted for review in International Journal on Artificial Intelligence Tools (ISSN: 0218–2130)
13. Pešić, S., et al.: Conceptualizing a collaboration framework between blockchain technology and the internet of things (2019). Accepted to CompSysTech'19
14. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 264–276. IEEE (2018)
15. VizLore: Chainrider blockchain as a service. https://chainrider.io/. Accessed 10 August 2019
16. Worldsibu: Worldsibu: The enterprise blockchain development platform. https://worldsibu.tech/. Accessed 10 August 2019
17. Yewale, A.: Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes. Ph.D. thesis, University of Nevada, Las Vegas (2018)

GraphQL Schema Generation for Data-Intensive Web APIs

Carles Farré(✉), Jovan Varga, and Robert Almar

Universitat Politècnica de Catalunya, BarcelonaTech, Barcelona, Spain
{farre,jvarga}@essi.upc.edu, robert.almar@est.fib.upc.edu

Abstract. Sharing data as a (non-)commercial asset on the web is typically performed using an Application Programming Interface (API). Although Linked Data technologies such as RDF and SPARQL enable publishing and accessing data on the web, they do not focus on mediated and controlled web access that data providers are willing to allow. Thus, recent approaches aim at providing traditional REST API layer on top of semantic data sources. In this paper, we propose to take advantage of the new GraphQL framework that, in contrast to the dominant REST API approach, exposes an explicit data model, described in terms of the so-called GraphQL schema, to enable precise retrieving of only required data. We propose a semantic metamodel of the GraphQL Schema. The metamodel is used to enrich the schema of semantic data and enable automatic generation of GraphQL schema. In this context, we present a prototype implementation of our approach and a use case with a real-world dataset, showing how lightly augmenting its ontology to instantiate our metamodel enables automatic GraphQL schema generation.

Keywords: GraphQL · Data-Intensive Web APIs · Semantic metamodel

1 Introduction

Ontology-driven approaches have proven successful to deal with data integration challenges thanks to the inherent simplicity and flexibility of ontologies, which make them suitable to define the required unified view [9]. Ontologies may be represented in the form of Resource Description Framework (RDF) [4] triples and then made web-accessible via SPARQL endpoints [3]. Nevertheless, web data access at SPARQL endpoints is uncontrolled and unpredictable, compromising the performance, the response time, and even the availability of the service [2]. Moreover, exposing data directly as RDF triples is quite distinct and even disconnected from the "mainstream" Web API development built on HTTP and JSON [8]. Therefore, it is unsurprising that recent approaches aim at providing a REST API layer on top of semantic data sources [5,7,8].

The GraphQL framework proposed by Facebook [6] represents an alternative to the REST APIs. One of its key benefits is that it enables the retrieval of only required data with a single request. This feature is enabled by describing the available data in terms of a *GraphQL schema*. In this way, instead of using several REST method calls, the user of a GraphQL service can define custom requests that precise the data to be retrieved by using the schema information. Therefore, the GraphQL Schema is a foundation for the flexibility introduced.

In this paper, we couple GraphQL and ontologies to take the most out of the two technologies. In particular, we propose GQL as an RDF-formalized semantic metamodel for GraphQL schema and explain how to take advantage of this metamodel to support the semi-automatic generation of a GraphQL schema from a given RDF ontology. Our contributions can be summarized as follows.

- We define GQL, a semantic metamodel for GraphQL Schema.
- We provide a prototype that, given a semantic schema model instantiating GQL, automatically generates the corresponding GraphQL Schema.
- We present a use case with a real-world dataset in order to demonstrate the feasibility of our approach. The use case shows how lightly augmenting the dataset ontology to instantiate our metamodel enables automatic GraphQL schema generation. Moreover, we explain how our prototype is also able to generate a proof-of-concept GraphQL runtime service that answers the queries posed against its GraphQL schema using the available data.

The rest of the paper is structured as follows. Section 2 summarizes the key aspects of GraphQL. Section 3 describes our semantic metamodel for GraphQL Schema: the GQL Metamodel. Section 4 demonstrates, through a use case, how to take advantage of the GQL Metamodel to enable the generation of GraphQL schemas and services. Section 5 discusses the related work. Finally, Sect. 6 presents the conclusions and further work.

2 GraphQL

This section provides a short introduction on GraphQL that we have synthesized from the original documentation published by Facebook [6]. A *GraphQL service* provides web-based access to its underlying data sources. The *GraphQL query language* allows defining the request payloads to be sent to the GraphQL service unique endpoint. Such payloads are JSON-like sentences that may contain one or more operations. There are three types of operations: queries, mutations, and subscriptions. In this paper, we focus on the querying capabilities of GraphQL.

GraphQL queries must be written in terms of the *GraphQL schema* that describes the data sources that the GraphQL service exposes. A GraphQL schema defines a set of object *types*, which are similar to object classes but without operations. In this way, an object type specifies a list of attributes, named *fields*, each of which yields one or more values of a specific type. This latter may be either a *scalar* type (String, Integer, etc.) or another object type.

For the sake of an example, Fig. 1(a) presents a (fragment of the) GraphQL schema that we obtain for our use case dataset about movies (for further details see Sect. 4). Figure 1(b) presents a corresponding GraphQL query that, for a

```
type film {
        genre : [film_genre]
        director : director
        title : String
        actor : [actor]
        date : String
        country : country
        filmid : Int
        performance : [performance]
        idInstance : String!
}

type director {
        director_name : String
        director_directorid : Int
        idInstance : String!
}

type film_genre {
        film_genre_name : String
        film_genre_film_genreid : Int
        idInstance : String!
}

type country {...}
type actor {...}
type performance {...}

type Query {
        allfilms: [film]
        getfilm(id: String!): film
        ...
}
```
(a)

```
getfilm (id: ".../film/100") {
  title
  date
  director {
    director_name
  }
  genre {
    film_genre_name
  }
}
```
(b)

```
{
  "data": {
    "getfilm": {
      "title": "Disraeli",
      "date": "1929",
      "director": {
        "director_name": "Alfred Green"
      },
      "genre": [
        {
          "film_genre_name": "Indie"
        },
        {
          "film_genre_name": "Biographical"
        },
        {
          "film_genre_name": "Drama"
        }
      ]
    }
  }
}
```
(c)

Fig. 1. Example GraphQL schema (a), query (b), and result (c).

given `film` (`id: "../film/100"`[1]), returns its `title`, `date`, director's name (`director_name`), and the names of the genres (`film_genre_name`) to which it belongs. Finally, Fig. 1(c) shows a JSON result returned by the GraphQL service after processing the GraphQL query in Fig. 1(b). This example illustrates two of the main features of the GraphQL query language:

1. GraphQL queries define templates for the data to be retrieved by selecting the specific scalar fields that are of interest to the user.
2. GraphQL queries can be nested, i.e., for each field of the object type in a GraphQL query, even if it is a list of objects, a GraphQL (sub) query can be defined. The query displayed in Fig. 1(b) contains two sub-queries: `director { director_name }` and `genre { film_genre_name }`, which retrieve the names of the objects `director` and `genre` associated with a `film`.

The GraphQL schema in Fig. 1(a) defines several "normal" object types (`film`, `director`, ...) together with the `Query` type. Every GraphQL schema

[1] Shorthand for "http://data.linkedmdb.org/resource/film/100".

must provide a `Query` type to define the available *entry points* for querying the GraphQL service. Consequently, any valid GraphQL query needs to be rooted in one of the supplied entry points. In the example, we show two possible entry points, namely `allfilms` and `getfilm`. The former entry point allows querying the whole collection of films, whereas the latter one allows the direct access to just one film giving its id, as it is the case of the query shown in Fig. 1(b).

3 A Semantic Metamodel

In this section, we define *GQL* as a semantic metamodel for GraphQL schema. The metamodel abstraction level is needed as each API has its own model (i.e., schema) defined in terms of GraphQL schema, which in turn, has its data instances, and such setting can strongly benefit from metamodeling [12]. The metamodel is formalized in RDF and, given the challenge of metamodeling, we first explain the modeling principles, after which we explain the metamodel design and its elements.

3.1 Modeling Principles

As GraphQL schema represents the data model of the data exposed by an API, we designed GQL by following the modeling principles already applied in similar cases. Our approach is inspired by the RDF Data Cube vocabulary, which is the W3C recommendation, and its extension the QB4OLAP vocabulary that aim at representing multidimensional schema to enable OLAP analysis on the SW [13].

GQL includes the following kinds of elements – class, class instance, schema property, and data property (see Fig. 3). The class and schema property elements are used to define an API schema and, thus, they are instantiated only once. In RDF, the former is defined as an instance (via rdf:type) of RDFS class (i.e., rdfs:Class), while latter is an instance of RDFS property (i.e., rdf:Property), e.g., see lines 1 and 2 in Fig. 2. The class instance elements are predefined instances (via rdf:type) of a previously defined class kind in the API schema, e.g., see line 3 in Fig. 2. Finally, the data property elements are instantiated at the API schema level and, if data are also represented with RDF, can be used to retrieve the data instances automatically. For this reason, a data property element is defined as an instance (i.e., via rdf:type) of both the RDFS class (so that it can be instantiated at the schema level) and the subclass of RDFS property so that it can be used as the property in GQL, e.g., see lines 4 and 5 in Fig. 2. At the schema level, a data property element is instantiated as an RDFS property that can be used to retrieve concrete data instances with which it is used and we provide further details on this in the following subsections.

```
gql:Scalar rdf:type rdfs:Class .
gql:hasModifier rdf:type rdf:Property .
gql:Int rdf:type gql:Scalar .
gql:Field rdf:type rdfs:Class .
gql:Field rdfs:subClassOf rdf:Property .
```

Fig. 2. Example of instance level triples

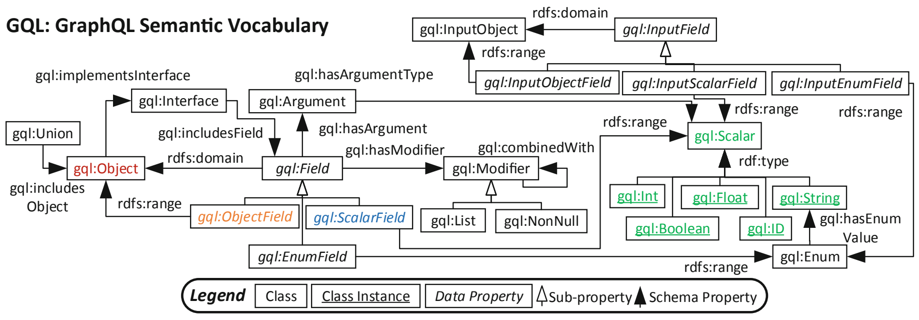


Fig. 3. GQL vocabulary

3.2 GQL Metamodel

GraphQL Schema is organized around GraphQL types. To define our GQL metamode, we first introduce the gql:Type class as the superclass for all GraphQL types. Then, we introduce the following classes:

- gql:Scalar representing primitive values such as string.
- gql:Object organizing scalars or other objects in a single struct.
- gql:Enum representing a set of allowed scalar values.
- gql:InputObject representing complex structs to be used with field arguments (see the next subsection for more details on field arguments).
- gql:Interface representing an abstract type defining a list of fields to be implemented by an object.
- gql:Union representing a list of possible types for a certain field.
- gql:List representing a list of elements of a certain type.
- gql:NonNull denoting that a certain filed cannot take a null value.

For the last two GraphQL types above we introduce the superclass gql:Modifier as they represent modifiers over the other types.

The complete GQL metamodel (excluding gql:Type) is depicted in Fig. 3.[2] In addition to classes for the previously introduced types, the gql:Argument class represents an argument for an object field that determines the value to be returned for that object field. Next, we introduce the data property elements

[2] Metamodel triples: http://www.essi.upc.edu/~jvarga/gql/gql.ttl.

of GQL. As in RDF, each property is directed, a property has its source and target, defined via domain (i.e., rdfs:domain) and range (i.e., rdfs:range) properties.

An object can have different fields that are represented with the gql:Field data property element. A field is always related to an object as its domain, and can relate to another object, scalar, or enum as its range. Thus we define three gql:Field sub-properties – gql:ObjectField having another object as range, gql:ScalarField having a scalar as range, and gql:EnumField having an enum as range. In the same way, we define gql:InputField with its sub-properties gql:InputObjectField, gql:InputScalarField, and gql:InputEnumField that define field for gql:InputObject.

Furthermore, GQL defines the following data properties. If object implements an interface, gql:implementsInterface is used to link the two. Moreover, a union of objects is defined via gql:includesObject that links a union with an object that it includes. The argument for a field is specified via gql:hasArgument linking a field with an argument. Furthermore, the argument is linked with its scalar type via gql:hasArgumentType. Each field can also be linked to a modifier via gql:hasModifier where modifiers can be mutually interlinked via gql:combinedWith, e.g., a field can be a list of non-null values. An enum can be linked to a string value via gql:hasEnumValue.

Considering the predefined class instances, the primitive values considered by GraphQL are defined as instances of gql:Scalar via rdf:type. These include gql:Int, gql:Float, gql:String, gql:Boolean and qgl:ID, and we explicitly define all of them to comply with the GraphQL specification. Note that gql:ID represents a unique identifier used to identify objects.

4 Automation

In this section, we explain how using the GQL Metamodel enables the semi-automatic generation of GraphQL schemas from a given ontology. The generation process consists of three steps:

1. *Annotation of the dataset schema* with the GQL Metamodel so that it becomes a valid instantiation of the metamodel.
2. *Automatic generation of a GraphQL schema* from the annotated ontology.
3. *Automatic generation of a GraphQL service* that exposes the generated GraphQL schema and the available data related to the annotated ontology.

The automation of step 1 is a non-trivial task that would require the definition and identification of many mappings from ontology patterns to GraphQL constructs. Nevertheless, a user familiar with the dataset (e.g., dataset publisher) requires small manual efforts to perform this task as it involves only the ontology TBox (i.e., the dataset schema) that is typically only a small part of the whole dataset (see the sequel for the details in our use case). Thus, currently step 1 is manually performed, while its automation is part of the future work.

To automate steps 2 and 3, we implemented a prototype. This prototype is a Web application with a simple interface to provide the required inputs, namely the necessary parameters to connect to the RDF triple store where the input ontology is stored. As an output, the prototype produces a GraphQL schema and a ready-to-deploy GraphQL service implementation.[3]

To illustrate the feasibility of our approach, we present a use case with the real-world dataset from the Linked Movie Database[4], which contains a total of 6,148,121 triples. This dataset contains information on 53 different concepts with a total of 222 different properties. For the purpose of the use case example, we have selected 7 concepts, namely Director, Person, Performance, Country, Film, Film_genre, and Actor, as it is shown in Fig. 4. For these concepts, we consider a total of 20 properties whose names label the arrows in Fig. 4: filmid, title, genre, etc. Notice that the range of these properties can either be a concept (e.g., Film_genre in the case of the property genre of Film) or a scalar field (e.g., String in the case of the property title of Film).

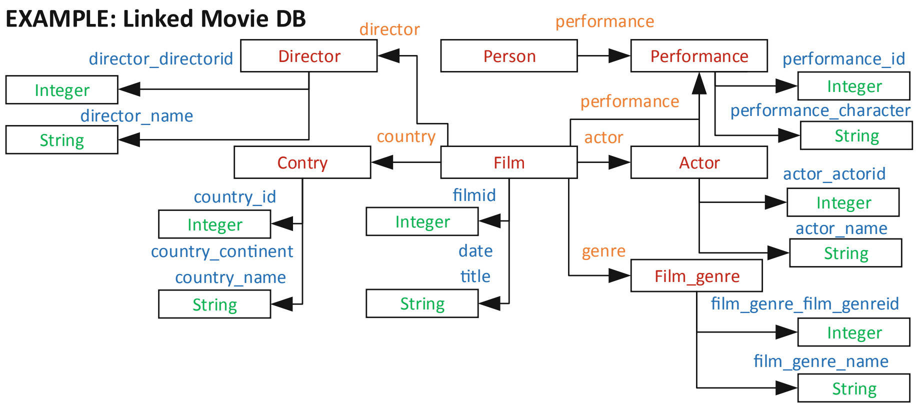


Fig. 4. Use Case dataset

4.1 Step 1: Annotation of the Dataset

The purpose of this step it to add the necessary meta-data annotations (i.e., TBox statements) in the form of extra RDF triples so that the resulting ontology is an instance of our GQL metamodel. Accordingly, for each concept of the input ontology, we should add a triple of the form: *concept* rdf:type gql:Object.

For example, Fig. 5 shows the triples added for two properties (dc:title and movie:genre) that have a film (movie:film) as their domain. Notice that the triples added in lines 1–2 and 5–7 are not part of the annotation with any GQL concept but are generic meta-data annotations defining the schema for the dataset.

[3] Prototype available at https://github.com/genesis-upc/Ontology2GraphQL.
[4] https://old.datahub.io/dataset/linkedmdb.

As this schema information was missing in the dataset, we defined it based on the data instances (i.e., ABox). The semantic enrichment specific for GQL extension is presented in lines 3–4 and 8–10 in Fig. 5, and this enriched schema information is necessary for the automation procedures that we describe below. In general, the number of triples to be added depends on the semantic quality of the original ontology (i.e., if the schema information is available) but should generally be small. In particular, for our use case, the resulting annotated ontology[5] consists of 91 new triples, where only 46 triples are related to GQL-specific annotations. The remaining 45 triples correspond to the definition of the ontology TBox (i.e., dataset schema) that was in missing.

```
1  dc:title rdf:type rdf:Property .
2  dc:title rdfs:domain movie:film .
3  dc:title rdf:type gql:ScalarField .
4  dc:title rdfs:range gql:String .
5  movie:genre rdf:type rdf:Property .
6  movie:genre rdfs:domain movie:film .
7  movie:genre rdfs:range movie:film_genre .
8  movie:genre rdf:type gql:ObjectField .
9  movie:genre gql:hasModifier ex:l2 .
10 ex:l2 rdf:type gql:List .
```

Fig. 5. Example of added triples

4.2 Step 2: Automatic Generation of a GraphQL Schema

Taking the annotated ontology, our prototype generates a GraphQL schema. For example, Fig. 1(a) depicts a fragment of the GraphQL schema generated from the annotated ontology obtained in the previous subsection.[6]

The GraphQL schema generation is fully automated following the next steps:

- For each gql:Object in the ontology, a GraphQL Type is produced.
- For each gql:ScalarField, the corresponding scalar field is produced.
- For each gql:ObjectField, the corresponding object field is produced with its modifiers (i.e., single object or list).

In addition, our prototype also deals with the following enrichments:

- **Identifiers**. The prototype adds a scalar field `idInstance` to each Type. This field is required by the GraphQL service implementation in order to properly identify each object that can be retrieved.
- **Query entry points**. As we explained in Sect. 2, GraphQL schemas must define the so-called entry points. Our prototype automatically generates a pair of entry points for each Type: one to retrieve all the instances of the type and the other one to retrieve a single instance given its identifier.

[5] https://git.io/linkedmdb.ttl.
[6] The whole GraphQL schema can be found at https://git.io/linkedmdb.graphql.

4.3 Step 3: Automatic Generation of a GraphQL Service

Our prototype is also able to produce a proof-of-concept GraphQL-service implementations for the GraphQL schemas that it generates. This is again a fully automatic procedure. These GraphQL service implementations are based on the graphql-java framework[7]. Assuming that all the data is stored in an RDF triple store such as Virtuoso, resolvers are automatically implemented in terms of SPARQL queries against the triple store. Such resolvers are functions that specify how the types, fields and entry points in the schema are connected to the data sources, and they are called at runtime to fetch the corresponding data. Our current implementation allows generating resolvers for the following GraphQL-schema elements: scalar fields, object fields, interfaces, lists of scalar fields, lists of object fields, list of interfaces, and nulls. The following elements are not yet supported: unions, input types, non-id field arguments, and enumerations.

In the context of our use case, the prototype is able to generate a GraphQL service able of answering any GraphQL query posed to the GraphQL Schema that it supports. Consequently, the resulting GraphQL service can retrieve data for 7 types with their 20 properties stored in 1,287,354 triples.

5 Related Work

Providing "a GraphQL interface for querying and serving linked data on the Web" is the aim of the tool HyperGraphQL[8]. However, in this case, the GraphQL Schema itself needs to be generated manually with additional custom (i.e., tool-specific) annotations to link the exposed data with the underlying Linked Data sources (SPARQL endpoints and or local files with RDF dumps).

GraphQL-LD [11] represents another proposal aimed at providing a bridge between GraphQL users and Linked Data publishers. In this case, the approach consists of two complementary methods: one for transforming GraphQL queries to SPARQL, and a second one for converting SPARQL results to a "GraphQL query compatible response". In the case of GraphQL-LD, no GraphQL Schema is generated nor made available to end users. In this way, one of the key foundations of the whole GraphQL approach, the GraphQL Schema, is entirely missing.

In [10] we find the so far unique proposal that has addressed the automatic generation of GraphQL schemas, to the best of our knowledge. The approach consists in defining one-to-one mappings from elements of UML Class diagrams to GraphQL schema constructs. However, in our case, the ability to have both the meta-data and the data in the same triple store allows us to generate proof-of-concept GraphQL services automatically that can expose such data.

[7] https://github.com/graphql-java/graphql-java.
[8] https://www.hypergraphql.org.

6 Conclusions and Further Work

In this paper we have presented a semantic-based approach to generate GraphQL schemas. At the core of this approach lies the definition of a semantic metamodel for GraphQL Schema. We have also shown how with some little initial effort, e.g., manually adding just 91 triples into a 6M-triple dataset, we can automatically generate a GraphQL schema and service able of retrieving the data stored in 1,287,354 triples.

As future work, we plan to advance towards the automation of the annotation of ontologies. In a broader context, we want to integrate our approach and tool in the framework described in [1] to tackle the generation and evolution of data-intensive Web APIs.

Acknowledgements. This work is funded by the Spanish project TIN2016-79269-R.

References

1. Abelló, A., Ayala, C.P., Farré, C., Gómez, C., Oriol, M., Romero, O.: A data-driven approach to improve the process of data-intensive API creation and evolution. In: Proceedings of the CAiSE-Forum-DC, pp. 1–8 (2017)
2. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: ready for action? In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 277–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_18
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. **5**(3), 1–22 (2009). https://doi.org/10.4018/jswis.2009081901
4. Cyganiak, R., et al.: Resource description framework (RDF): Concepts and abstract syntax (2014). http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
5. Daga, E., Panziera, L., Pedrinaci, C.: A basilar approach for building web APIs on top of SPARQL endpoints. In: Third Workshop on Services and Applications over Linked APIs and Data, pp. 22–32 (2015)
6. Facebook Inc: GraphQL, June 2018. http://facebook.github.io/graphql
7. Groth, P.T., Loizou, A., Gray, A.J.G., Goble, C.A., Harland, L., Pettifer, S.: Api-centric linked data integration: the open PHACTS discovery platform case study. J. Web Sem. **29**, 12–18 (2014). https://doi.org/10.1016/j.websem.2014.03.003
8. Meroño-Peñuela, A., Hoekstra, R.: Automatic query-centric API for routine access to linked data. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 334–349. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_30
9. Nadal, S., Abelló, A.: Integration-oriented ontology. In: Encyclopedia of Big Data Technologies, pp. 1–5. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-63962-8_13-1
10. Rodriguez-Echeverria, R., Cánovas Izquierdo, J.L., Cabot, J.: Towards a UML and IFML mapping to GraphQL. In: Garrigós, I., Wimmer, M. (eds.) ICWE 2017. LNCS, vol. 10544, pp. 149–155. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74433-9_13
11. Taelman, R., Sande, M.V., Verborgh, R.: Graphql-ld: linked data querying with graphql. In: ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks (2018). http://ceur-ws.org/Vol-2180/paper-65.pdf

12. Varga, J., Romero, O., Pedersen, T.B., Thomsen, C.: Analytical metadata modeling for next generation BI systems. J. Syst. Softw. **144**, 240–254 (2018). https://doi.org/10.1016/j.jss.2018.06.039
13. Varga, J., Vaisman, A.A., Romero, O., Etcheverry, L., Pedersen, T.B., Thomsen, C.: Dimensional enrichment of statistical linked open data. J. Web Sem. **40**, 22–51 (2016). https://doi.org/10.1016/j.websem.2016.07.003

Digital Dices: Towards the Integration of Cyber-Physical Systems Merging the Web of Things and Microservices

Manel Mena(✉), Javier Criado, Luis Iribarne, and Antonio Corral

Applied Computing Group (TIC-211), University of Almería, Almería, Spain
{manel.mena,javi.criado,luis.iribarne,acorral}@ual.es

Abstract. One of the main issues of devices and platforms related to Internet of Things (IoT) is that there exists a broad spectrum of different protocols addressing those devices. Management and connection to those things create integrability and usability issues. Hence, there is a need for a solution that facilitates the communication between different devices and platforms. The Web of Things (WoT) tries to handle interoperability issues by describing interfaces and interaction patterns among things. Thanks to the models proposed by the WoT, it is possible to decouple the description of things from the protocols handling the communication and implementation strategies. This article proposes Digital Dice as an abstraction of IoT devices inspired by the concept of Digital Twin, but capable of leveraging the advantages of microservices architectures. We focus on the creation of Digital Dices from WoT models. A Digital Dice consists in different facets that are able to handle a particular aspect of a thing, hence different WoT descriptions models will result in different microservices related to that particular thing. An architecture to handle multiple Digital Dices and their replicas is also proposed.

Keywords: Cyber-physical systems · IoT · Microservices · Web of Things · Digital Twins

1 Introduction

Different protocols related to the Internet of Things (IoT) must be taken into account to establish an ecosystem of devices capable of handling different aspects. Those protocols are usually divided into different layers [1], and their great number, heterogeneity and use of different technologies brings about the problem of interoperability between devices and/or platforms.

Related to the interoperability, there are solutions that try to support the integration of IoT devices in Smart Space environments [2], such as Node-RED, Eclipse Kura, Prodea, etc. These approaches are designed to control a limited

This work was funded by the EU ERDF and the Spanish MINECO under Project TIN2017-83964-R. Manel Mena has been funded by a Grant of the Spanish Government FPU17/02010.

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 195–205, 2019.
https://doi.org/10.1007/978-3-030-32065-2_14

number of devices, and are build as monolithic applications, so they lack a good scalability. Another common problem is to have bottlenecks in the communication due to the restrictions on most IoT devices (*e.g.*, low energy consumption, low computing power). In addition, there is a need to virtualize this type of devices for the purpose of carrying out tests without influencing the business processes [3]. To solve this last problem, the concept of Digital Twin (DT) was introduced as a virtual representation of a physical element, system or device, which can be used for different purposes [4]. However, this concept focuses on a monolithic approach for the management and representation of devices, and lacks of multiple levels that specifically addresses each facet of a device.

In this paper, we introduce the concept of Digital Dice (DD), an abstraction of IoT devices or cyber-physical systems capable of leveraging the advantages of microservices architectures. Furthermore, we explain how Digital Dices are closely related to the Web of Things (WoT) framework [5]. The WoT was created to define a common definition schema for Things. In particular, our proposal makes use of the Thing Description (TD) defined by the WoT. The Thing Description offers an overview of the features of a *thing*, as well as the endpoints to request those features. The TD gives a solution to the problem of feature discovery on *things*. The Digital Dices are a software abstraction of *things* that make use of TD to establish a connection to the thing as well as generate another Thing Description with new endpoints, offering a common communication protocol for the final user no matter the protocols used internally by the IoT device or the Cyber-physical System. Besides, we study the conversion between Things Description models and the different facets that are used by our DD.

The main contributions of Digital Dices are (*i*) the definition of a common communication protocol to control all kind of devices, a behaviour capable of lessening the number of requests on a IoT device; (*ii*) the capability of handling numerous request just replicating the facets needed; and (*iii*) the possibility using our Digital Dice as a Thing in other systems compatible with WoT.

This paper is structured as follows. In Sect. 2 we describe our Digital Dices as a solution to solve the problems described above. In Sect. 3 we study the Thing Description model [5] and how to convert Thing Descriptions to Digital Dices. Section 4 describes an example scenario. Section 5 offers a overview of related work, solutions and systems found in the literature. Finally, Sect. 6 describes the advantages of our Digital Dices and propose the future work.

2 Background

Digital Dices, like Digital Twins, are virtual representations of physical elements, but they offer several improvements. A digital dice proposes a virtual abstraction of IoT devices, cyber-physical systems or other virtual devices that is based on microservices, and aims to be agnostic to the protocols used by said devices. This DD should be compatible with the Web of Things framework, specifically the Thing Description model. Thanks to this fact, our digital dices can be totally compatible with different systems that make use of the standard, like Mozilla Web Thing framework [6].

The concept of *dice* (multiple faces) is given by how the microservices that represent our devices are characterized. These microservices oversee different aspects of the devices as follows:

(a) *Controller.* This microservice handles the communication with the user, and it is the one that manages the orchestration with the rest of the facets or directly to the IoT device as need.
(b) *Data Handler.* This facet is the one that handles the communication with the underlying database. This database has two main functions; it can log different requests done by the user so we can trace possible problems originated in our Digital Dices, and it can act as a buffer for the IoT device. This is done as follows. First, it tries to recover the requested data from the database before obtaining it from the physical device; if the data requested is newer than the time threshold configured (by default 10 s) by our DD, then this data will sent as response.
(c) *Event Handler.* This aspect processes the events generated in the IoT device. At the same time, it also will manage the connection with a future possible Complex Event Processing (CEP) subsystem [7].
(d) *User Inteface (UI).* The user interface established a micro-frontend for each feature controlled by our DD. The features can offer a UI that will be declared in the TD model that supersedes our DD. Besides those individual interfaces for each feature, we have a method in our UI microservice that can make a composition of all the individual features that have a UI. We call it the global UI of our Digital Dice. This approach provides the user with a reusable UI to interact with the device.

It is important to note that not all the facets will be always part of a DD. Furthermore, these facets can be extended, for instance, to include a Voice Activated Interface or an Open Data Handler capable of proactively send data dumps of a time frame worth of information into an Open Data system.

The connection of facets with IoT devices is one of the main problems that must be addressed. With this aim, a library capable of managing multiple protocols is required. Moreover, we need to establish what microservices have a direct connection with the IoT devices. To that end, we classified them in: (a) Hard Related Facets, with a direct connection with the IoT device (*e.g.*, Data Handler, Controller, Event Handler), and (b) Soft Related Facets, representing those that do not establish a direct connection with the IoT device (*e.g.*, UI).

The facets of the Digital Dices establish communication with the users following the standards, mechanisms and technologies by the W3C. One of the challenges that we face is to establish a system architecture for Digital Dices. This architecture has to be capable of sustaining multiple copies of the same microservices, having load balancing between them and detecting when a microservice is being over used so it can start a replica of it.

The microservice architecture [8] that we propose for the management of our Digital Dices is shown in the Fig. 1. It establishes a possible configuration proposed for the architecture. First, the `Edge` layer is composed basically by two

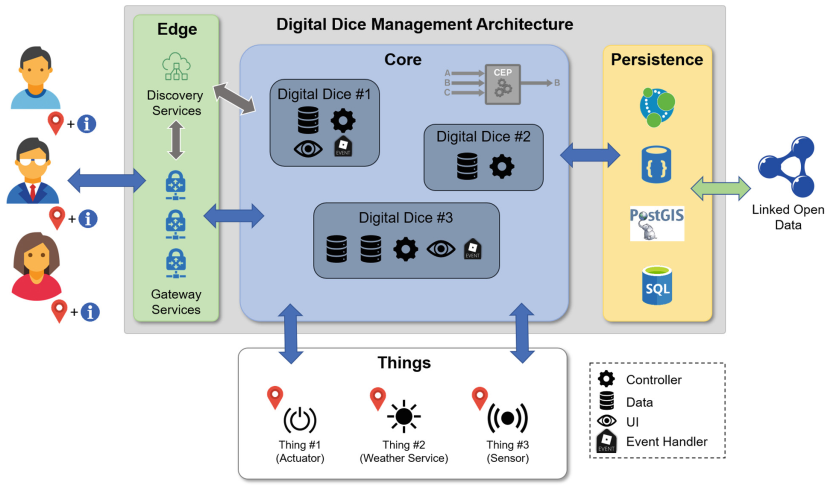


Fig. 1. Digital Dice architecture

types of microservices: (a) Gateways that will be in charge of redirecting the relevant requests to our Digital Dices, and (b) Discovery services, that keeps a register of each microservice contained in our Digital Dice Architecture. The Discovery Service allows our Gateways to redirect the request to the appropriate instance when necessary, allowing load balancing between the different replicas of said instances. Besides the *Edge*, the architecture has the *Core* of our system, which is where our Digital Dices and a series of auxiliary microservices are framed, such as authentication services or microservices for CEP. In addition, we will have the *Persistence* layer, which is composed of databases and possible services associated with them, for example different Open Data services. The *Things* layer of the architecture represents the physical devices associated with our Digital Dices, as well as external services, virtual components, etc.

Figure 1 illustrates the management of three Digital Dices. The DD #1 has the four facets described and it is responsible of handling an actuator to turn a switch on and off. The DD #2 only has two facets active because the interaction with the climatological information service is done through the Data Handler and Controller facets. The DD #3 has a duplicate Data Handler facet because the threshold number of accesses has been exceeded.

3 Thing Description to Digital Dice

In this section we describe how the conversion between a Thing Description (TD) and a Digital Dice (DD) is handled. First, we need to describe the formal model and a common representation for a TD, that can be considered as the entry point of a *thing*. This model is used as a centerpiece of our proposal as it helps us to define the actions, properties and events managed by our DD. At the same time, this model can be used as a starting point to generate a DD semi-automatically, by applying Model-to-Text transformation [9].

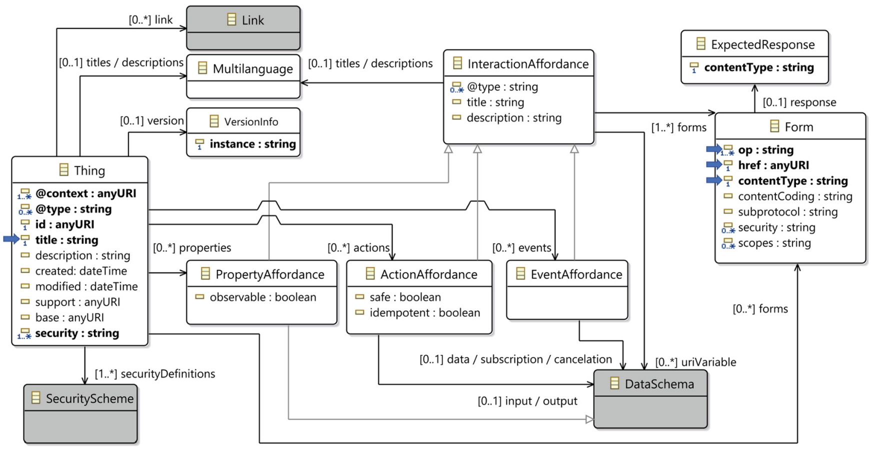


Fig. 2. Thing Description Model

Figure 2 shows an overview of the TD model. A *thing* is made up by the *properties*, *actions* and *events*. All the features defined by the TD are subclasses of *interaction affordance*, and it is made up by one or more *forms*. These forms will help us to define the methods that compose the DD generated by the TD. Figure 2 shows three fields in the form model that are mandatory, those three fields will be used to create our DD. The different possible values of each field can be consulted in the WoT TD definition [5]. Sometimes, those fields are omitted, that just means that they get their default value:

(*a*) `op` (operation) field has an array of string with the elements *readproperty* and *writeproperty*, if the feature is a `PropertyAffordance`. Furthermore, it will be an *invokeaction*, if it is an instance of `ActionAffordance`. *Subscribeevent* will be used as a default if it is an instance of `EventAffordance`;
(*b*) `contentType` field has as default value *application/json*.

The URIs generated in our DD will follow the pattern `https://{ip-address}:{port}/{thing.name}/{property|action|event}/{IntAffordance.title}/`. Besides the features defined in our DD, we have to offer an URI with the Thing Description Model that defines our DD, this can be accessed through `https://{ip-address}:{port}/{thing.name}/`.

Figure 3 describes the microservices or facets generated by the conversion of a TD into a DD. Properties always define a controller microservice, but the data handler will only be created if the property contains the operation *writeproperty*. In the same figure, we can depict how actions generate two microservices, a controller and a data handler. Nevertheless, if the *thing* has already a property created, it will only add the necessary methods to the corresponding microservices. In the case of the events, a controller and an event handler will be created.

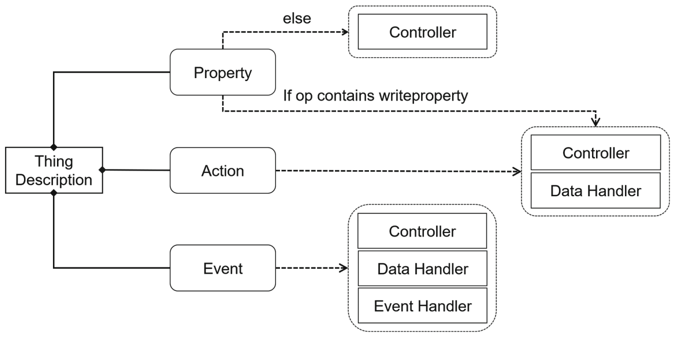


Fig. 3. Thing Description to Digital Dice.

As we explained in Sect. 2, our Digital Dice has another facet, the UI, that will be generated as a microservice. This microservice will be only generated if a property, action or event contains in `contentType` the parameter `ui` = `true`. This flag establishes that the DD has to create a visualization based on the data type of such feature. The visualization will be a micro-frontend, generated as an <`iframe`>, <`portal`> or <`component`> form. If our DD has one or more UI components (Fig. 4), a new method will be created in the UI microservice. This method will show a composition of all the generated components, and it will appear as a link in the TD that represents our DD.

Figure 4 shows how a Thing Description defines different actions, properties and events, and how those features can have associated different micro-frontend components. In this case, properties pi and p2, and action a3 have associated

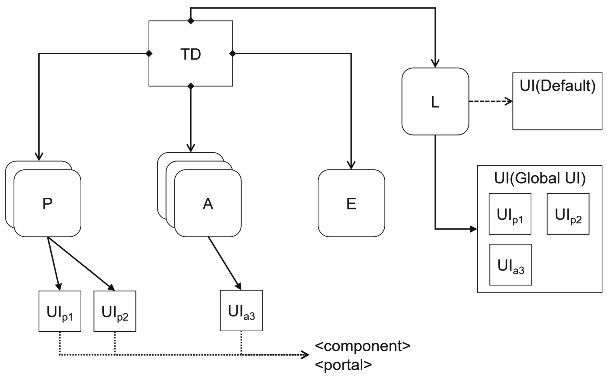


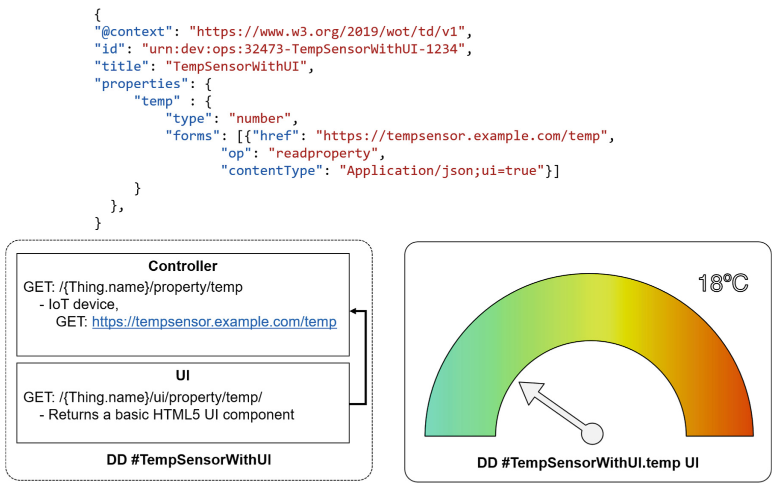
Fig. 4. Global UI composition.

UIs. Besides those three UIs, a global one will be generated as a link on the Thing Description with a composition of every UI component.

4 Example Scenario

In order to evaluate or approach, we propose an example scenario to transform a thing description into a digital dice. Figure 5 shows a temperature sensor represented by a TD and the respective microservices and methods created after the translation. The thing description represents a device with one property `temp` (number type) and contains a form with the address for obtaining the value from the physical device. The field `op` is established as *readproperty*, that means that only a controller will be created (because it is not *writeproperty*). Furthermore, the `contentType` has the parameter `ui` = `true`, which represents that our digital dice needs a UI microservice for the property.

As we can see in Fig. 5, two microservices are created. In the controller, a GET method responding with a simple number for the `temp` is deployed. The fact of being a GET method is because *readproperty* is bound to a GET method. The `temp` method builds a response with the data recovered from the IoT device. The UI microservice will make use of the same method of the controller to create a visual component that shows the value of `temp`. Since our DD has a feature with an UI, a composition with it will be created as a *link* in the TD of our DD. It should be stressed that the thing description depicting our digital dice does not has to be the same as the one from the original TD. It will probably have the same number of features, but with different `href` (the ones from our DD) and data related to the global UI composed by the declared UI features.



```
{
"@context": "https://www.w3.org/2019/wot/td/v1",
"id": "urn:dev:ops:32473-TempSensorWithUI-1234",
"title": "TempSensorWithUI",
"properties": {
     "temp" : {
         "type": "number",
         "forms": [{"href": "https://tempsensor.example.com/temp",
                "op": "readproperty",
                "contentType": "Application/json;ui=true"}]
     }
   },
}
```

Controller
GET: /{Thing.name}/property/temp
- IoT device,
GET: https://tempsensor.example.com/temp
UI
GET: /{Thing.name}/ui/property/temp/
- Returns a basic HTML5 UI component
DD #TempSensorWithUI
18°C
DD #TempSensorWithUI.temp UI

Fig. 5. TD to DD - TempSensorWithUI

Listing 1.1 shows the code generated for the Controller microservice. We use `Java` even though the process can be extended to any other programming language. The code utilizes of a set of properties (lines 3 and 5), first the `env` variable has the configuration of the microservice, with parameters such as the Thing Description of the represented device, ports, Discovery Service address, and parameters related to the database connection, among others. Secondly the `propertiesMongoRepository` (line 5) includes all the properties found in the related device, in this case the temperature. Furthermore, we can see the two methods generated (lines 7–14 and lines 15–22) following the specification defined in Sect. 3. The first method returns the TD managed by the DD. The second method recovers the temperature and generate a response with the default content type (`application/json`) for the user. The other microservice will make use of this last method to generate the UI.

```
1   public class Controller {
2   @Autowired
3   private Environment env;
4   @Autowired
5   private PropertyRepository propertiesMongoRepository;
6   @GetMapping("/TempSensorWithUI",produces="application/json")
7   public ResponseEntity returnTD() {
8       try {
9           String td=env.getProperty("td");
10          return new ResponseEntity(td, HttpStatus.OK);
11      } catch (Exception e) {
12          return new ResponseEntity(e, HttpStatus.INTERNAL_SERVER_ERROR);
13      }
14  }
15  @GetMapping("/TempSensorWithUI/property/temp",produces="application/json")
16  public ResponseEntity getPropertyTemp() {
17      try {
18          PropertyData temp=propertiesMongoRepository.findPropertyTemp();
19          return new ResponseEntity(temp, HttpStatus.OK);
20      } catch (Exception e) {
21          return new ResponseEntity(e, HttpStatus.INTERNAL_SERVER_ERROR);
22  }...
```

Listing 1.1. Controller generated code (TempSensorWithUI)

5 Related Works

Linking IoT devices with the Web is not a new idea. Authors like Guinard et al. [10] are working actively in make this a fact. They propose a Web of Things architecture and best-practices based on the RESTful principles. Our approach of DD tries to go a step further leveraging the latest trends in the Web Services architecture, *i.e.*, the use of microservices as a building block of our solution. As we explained, Digital Dices are closely related to the WoT [5], being this a reference framework that seeks to counter the gap found in the IoT world. The idea of using the WoT comes from the need of making our DD concept compatible with other systems and software, such as Mozilla WebThings [6], which offers a unifying application layer and links together multiple underlying IoT protocols using existing web technologies. The Digital Dices can be used as virtual devices in said system.

More closely related to our approach, the authors in [11] propose a solution based on Jolie and Java programming languages to manage a prototype platform supporting multiple concurrent applications for smart buildings. This proposal uses an advanced sensor network as well as a distributed microservices architecture. The solution has the caveat of being focused on a specific domain thus not really giving a broad solution to the management different devices.

Other proposals such as [12] and [13] offer a general solution to the use of microservices in a non-domain specific approach. The solution proposed by [12] makes use of 8 different microservices to separate aspects of an IoT centric systems, such as security, events, devices, etc. But, from our point of view, this solution does not really take advantage of the power of microservices. In contrast, the Digital Dice Architecture is a more fine grained solution, because our proposal handles each aspect of a device independently as a microservice.

At a higher level, Niflheim approach [14] proposes an end-to-end middleware for a cross-tier IoT infrastructure that provides modular microservice-based orchestration of applications to enable efficient use of IoT infrastructure resources. We see Niflheim as a complementary system, since it could provide a reliable architecture for the deployment of our Digital Dice Architecture. The IoT-A project [15] is an Architectural Reference Model (ARM) that establishes a common understanding of the IoT domain and provides to developers a common technical foundation and a set of guidelines for deriving a concrete IoT architecture. This ARM is taken into account in both the WoT and the Digital Dice architectures to establish the communication between different sets of devices.

6 Conclusions and Future Work

The aim of this proposal is to improve interoperability, integration and management between both real and virtual IoT systems and devices. To do this, the functionality of IoT devices will be abstracted to a set of microservices making use of the standards set by the WoT.

The use of microservices architectures allows us to establish choreography mechanisms that take into account the requirements of the system, permits a better use of resources and facilitates the maintenance. This article offers a solution that let us establish a way to convert external IoT devices described as a WoT Thing Description into Digital Dices. At the same time, Digital Dices offer a Thing Description themselves so they can be used by external systems seamlessly. The example scenario was designed to understand conversion process.

There are multiple advantages when using Digital Dices. Being a software abstraction let us define a common communication language no matter which device our Digital Dice is representing, thus defining a common pattern to connect to features defined by said device. The internal behaviour of our Digital Dice lessens the number of request received by the device, in some cases there is actually no need for us to connect with the device when we want to recover a property value. Our system, by definition, is based on microservices, this allows us to replicate only the facets of the system that receive more requests, this give our system flexibility, thus always trying to minimize the use of resources. Another advantages is the compatibility of our solution with the WoT definition

Schema, this offers other systems like the Mozilla IoT Gateway the possibility of making use of the schema defined by our Digital Dice to interact with.

The main disadvantages of using Digital Dices are mainly inherent to the use of microservices. First, the architectural complexity that microservices usually requires. It is easier to develop a monolithic application than a software based in microservices. Furthermore this kind of software requires outside gateways, discovery and other auxiliary software to choreograph the communication inside our system. Besides this fact, in some circumstances DD can be slower to respond than direct connection to IoT devices but usually more reliable.

For future work, we want to develop Digital Dices with different programming languages so we can analyze which ones offer a better performance. Comparing the Digital Dice performance and reliability with IoT devices and other software solutions is also one of the next steps of our further work. The possible extension of the WoT Thing description model to better suit our concept of Digital Dice is our next research objective, especially trying to define complex events in the model and keeping the compatibility of the Thing Description of W3C.

References

1. Postscapes: IoT Standards and Protocols. https://bit.ly/2iAWbky. Accessed 24 May 2019
2. Ngu, A., Gutierrez, M., Metsis, V., Nepal, S., Sheng, Q.: IoT middleware: a survey on issues and enabling technologies. IEEE Internet Things J. **4**(1), 1–20 (2016)
3. Shetty S.: How to Use Digital Twins in Your IoT Strategy. https://gtnr.it/2FFU4al. Accessed 24 May 2019
4. Tuegel, E., Ingraffea, A., Eason, T., Spottswood, M.: Reengineering aircraft structural life prediction using a digital twin. Int. J. Aerosp. Eng. **2011**, 14 p, (2011). Article ID 154798
5. W3C: Web of Things. https://www.w3.org/WoT/. Accessed 28 May 2019
6. Mozilla Foundation: Mozilla IoT Web of Things. https://iot.mozilla.org/. Accessed 28 May 2019
7. Angsuchotmetee, C., Chbeir, R.: A survey on complex event definition languages in multimedia sensor networks. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems, pp. 99–108. ACM (2016)
8. Nadareishvili, I., et al.: Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media Inc., Sebastopol (2016)
9. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Syst. J. **45**(3), 621–645 (2006)
10. Guinard, D., Trifa, V.: Building the Web of Things: With Examples in Node.js and Raspberry Pi. Manning Publications Co., New York (2016)
11. Khanda, K., Salikhov, D., Gusmanov, K., Mazzara, M., Mavridis, N.: Microservice-based IoT for smart buildings. In: 31st International Conference on Advanced Information Networking and Applications Workshops, pp. 302–308. IEEE (2017)
12. Long, S., Yan, L., Memon, R.H.: An open IoT framework based on microservices architecture. China Commun. **14**(2), 154–162 (2017)
13. Vresk, T., Čavrak, I.: Architecture of an interoperable IoT platform based on microservices. In: 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, pp. 1196–1201 (2016)

14. Small, N., Akkermans, S., Joosen, W., Hughes, D.: Niflheim: an end-to-end middleware for applications on a multi-tier IoT infrastructure. In: IEEE 16th International Symposium on Network Computing and Applications (NCA), pp. 1–8. IEEE (2017)
15. Bauer, M., et al.: Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0 (2013)

A Smart Living Framework: Towards Analyzing Security in Smart Rooms

Walid Miloud Dahmane[1](✉), Samir Ouchani[2](✉), and Hafida Bouarfa[1](✉)

[1] Computer Science Department, Saad Dahlab University, Blida, Algeria
walid.miloud.dahmane@gmail.com, hafidabouarfa@hotmail.com
[2] LINEACT, École d'Ingénieur CESI, 13545 Aix-en-Provence, France
souchani@cesi.fr

Abstract. Indoor environments play a main role on people living, work, social life, and health. Especially, recent statistics show that people who are often most susceptible to the adverse health effects tend to spend more time indoors. To overcome these issues, modern rooms are well equipped with different kind of connected objects (IoT) in order to facilitate the life of inhabitants and to manage better the indoor environment by automatically controlling the indoors problems (humidity, temperature, noise, light ...) with respect to the experts recommendations and inhabitants hopes. So, the quality of indoor living depends mainly on the environments properties, their precise measurements through IoT, and how safely manage and control them. In this context, we propose a smart living framework covering a global architecture of smart rooms and their related components (sensors, actuator, server ...), and also a management system for a secure communication network respecting the existing safety and security standards. Further, we present the appropriate semantics for each part of the framework in the form of state machines. Further, we ensure the functional correctness of the proposed framework by implementing the developed state machines in the validation and verification tool Uppaal, where it gives us a satisfactory result for different scenarios.

Keywords: Smart environment · Indoor living · Smart room · IoT · UPPAAL · Simulation

1 Introduction

For a better living quality, the smart spaces paradigm aims at constructing advanced service infrastructures that follow the ubiquitous computing approaches where smart objects are executed on a variety of digital devices and services are constructed as interaction of agents in a communication environment [1]. The main feature of this technology is the integration of heterogeneous and action elements (actuators) in a distributed system which performs different actions based on the information gathered by the sensors combined with the requirements of the particular application. Intelligent information systems enable the processing of multimodal data collected by the sensors, so as to reconcile heterogeneous information and safe conclusions on the facts giving rise to the activation of the necessary actions to address the consequences of these events [2].

The room has several factors that can affect it or the life of inhabitants or both at the same time (temperature, humidity, noise, light, *etc*). Nowadays different numerical models are available to describe the vapor balance of transient water in a room and predict indoor humidity [3]. In general, sensors communicate directly with the home gateway and feed the system information with regards to the obtained environment measures, for example light intensity inside a particular room, temperature inside and outside the home and motion sensing to name a few [4].

In this paper, we propose a smart living framework by modeling the different components needed for an indoor environment and developing a trustworthy architecture that ensure the well functioning correctness of such system, and also its configuration and control. First, we rely on the existing limitations and the requirements for a room that can affect the inhabitant like humidity, the temperature, loud noise, the challenges of handicapped, dangerous natural and artificial phenomena such as earthquake and fire.

The proposed framework is a web service based solution where sensitive nodes are indoor planted and their measures change in real time. The architecture proposed for the framework considers different classes of nodes. A database node containing the collected data by sensors, a server node that ensures the communication and the reliability between nodes, and reacts when necessary by sending the appropriate control commands; the actuator node executes the received commands from server (actuator) and external actors who can extract or edit room data. The architecture uses MQTT protocol [5] to ensure a reliable communication between the predefined internal nodes. Further, the architecture implements a precise constraints and requirements for the communication and during executing actions. Otherwise, the nodes do not respecting certain conditions are considered as Unacceptable nodes. Finally we ensure the functional correctness of the nodes and their safe communication by simulation in, the verification and validation tool, Uppaal [6] by creating different scenarios. The results show that the proposed framework is a deadlock free and respecting the indoor living requirements.

Outlines. The remainder of this paper is organized as follows. Section 2 presents the related work and compares it with the proposed framework detailed in Sect. 3. Then, the implementation with the experimental results are shown in Sect. 4. Finally, Sect. 5 concludes this paper and provides hints on the possible future works.

2 Related Work

In literature, we review the existing work related to IoT modeling, functional analysis, network architectures, and application in real life with concrete cases.

Ouchani *et al.* [7] proposes a security analysis framework for IoT that covers the probability and costs of actions, formalizes IoT, analyzes the correctness and measures their security level by relying on the probabilistic model checking PRISM. To ensure the functional correctness of an IoT-based system, Ouchani develops five steps: defines the IoT components, formalizes the architecture in a process algebra expression. Then, it expresses the IoT requirements in PCTL and transforms the IoT model into the PRISM input language. Finally, PRISM checks how much a requirement is ensured on the IoT model. However, the proposed framework involves a large amount of data and messages which make the probabilistic model checking expensive in terms of time and memory.

Moreno-Salinas *et al.* [8] proposes a method that detects the optimal position of sensors to receive information from several targets. To find the perfect place, they rely on FIM[1] to measure the amount of information that a random variable (sensor) carries about a parameter that is sometimes unknown (target). After several progressive tests, they use two separated tests, the first tries to find the optimal position for a sensors that receives from a target transmitter with a known placement. The second one finds the optimal positions of sensors with unknowns positions. However, FIM showed significant results for a small amount of objects but the cost of calculation time is expensive when the target is unknown in a known area.

Centenaro *et al.* [9] studies a start topology of LPWANs[2] in smart cities where the used network $LoRa^{\text{TM}}$. The goal is to know the number of gateways needed to cover the city (inexpensive or not), and to know the benefits in return after deployment. They experimented two tests, the first installs $LoRa^{\text{TM}}$ network in a 19-history building to measure building temperature and humidity, using one single gateway and 32 nodes. The second estimates the number of gateways required to cover the city of Padova. They placed a gateway with no antenna gain at the top of two storey buildings to assess the 'worst case' coverage of the topology, $LoRa^{\text{TM}}$ technology allows to cover a cell of about 2 km of radius. With simple calculations they concluded that to cover Padova city witch has about $100\,\text{km}^2$, it needs to 30 gateways. At present, $LoRa^{\text{TM}}$ has acceptable coverage in worst cases, but the number of gateway ports is limited and does not satisfy progressive evolution of IoT technology.

Sanchez *et al.* [10] try to rend Santander city (Spain) smart by controlling air quality, noise, temperature, luminosity, irrigation monitoring and environmental station. They build an architecture based on three levels: the IoT object level encloses IoT peripherals like IoT sensors and APIs, the gateway level, and the IoT server level related to cloud services. They did experiments to organize the irrigation of plants by measuring temperature and humidity of soil. The architecture allows users to monitor and control their resources using OTAP technology since the proposed solution is not a wired.

Zanella *et al.* [11] apply the principles of smart city for Padova city to collect environmental data. The architecture is composed of constrained IoT

---

[1] Fisher information matrix.

[2] LowPower Wide Area Networks.

sensors, a database server which use technologies CoAP[3], 6LoWPAN[4], unconstrained devices that use traditional technologies like HTML. The interconnection between users and sensors is made by an intermediary gateway and HTTP-CoAP proxy-grown that plays the role of translator between the two sides. During a week of tests, the results show how do people react with different situations and phenomena, for example benzene consumption at the end of weeks. This architecture allows the compatibility between constrained and unconstrained devices by a cross proxy. In general, the constrained physical and link layer technologies are characterized by a low energy consumption, the transfer rate and data processing in constrained devices is relatively low, but the dependence on unconstrained ones increase in cost.

Based on the reviewed literature, we found few works that detail well the components of an indoor environment and their formal semantics, and less of them discussing a trustworthy communication between components. The proposed contribution considers these issues and we believe it is easy to extend.

3 Framework

This section details the proposed secure network and communication system for smart rooms. First, we present the overall system architecture and the components related to the system, then we detail the semantic of each used object and node, Finally we describe the communication protocol and the data management in the proposed system.

3.1 Architecture

Figure 1 depicts the main components of the proposed architecture, which is based on three levels, detection, analysis, and action level. The detection level allows to sense the status of a room in real time then it makes a declaration in case a contradictory status (fire, noise, humidity, etc.), the nodes of this level are mainly the sensors. The analysis level has nodes that import the data (input data) to analyze them then extract the commands (output data). Nodes of this level can be either: web server, broker, database, and smartphone. The action level contains actuators that execute the physical actions according to the received commands form analysis level.

3.2 Smart Objects

An object can be defined by its static attributes and dynamic behavior. The static attributes can be: identification [12], connectivity [13], battery life [14], powered by electricity, data security [15], small size, high product quality, constrained device [16], price [17], service availability [18], minimum error [19], easy

[3] Constrained Application Protocol.

[4] IPv6 Low power Wireless Personal Area Networks.

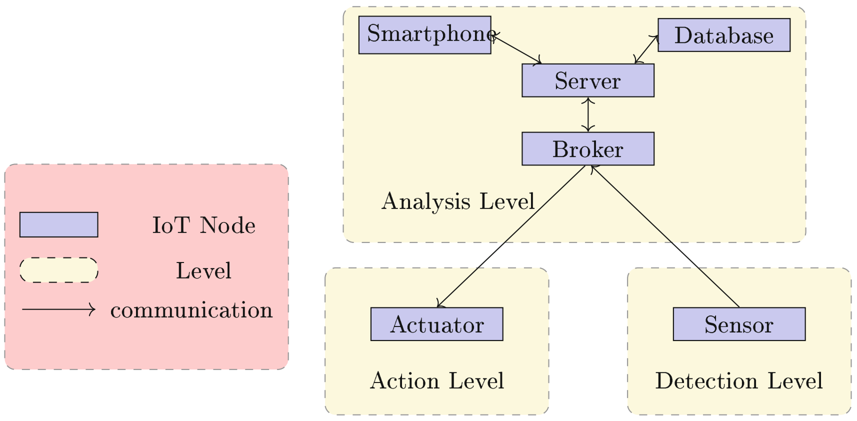


Fig. 1. Architecture of smart room.

to maintain, required a low connection rate [20], interoperability of nodes [21]. The dynamic defines its behavior that relies on its proper actions, mainly: turn on [22], turn off [22], send [23], receive [23], collect data [23], apply action [23], encrypt, decrypt, and authenticate. Definition 1 defines formally a general smart node can be a Sensor, Actuator, Broker, Database, Server or Smartphone.

Definition 1 (Smart node).
A smart node is a tuple of $\langle O, Prop_o, Fonc_o, Behav_o \rangle$, *where:*

1. *O is a finite set of IoT objects written in the form $\{O_i \mid i \in N\}$ where $o_\varnothing \in O$ is an empty object.*
2. *$Prop_o$: O: $\rightarrow 2^A$ is a function returning an object properties where $A = \{$Id, Co, BLi, PEl, DSe, SSi, HPr, CDe, LPr, SAv, MEr, EMa, RLo, INo$\}$ that precise respectively: identification, connectivity, battery life, powered by electricity, data security, small size, high product quality, constrained device, low price, service availability all the time, minimum error, easy to maintain, required a low connection rate, interoperability of nodes*
3. *$Fonc_o$ are the set of functionalities/actions of objects, where $fonc_o = \{turn\_on_o$, $turn\_off_o$, $send_o(O_i, O_j)$, $receive_o(O_i, O_j)$, $collect\_data_o$, $apply\_action_o$, $consume\_energy_o$, $encrypt_o$, $decrypt_o$, $authenticate_o(O_i, O_j)$ où $O_i, O_j \in O\}$. $turn\_on_o$ and $turn\_off_o$ to turn on or turn off the object, $send_o(O_i, O_j)$ et $receive_o(O_i, O_j)$ to send or receive the information from O_i to O_j, $collect\_data_o$ to collect the received information, $apply\_action_o$ to apply an action after getting command, $consume\_energy_o$ the ability to raise the energy level, $encypt_o$ and $decrypt_o$ encrypt or decrypt the message, $authenticate_o(O_i, O_j)$ the object O_i authenticate in the object O_j.*
4. *$Behav_o$: $O \rightarrow E_o$ returns the expression E_o that defines the behavior of an object in the dominant case; where: E_o =Start.Action.End; where Action = $Fonc_i | Fonc_i.Action$, $i \in N$*

3.3 Measurements

We describe here a selection of natural measurements from others that we took into consideration.

- Light: if the noise level measured by the noise sensor in the room reaches a high limit and the room has a low light level as at night, the lights automatically turn on, this case solves the problem of crying the children in the room. In another case, if an inhabitant wants to light a room, the sensor of the light placed on the outer face of the window senses the degree of the sunlight, and if it is enough, the windows will be opened with a turn off of the lamps. This action helps in saving energy.
- intelligent doors and windows: persons with reduced mobility that move by a trolley find it difficult to open the door, so a detector is placed on the door in order to detect the patient trolley. Also other vibration sensors are placed on the wall to detect the earthquake, and if the level is strong, the actuators receive commands that allow the opening of doors and windows to facilitate the exit and to decrease the pressure which can cause a burst of glass, and the another actuators cut the electricity.
- Temperature: to control the energy, the room must contain two temperature sensors, indoor and outdoor. If the temperature service in the server receives the air information from the sensors, the server sends a command to the air conditioner to adjust the temperature level or turn off.
- Humidity: the humidity sensors are placed on the room walls and periodically they measure the level of humidity. If it exceeds the required limit, the sensors declare the humidity service which informs the resident by email for this case, then it gives orders to the actuators to open the windows of the room, and turn on a fan for air circulation in the room.
- Fire: fire sensors trigger automatically the case of fire by measuring the proportion of smoke, and it sends a signal to the fire service in the server, which send commands to the actuator to open doors and windows, spraying water, and the owner of the house and firefighters receive an alert message.

3.4 Communication Protocols

The communication between client-server nodes is based on two protocols: MQTT and HTTP. The former, a publish-subscribe mail protocol, is used when sensors and actuators are clients; and the latter is applied for other clients like smartphones which is based on Internet. Figure 2 shows the whole communication between nodes, where the main steps are described as follows.

1. A sensor *publish* the data to the broker.
2. The database *subscribes* into the Broker in order to periodically keep track of the retrieved data.
3. The server *subscribes* in the Broker and *receives* the published sensors data.
4. The web server, including smart applications, *presents* the appropriate command, and *pulls* it onto the MQTT Broker.

5. The actuators *subscribe* in the Broker then it *receive* and *execute* the commands.
6. The application *retrieves* or *updates* the database values.
7. External actors, through web and smart applications, communicate securely with web server by encryption method like RSA.

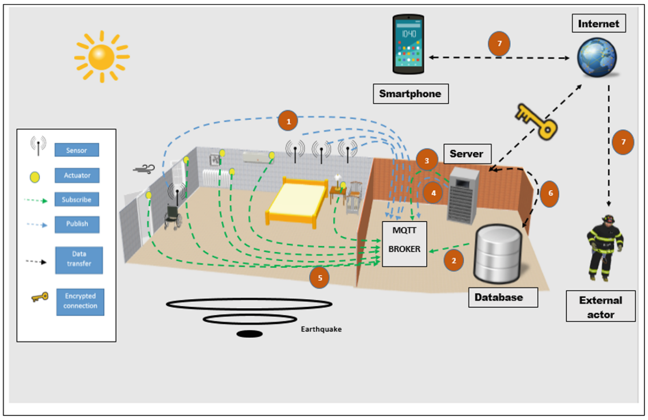


Fig. 2. The communication steps.

4 Experimental Results

In this section, we show the effectiveness of the proposed framework on two real cases scenarios. As mentioned, we use Uppaal, an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata extended with data types. For each real scenario, we instantiate from the predefined state machines the proper ones for each scenario. The first scenario shows the correct functioning of the architecture, and it is about how it reacts in case of a fire (for example) and the second is about security.

The First Scenario: Here, we will check a general case, where a fire is in a smart room, and we will look to the reaction of sensors, then we exploit collected information submitted to server and smartphones, and also retrieved data from database. The scenario is unfolded as follows.

1. We turn on all smart room device, and we make both the server, database, and actuator subscribe in the Broker to receive the acknowledgment messages from it. Then, we make the Smart-phone authenticate to the server in order to exchange the messages between them (client-server).
2. We increase a parameter that represents the degree of smoke, and when it is greater than or equal to a threshold already defined, the condition which identifies that there is a fire is verified, then the sensor goes to the transmission state after sending a message to Broker.

3. When the Broker receive the message, it sends values to database to store current changes, and it sends to the server if the last two machines are subscribe in Broker, else the transmission process will be stopped.
4. When the server receive the message, it discover its type (Broker message), it reacts with the new value and it delivers a signal command to the Broker. As the smart-phone authenticate to the server, the server can send an encrypted alert message to it.
5. Broker passes the command to the specified actuator according to the topic, and as a result, it will be in the action state.
6. At this point, we test the ability of the smart-phone to access and retrieve the stored values from the database, as also the user wants to see the history events recorded within a period of time. So the Smart-phones send an encrypted select command to the Server.
7. The server checks the command and the authentication of the Smart-phone, if they are true it receives the command and delivers it to the database in the form of SQL command, else it stops the transmission process.
8. When the database receives command, it detects its type (select command), then it sends the data to the server without changing the stored data.
9. The server receives the database request, then it sends to the smart-phone the encrypted request that allows the smart-phone application displays the message after decryption.
10. Then, the smart-phone wants to update the data in the database. To do that, the smart-phone sends the encrypted message to the server if it is authenticated.
11. The server receives the command and identifies its type of command. Then it decrypts the message and delivers the SQL command to the database.
12. The Database detects the update command and resends the data to the server with changing of data stored in the Database.
13. The server receives Database request, then it delivers to the Smart-phone his encrypted message to inform the user of access the operation, The Smart-phone decrypts and displays the message.

The Second Scenario: This part checks the exactitude two security concepts. We check the confidentiality of information published by the broker and the subscribe objects (server in this case).

1. Turn on all smart room devices, we make the smart-phone authenticates into the server, and for the subscription in the broker we only subscribe both the database and the actuator (without server).
2. We increase a smoke parameter to move the sensor detection state then publication state.
3. The broker sends the received information only to the database.
4. The server cannot receive the information from the broker, because it has not subscription in the broker.

From this experiment results, we observe that the states machine work very well, together and in communication. All scenarios progress without deadlock and with a correct behavior. We conclude that all state machines form a correct and complete system, they execute without errors, and easy to deploy.

5 Conclusion

The present contribution develops a smart living framework that ensures the well correctness of a web based solution controlling smart rooms and checks its accuracy in real time. The architecture is complete in terms of components and safe communication, as well it is easy to modify, to extend, and to deploy. The framework defines precisely with an adequate semantics the most possible nodes needed for a smart living environment that facilitate easily their deployment and implementation. Further, the framework proposes an automatic way to check the well correctness of the developed smart living environment which reduce any unexpected failure. The experiments by simulation shows a significant where all implemented scenarios are free from deadlocks in the normal scenarios and respect the needed requirements.

As future work, we intend to extend the current work in several directions. First, we will show the flexibility and the scalability of the proposed solution to cover smart homes and buildings. We will also target the optimization problem for different smart objects parameters, like cost, position devices, coverage, etc. Further, we look to propose a blockchain solution to secure the communication of the proposed architecture and apply it on real cases.

References

1. Korzun, D.G., Balandin, S.I., Gurtov, A.V.: Deployment of smart spaces in internet of things: overview of the design challenges. In: Balandin, S., Andreev, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART -2013. LNCS, vol. 8121, pp. 48–59. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40316-3_5
2. Sfikas, G., Akasiadis, C., Spyrou, E.: Creating a smart room using an IoT approach, May 2016
3. Janssens, A., De Paepe, M.: Effect of moisture inertia models on the predicted indoor humidity in a room. In: Proceedings of the 26th AIVC Conference, January 2005
4. Tejani, D., Al-Kuwari, A., Potdar, V.: Energy conservation in a smart home, May 2011
5. Tang, K., Wang, Y., Liu, H., Sheng, Y., Wang, X., Wei, Z.: Design and implementation of push notification system based on the MQTT protocol. In: 2013 International Conference on Information Science and Computer Applications (ISCA 2013). Atlantis Press, October 2013
6. Uppaal (2019). http://www.uppaal.org/
7. Ouchani, S.: Ensuring the functional correctness of IoT through formal modeling and verification. In: Abdelwahed, E.H., Bellatreche, L., Golfarelli, M., Méry, D., Ordonez, C. (eds.) MEDI 2018. LNCS, vol. 11163, pp. 401–417. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00856-7_27

8. Moreno-Salinas, D., Pascoal, A.M., Aranda, J.: Optimal sensor placement for multiple target positioning with range-only measurements in two-dimensional scenarios. Sensors **13**(8), 10674–10710 (2013)
9. Centenaro, M., Vangelista, L., Zanella, A., Zorzi, M.: Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. IEEE Wirel. Commun. **23**, 60–67 (2016)
10. Sanchez, L., et al.: Smartsantander: IoT experimentation over a smart city testbed. Comput. Netw. **61**, 217–238 (2014). Special issue on Future Internet Testbeds – Part I
11. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014)
12. Devasena, C.L.: IPv6 low power wireless personal area network (6LoWPAN) for networking internet of things (IoT) - analyzing its suitability for IoT. Indian J. Sci. Tech. **9**, 30 (2016)
13. Andreev, S., Galinina, O., Pyattaev, A., Gerasimenko, M., Tirronen, T., Torsner, J., Sachs, J., Dohler, M., Koucheryavy, Y.: Understanding the IoT connectivity landscape: a contemporary M2M radio technology roadmap. IEEE Commun. Mag. **53**(9), 32–40 (2015)
14. Fafoutis, X., Elsts, A., Vafeas, A., Oikonomou, G., Piechocki, R.: On predicting the battery lifetime of IoT devices: experiences from the sphere deployments. In: Proceedings of the 7th International Workshop on Real-World Embedded Wireless Systems and Networks, RealWSN 2018, pp. 7–12. ACM, New York, NY, USA (2018)
15. Babar, S., Mahalle, P., Stango, A., Prasad, N., Prasad, R.: Proposed security model and threat taxonomy for the internet of things (IoT). In: Meghanathan, N., Boumerdassi, S., Chaki, N., Nagamalai, D. (eds.) CNSA 2010. CCIS, vol. 89, pp. 420–429. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14478-3_42
16. Sehgal, A., Perelman, V., Kuryla, S., Schonwalder, J.: Management of resource constrained devices in the internet of things. IEEE Commun. Mag. **50**(12), 144–149 (2012)
17. Aazam, M., Huh, E.: Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp. 687–694, March 2015
18. Desai, P., Sheth, A., Anantharam, P.: Semantic gateway as a service architecture for IoT interoperability. In: 2015 IEEE International Conference on Mobile Services, pp. 313–319, June 2015
19. Kingatua, A.: IoT system tests: checking for failure
20. Chen, Y., Kunz, T.: Performance evaluation of IoT protocols under a constrained wireless access network. In: 2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT), pp. 1–7, April 2016
21. Xiao, G., Guo, J., Xu, L.D., Gong, Z.: User interoperability with heterogeneous iot devices through transformation. IEEE Trans. Industr. Inf. **10**(2), 1486–1496 (2014)
22. IoT sensors (2019). https://fiware-tutorials.readthedocs.io/en/latest/iot-sensors/
23. Zhu, Q., Wang, R., Chen, Q., Liu, Y., Qin, W.: IoT gateway: bridgingwireless sensor networks into internet of things. In: 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 347–352, December 2010

Database Theory and Rigorous Methods

Concurrent Computing with Shared Replicated Memory

Klaus-Dieter Schewe[1](✉), Andreas Prinz[2], and Egon Börger[3]

[1] UIUC Institute, Zhejiang University, Haining, China
kdschewe@acm.org

[2] Department of ICT, University of Agder, Agder, Norway
andreas.prinz@uia.no

[3] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
boerger@di.unipi.it

Abstract. Any concurrent system can be captured by a concurrent Abstract State Machine (cASM). This remains valid, if different agents can only interact via messages. It even permits a strict separation between memory managing agents and other agents that can only access the shared memory by sending query and update requests. This paper is dedicated to an investigation of replicated data that is maintained by a memory management subsystem, where the replication neither appears in the requests nor in the corresponding answers. We specify the behaviour of a concurrent system with such memory management using concurrent communicating ASMs (ccASMs), provide several refinements addressing different replication policies and internal messaging between data centres, and analyse their effects on the runs with respect to consistency. We show that on a concrete level even a weak form of consistency is only possible, if view serialisability can be guaranteed.

1 Introduction

Abstract State Machines (ASMs) have been used successfully to model sequential, parallel and concurrent systems [7]. They are mathematically grounded in behavioural theories of sequential algorithms [10], parallel algorithms [3,8] and concurrent algorithms [4], by means of which it is proven that they capture the corresponding, axiomatically defined class of algorithms.

In particular, the behavioural theory of asynchronous concurrent systems shows that every concurrent system can be step-by-step simulated by a concurrent Abstract State Machine (cASM) [4]. The theory provides an important breakthrough in the theory of concurrent and distributed system, as it avoids mimicking concurrent behaviour by means of sequentiality using techniques such as interleaving and choice by an unspecified super-agent (see the semantics defined in [12]). This modernised theory of concurrency is so far only realised in cASMs. The proof of the concurrent ASM thesis was first only conducted for families of sequential algorithms, but the generalisation to families of parallel algorithms does not cause serious difficulties as sketched in [16].

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 219–234, 2019.
https://doi.org/10.1007/978-3-030-32065-2_16

The theory can be applied to many different models of concurrent computation (see e.g. [1,2,9,13,18,19]). It also remains valid, if different agents can only interact via messages [5]. This includes the case of a strict separation between memory managing agents and other agents that can only access the shared memory by sending query and update requests. Naturally, the expectation of an agent a sending a request to a memory management agent b is that the returned result is the same as if the agent a executed its request directly on the shared locations. This is challenged, if the memory management subsystem replicates the data as e.g. in the noSQL database system Cassandra [15].

In this paper we investigate the case of replicated data maintained by a memory management subsystem, where the replication should not appear in the requests nor in the corresponding answers, which is a standard requirement in distributed databases with replication [14, Chap. 13]. Our objective is that the behaviour of the memory management subsystem must be understandable from the specification so that additional consistency assurance measures can be added if necessary.

For instance, consider four agents $a_1, \dots, a_4$ having the following respective programs (where; denotes sequential execution, and $Print(x)$ means to read the value x and to copy it to some output):

$$x := 1 \mid y := 1 \mid Print(x); Print(y) \mid Print(y); Print(x)$$

Then there is no concurrent run where (1) initially $x = y = 0$, (2) each agent makes once each possible move, (3) a_3 prints $x = 1, y = 0$, and (4) a_4 prints $x = 0, y = 1$. However, if x and y are replicated, say that there are always two copies, and an update by the programs a_1 or a_2 affects only a single copy, such an undesirable behaviour will indeed be enabled.

We assume a rather simple model, where shared data is logically organised in relations with primary keys, and data can only be accessed by means of the primary key values. We further assume relations to be horizontally fragmented according to values of a hash function on the primary key values, and these fragments are replicated. Replicas are assigned to different nodes, and several nodes together form a data centre, i.e. they are handled by one dedicated data management agent. In addition, values in replicas carry logical timestamps set by the data centres and stored in the records in the nodes of the memory management subsystem. This allows us to formulate and investigate policies that guarantee consistency.

For retrieval of a set of records a specified number of replicas has to be read, and for each record always the one with the latest timestamp will be returned. Depending on how many replicas are accessed the returned records may be (in the strict sense) outdated or not. Likewise, for the update of a set of records timestamps will be created, and a specified number of replicas of the records will be stored. Success of retrieval or update will be returned according to specified read- and write-policies.

In Sect. 2 we will first specify the behaviour of a concurrent system with shared data requiring that all agents interact with this subsystem for data retrieval and updates using appropriate SEND and RECEIVE actions. The memory management subsystem is specified by a separate collection of agents. In Sect. 3 we investigate a refinement concerning policies how many replicas are to be read or updated, respectively. We show that some combinations of replication policies enable *view compatibility*, which formalises the expectation above. In Sect. 4 we refine our specification taking the communication between data centres into account, and address the enforcement of the read and write policies. We obtain a complete, though not necessarily correct refinement, and as a consequence view compatibility cannot be guaranteed anymore. We even show that view compatibility implies view serialisability. Finally, we conclude with a brief summary and outlook.

This paper contains only a short version of our work on the subject, but a technical report with full details is available in [17]. In particular, proofs are only sketched here, but full-length proofs appear in this technical report.

2 Shared Memory Management with Replication

We assume some familiarity with Abstract State Machines (ASMs) (see [7, Sect. 2.2/4]). The *signature* Σ of an ASM is a finite set of function symbols f, each associated with an arity ar_f. A *state* S is a set of functions f_S of arity $n = ar_f$ over some fixed base set B, given by interpretations of the corresponding function symbol f. Each pair $(f, (v_1, \dots, v_n))$ comprising a function symbol and arguments $v_i \in B$ is called a *location*, and each pair (ℓ, v) of a location ℓ and a value $v \in B$ is called an *update*. A set of updates is called an *update set*. The evaluation of terms is defined as usual by $val_S(f(t_1, \dots, t_n)) = f_S(val_S(t_1), \dots, val_S(t_n))$. ASM *rules* r are composed using

assignments. $f(t_1, \dots, t_n) := t_0$ (with terms t_i built over Σ),
branching. IF φ THEN r_+ ELSE r_-,
parallel composition. FORALL x WITH $\varphi(x)$ $r(x)$,
bounded parallel composition. $r_1 \dots r_n$,
choice. CHOOSE x WITH $\varphi(x)$ IN $r(x)$, and
let. LET $x = t$ IN $r(x)$.

Each rule yields an update set $\Delta(S)$ in state S. If this update set is consistent, i.e. it does not contain two updates $(\ell, v), (\ell, v')$ with the same location ℓ and different values $v \neq v'$, then applying this update set defines a successor state $S + \Delta(S)$.

2.1 Concurrent Communicating Abstract State Machines

A *concurrent ASM* (cASM) $\mathcal{CM}$ is defined as a family $\{(a, asm_a)\}_{a \in \mathcal{A}}$ of pairs consisting of an agent a and an ASM asm_a. Let Σ_a denote the signature of the ASM asm_a. Taking the union $\Sigma = \bigcup_{a \in \mathcal{A}} \Sigma_a$ we distinguish between $\mathcal{CM}$-states built over Σ and local states for agent a built over Σ_a; the latter ones are simply projections of the former ones on the subsignature.

A *concurrent run* of a concurrent ASM $\mathcal{CM} = \{(a, asm_a)\}_{a \in \mathcal{A}}$ is a sequence $S_0, S_1, S_2, \dots$ of $\mathcal{CM}$-states, such that for each $n \geq 0$ there is a finite set $A_n \subseteq \mathcal{A}$ of agents such that S_{n+1} results from simultaneously applying update sets $\Delta_a(S_{j(a)})$ for all agents $a \in A_n$ yielded by asm_a in some preceding state $S_{j(a)}$ ($j(a) \leq n$ depending on a), i.e. $S_{n+1} = S_n + \bigcup_{a \in A_n} \Delta_a(S_{j(a)})$ and $a \notin \bigcup_{i=j(a)}^{n-1} A_i$.

In order to isolate agents responsible for a memory management subsystem we exploit *communicating concurrent ASMs* (ccASM) [5]. In a ccASM the only shared function symbols take the form of mailboxes. Sending of a message m from a to b means to update the out-mailbox of a by inserting m into it. This mailbox is a set-valued shared location with the restriction that only the sender can insert messages into it and only the environment, i.e. the message processing system, can read and delete them. The message processing system will move the message m to the in-mailbox of the receiver b. Receiving a message m by b means in particular that b removes m from its in-mailbox and performs some local operation on m. Therefore, in ccASMs the language of ASM rules above is enriched by the following constructs (see [5] for further details):

Send. `SEND`(⟨message⟩, from:⟨sender⟩, to:⟨receiver⟩),
Receive. `RECEIVE`(⟨message⟩, from:⟨sender⟩, to:⟨receiver⟩),
Received. `RECEIVED`(⟨message⟩, from:⟨sender⟩, to:⟨receiver⟩), and
Consume. `CONSUME`(⟨message⟩, from:⟨sender⟩, to:⟨receiver⟩).

If all shared data is organised in relations with a unique primary key, this can be modelled by a set of function symbols $\Sigma_{\text{mem}} = \{p_1, \dots, p_k\}$, where each p_i has a fixed arity a_i, and a fixed co-arity c_i, such that in each state S we obtain partial functions[1] $p_i^S : B^{a_i} \rightarrow B^{c_i}$.

A read access by an agent $a \in \mathcal{A}$ aims at receiving a subset of relation p_i containing those records with key values satisfying a condition φ, i.e. evaluate a term $p_i[\varphi] = \{(\boldsymbol{k}, \boldsymbol{v}) \mid \varphi(\boldsymbol{k}) \wedge \boldsymbol{v} \neq \mathit{undef} \wedge p_i(\boldsymbol{k}) = \boldsymbol{v}\}$. As p_i is not in the signature Σ_a, the agent a must send a read-request and wait for a response, i.e. it executes `SEND`(read(p_i, φ),from:a,to:$home(a)$) and waits until `RECEIVED`(answer(ans, p_i, φ),from:$home(a)$,to:a) becomes true. Then it can execute `RECEIVE`(answer(ans, p_i, φ), from:$home(a)$,to:a) to obtain the requested value.

We abstract from the details of the communication but assume the communication to be reliable. If there is no confusion, we omit the sender and receiver parameters in `SEND` and `RECEIVE`. The ans in the message must be a relation of arity $a_i + c_i$ satisfying the key property above. The agent can store such an answer using a non-shared function p_i^a or process it in any other way, e.g. aggregate the received values. This is part of the ASM rule in asm_a, which we do not consider any further.

In the `SEND` and `RECEIVE` rules we use a fixed agent $home(a)$ with which the agent a communicates. It will be unknown to the agent a, whether this

[1] In ASMs partial functions are captured by total functions using a dedicated value *undef*.

agent $home(a)$ processes the read-request or whether it communicates with other agents to produce the answer.

Analogously, for bulk write access an agent a may want to execute the operation p_i :& p to update all records with a key defined in p to the new values given by p. As this corresponds to an ASM rule FORALL $(\boldsymbol{k}, \boldsymbol{v}) \in p \;\; p_i(\boldsymbol{k}) := \boldsymbol{v}$, the agent a must send a write-request and wait for a response, i.e. it executes SEND(write(p_i, p),to:$home(a)$) and waits to receive an acknowledgement, i.e. to RECEIVE(acknowledge(p_i, p), from:$home(a)$).

We use the notation $\mathcal{CM}_0 = \{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(db, asm_{db})\}$ for the ccASM with a single memory agent db and $home(a) = db$ for all $a \in \mathcal{A}$. Thus, the rule of asm_{db} looks as follows:

```
IF RECEIVED(read($p_i, \varphi$),from:$a$) THEN
   CONSUME(read($p_i, \varphi$),from:$a$)
   LET $ans = \{(\boldsymbol{k}, \boldsymbol{v}) \mid \varphi(\boldsymbol{k}) \wedge p_i(\boldsymbol{k}) = \boldsymbol{v} \wedge \boldsymbol{v} \neq undef\}$ IN
      SEND(answer($ans, p_i, \varphi$),to:$a$)
IF RECEIVED(write($p_i, p$),from:$a$) THEN
   CONSUME(write($p_i, p$),from:$a$)
   FORALL $(\boldsymbol{k}, \boldsymbol{v}) \in p \;\; p_i(\boldsymbol{k}) := \boldsymbol{v}$
   SEND(acknowledge($p_i, p$),to:$a$)
```

2.2 Memory Organisation with Replication

For replication we use several *data centres*, and each data centre comprises several *nodes*. The nodes are used for data storage, and data centres correspond to physical machines maintaining several such storage locations. Let $\mathcal{D}$ denote the set of data centres. Then instead of a location $(p_i, \boldsymbol{k})$ there will always be several replicas, and at each replica we may have a different value.

Let us assume that each relation p_i is fragmented according to the values of a hash-key. That is, for each $i = 1, \dots, k$ we can assume a static hash-function $h_i : B^{a_i} \to [m, M] \subseteq \mathbb{Z}$ assigning a hash-key to each key value. We further assume a partition $[m, M] = \bigcup_{j=1}^{q_i} range_j$ such that $range_{j_1} < range_{j_2}$ holds for all $j_1 < j_2$, so each range will again be an interval. These range intervals are used for the horizontal fragmentation into q_i fragments of the to-be-represented function p_i: $Frag_{j,i} = \{\boldsymbol{k} \in B^{a_i} \mid h_i(\boldsymbol{k}) \in range_j\}$.

All these fragments will be replicated and their elements associated with a value (where defined by the memory management system), using a fixed *replication factor* r_i. That is, each fragment $Frag_{j,i}$ will be replicated r_i-times for each data centre. A set of all pairs $(\boldsymbol{k}, \boldsymbol{v})$ with key $\boldsymbol{k} \in Frag_{j,i}$ and an associated value $\boldsymbol{v}$ in the memory management system is called a replica of $Frag_{j,i}$.

Assume that each data centre d consists of n_i nodes, identified by d and a number $j' \in \{1, \dots, n_i\}$. Then we use a predicate $copy(i, j, d, j')$ to denote that the node with number j' in the data centre $d \in \mathcal{D}$ contains a replica of $Frag_{j,i}$. We also use $\mathcal{D}_i = \{d \in \mathcal{D} | \exists j, j'.copy(i, j, d, j')\}$. To denote the values in replicas we use dynamic functions $p_{i,j,d,j'}$ of arity a_i and co-arity $c_i + 1$ (functions we call again replicas). So we use function symbols $p_{i,j,d,j'}$ with $j \in \{1, \dots, q_i\}$, $d \in \mathcal{D}$

and $j' \in \{1, \dots, n_i\}$, and we request $h_i(\boldsymbol{k}) \in \mathit{range}_j$ for all $\boldsymbol{k} \in B^{a_i}$, whenever $\mathit{copy}(i, j, d, j')$ holds and $p_{i,j,d,j'}(\boldsymbol{k})$ is defined. For the associated values we have $p_{i,j,d,j'}(\boldsymbol{k}) = (\boldsymbol{v}, t)$, where t is an added timestamp value, and values $\boldsymbol{v}$ may differ from replica to replica.

Each data centre d maintains a logical clock clock_d that is assumed to advance (without this being further specified), and clock_d evaluates to the current time at data centre d. Timestamps must be totally ordered, differ if set by different data centres, and respect the inherent order of message passing, i.e. when data with a timestamp t is created at data centre d and sent to data centre d', then at the time the message is received the clock at d' must show a time larger than t. This condition can be enforced by adjusting clocks according to Lamport's algorithm [11]. For this let us define adjust_clock$(d, t) = \mathit{clock}_d := t'$, where t' is the smallest possible timestamp at data centre d with $t \leq t'$.

2.3 Internal Request Handling for Replicated Memory

When dealing with replication the request messages sent by agents a remain the same, but the internal request handling by the memory management subsystem changes. This will define a refined ccASM $\mathcal{CM}_1 = \{(a, \mathit{asm}_a^c)\}_{a \in \mathcal{A}} \cup \{(d, \mathit{asm}_d)\}_{d \in \mathcal{D}}$. We will use the notions of *complete* and *correct* refinement as defined in [7, pp. 111ff.].

Let M and M^* be cASMs (or ccASMs). We fix a correspondence relation $\sim$ between some states of M and some states of M^*. Then M^* is called a *correct refinement* of M iff for each run $S_0^*, S_1^*, \dots$ of M^* there is a run $S_0, S_1, \dots$ of M together with sequences $0 = i_0 < i_1 < \dots$ and $0 = j_0 < j_1 < \dots$ such that $S_{i_k} \sim S_{j_k}^*$ holds for all k, and if both runs are finite with final states S_f^* and S_f, respectively, then there exists an index ℓ with $S_{i_\ell} = S_f$ and $S_{i_\ell}^* = S_f^*$. We call M^* a *complete refinement* of M iff M is a correct refinement of M^*.

Consider a read request read(p_i, φ) received from agent a by data centre d. As data is horizontally fragmented, we need to evaluate several requests read$(p_{i,j}, \varphi)$ concerning keys $\boldsymbol{k}$ with $h_i(\boldsymbol{k}) \in \mathit{range}_j$, one request for each fragment index j, and then build the union so that $p_i[\varphi] = \bigcup_{j=1}^{q_i} p_{i,j}[\varphi]$. In order to evaluate $p_{i,j}[\varphi]$ several replicas of $\mathit{Frag}_{j,i}$ will have to be accessed. Here we will leave out any details on how these replicas will be selected and accessed, but the selection of replicas must comply with a *read-policy* that is left abstract for the moment.

When reading actual data, i.e. evaluating $p_{i,j,d,j'}(\boldsymbol{k})$ for selected key values $\boldsymbol{k}$, we obtain different time-stamped values $(\boldsymbol{v}, t)$, out of which a value $\boldsymbol{v}$ with the latest timestamp is selected and sent to a as the up-to-date value of $p_i(\boldsymbol{k})$. The requirement that timestamps set by different data centres differ implies that for given $\boldsymbol{k}$ the value $\boldsymbol{v}$ with the latest timestamp is unique. All records obtained this way will be returned as the result of the read request to the issuing agent a. Thus, we obtain the following ASM rule `AnswerReadReq`:

```
AnswerReadReq =
IF RECEIVED(read($p_i, \varphi$), from:$a$) THEN
    CONSUME(read($p_i, \varphi$), from:$a$)
    FORALL $j \in \{1, \dots, q_i\}$ CHOOSE $G_{i,j}$ WITH $\mathit{complies}(G_{i,j}, \text{read-policy})$
       $\wedge\, G_{i,j} \subseteq \{(d', j') \mid \mathit{copy}(i, j, d', j') \wedge d' \in \mathcal{D}_i \wedge 1 \le j' \le n_i\}$
    LET $t_{\max}(\boldsymbol{k}) = \max\{t \mid \exists \boldsymbol{v}', \bar{d}, \bar{j}.(\bar{d}, \bar{j}) \in G_{i,j} \wedge p_{i,j,\bar{d},\bar{j}}(\boldsymbol{k}) = (\boldsymbol{v}', t)\}$ IN
    LET $\mathit{ans}_{i,j} = \{(\boldsymbol{k}, \boldsymbol{v}) \mid \varphi(\boldsymbol{k}) \wedge h_i(\boldsymbol{k}) \in \mathit{range}_j \wedge \boldsymbol{v} \neq \mathit{undef} \wedge$
              $\exists d', j'.((d', j') \in G_{i,j} \wedge p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}, t_{\max}(\boldsymbol{k})))\}$ IN
    LET $\mathit{ans} = \bigcup_{j=1}^{q_i} \mathit{ans}_{i,j}$ IN SEND(answer($\mathit{ans}, p_i, \varphi$), to:$a$)
```

Note that the unique value $\boldsymbol{v}$ with $p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}, t_{\max}(\boldsymbol{k}))$ may be *undef* and that the returned ans may be the empty set.

For a write request write(p_i, p) sent by agent a to data centre d we proceed analogously. In all replicas of $\mathit{Frag}_{j,i}$ selected by a *write-policy* the records with a key value in p will be updated to the new value provided by p—this may be *undef* to capture deletion—and a timestamp given by the current time clock_d. However, the update will not be executed, if the timestamp of the existing record is already newer. In addition, clocks that "are too late" will be adjusted, i.e. if the new timestamp received from the managing data centre d is larger than the timestamp at data centre d', the clock at d' is set to the received timestamp. Thus, we obtain the following ASM rule PerformWriteReq to-be-executed by any data centre d upon receipt of an update request from an agent a:

```
PerformWriteReq =
IF RECEIVED(write($p_i, p$), from:$a$) THEN
   CONSUME(write($p_i, p$), from:$a$)
   FORALL $j \in \{1, \dots, q_i\}$ CHOOSE $G_{i,j}$ WITH $\mathit{complies}(G_{i,j}, \text{write-policy})$
       $\wedge\, G_{i,j} \subseteq \{(d', j') \mid \mathit{copy}(i, j, d', j') \wedge d' \in \mathcal{D}_i \wedge 1 \le j' \le n_i\}$
   LET $t_{current} = \mathit{clock}_{\texttt{self}}$ IN
       FORALL $(d', j') \in G_{i,j}$
         FORALL $(\boldsymbol{k}, \boldsymbol{v}) \in p$ WITH $h_i(\boldsymbol{k}) \in \mathit{range}_j$
               IF $\exists \boldsymbol{v}', t.p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}', t) \wedge t < t_{current}$ THEN
                  $p_{i,j,d',j'}(\boldsymbol{k}) := (\boldsymbol{v}, t_{current})$
         IF $\mathit{clock}_{d'} < t_{current}$ THEN adjust_clock($d', t_{current}$)
   SEND(acknowledge($p_i, p$), to:$a$)
```

Proposition 1. *The ccASM* $\mathcal{CM}_1 = \{(a, \mathit{asm}_a^c)\}_{a \in \mathcal{A}} \cup \{(d, \mathit{asm}_d)\}_{d \in \mathcal{D}}$ *is a complete refinement of* $\mathcal{CM}_0 = \{(a, \mathit{asm}_a^c)\}_{a \in \mathcal{A}} \cup \{(db, \mathit{asm}_{db})\}$.

Proof (sketch, see also [17]*).* The only differences between the ccASMs $\mathcal{CM}_0$ and $\mathcal{CM}_1$ are that $\mathit{home}(a) \in \mathcal{D}$ differs in the refinement, but in both cases the handling of a request is done in a single step. For both cases the determination of the answer requires a more sophisticated rule in the refinement. □

3 Refinement Using Replication Policies

We define *view compatibility* formalising the intuitive expectation of the agents that answers to sent requests remain the same in case of replication as without, because replication is completely transparent. We show that for particular combinations of concrete read- and write-policies $\mathcal{CM}_1$ guarantees view compatibility, which further implies that the refinement of $\mathcal{CM}_0$ by $\mathcal{CM}_1$ is correct.

3.1 View Compatibility

Informally, view compatibility is to ensure that the system behaves in a way that whenever an agent sends a read- or write-request the result is the same as if the read or write had been executed in a state without replication. For a formal definition we need the notion of an *agent view* of a concurrent run $S_0, S_1, \dots$ of the cASM $\{(a, asm_a^c)\}_{a \in \mathcal{A}}$ for an arbitrary agent $a \in \mathcal{A}$. Its view of the run is the subsequence of states $S_{a,0}, S_{a,1}, \dots$ in which a makes a move (restricted to the signature of a). For any state $S_k = S_{a,n}$ its successor state in the a-view sequence depends on the move a performs in $S_{a,n}$.

If a in $S_{a,n}$ performs a send step, it contributes to the next state S_{k+1} by an update set which includes an update of its out-mailbox, which in turn yields an update of the mailbox of *home*(a). But S_{k+1} is not yet the next a-view state, in which a will perform its next move. This move is determined by the *atomic request/reply assumption* for agent/db runs: If in a run an agent performs a send step, then its next step in the run is the corresponding receive step, which can be performed once the answer to the sent request has been received. By this assumption the next a-view state $S_{a,n+1} = S_l$ is determined by (what appears to a as) an environment action enabling the receive step by inserting the reply message into a's mailbox.

If a in $S_{a,n} = S_k$ performs a receive or an internal step, then besides the mailbox update to consume the received message it yields only updates to non-shared locations so that its next a-view state is the result of applying these updates together with updates of other agents to form $S_{k+1} = S_{a,n+1}$.

We further need the notion of a *flattening* which reduces the multiple values associated with replicas of a location ℓ to a single value: If $S_0, S_1, S_2, \dots$ is a run of $\mathcal{CM}_1 = \{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(d, asm_d)\}_{d \in \mathcal{D}}$, then we obtain a *flattening* $S'_0, S'_1, S'_2, \dots$ by replacing in each state all locations $(p_{i,j,d',j'}, \boldsymbol{k})$ by a single location $(p_i, \boldsymbol{k})$ and letting the value associated with $(p_i, \boldsymbol{k})$ be one of the values in $\{\boldsymbol{v} \mid \exists j, d', j'. \exists t. p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}, t)\}$.

Obviously, a flattening is a sequence of states of the concurrent ASM $\{(a, asm_a^c)\}_{a \in \mathcal{A}}$, but in most cases it will not be a run. Therefore, take an arbitrary subsequence $S'_{j_0}, S'_{j_1}, \dots$ of an arbitrary flattening $S'_0, S'_1, \dots$ (restricted to the signature of the agent a) of $S_0, S_1, \dots$. Then $S'_{j_0}, S'_{j_1}, \dots$ is called a *flat view* of agent a of the run $S_0, S_1, \dots$ if the following conditions hold: Whenever a performs a request in state S_k there is some S'_{j_i} such that $k = j_i$. If the corresponding reply is received in state S_m for some $m > k$, then $S'_{j_{i+1}} = S_m$. Furthermore, there exists some n with $k < n \leq m$ such that if the request is a write-request, then for each location ℓ with value v in this request $val_{S'_n}(\ell) = v$ holds, provided there exists an agent reading the value v, and if the request is a read-request, then for each location ℓ with value v in the answer $val_{S'_n}(\ell) = v$ holds. Whenever a performs a RECEIVE or an internal move in state S_k there is some j_i such that $S_k = S'_{j_i}$ and $S_{k+1} = S'_{j_{i+1}}$.

We say that $\{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(d, asm_d)\}_{d \in \mathcal{D}}$ is *view compatible* with the concurrent ASM $\{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(db, asm_{db})\}$ iff for each run $\mathcal{R} = S_0, S_1, S_2, \dots$ of $\{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(d, asm_d)\}_{d \in \mathcal{D}}$ there exists a subsequence of a flattening

$\mathcal{R}' = S'_0, S'_1, S'_2, \dots$ that is a run of $\{(a, asm^c_a)\}_{a\in\mathcal{A}} \cup \{(db, asm_{db})\}$ such that for each agent $a \in \mathcal{A}$ the agent a-view of $\mathcal{R}'$ coincides with a flat view of $\mathcal{R}$ by a.

3.2 Specification of Replication Policies

In the specification of ASM rules handling read and write requests by a fixed data centre d we used sets $G_{i,j} \subseteq C_{i,j}$ with $C_{i,j} = \{(d', j') \mid copy(i, j, d', j')\}$ as well as an abstract predicate $complies(G_{i,j}, \text{policy})$. Let us now define the most important policies All, One, and Quorum—more policies are handled in [17]. These policies differ in the number of replicas that are to be accessed. As the name indicates, the predicate $complies(G_{i,j}, \text{All})$ can be defined by $G_{i,j} = C_{i,j}$, i.e. all replicas are to be accessed. For One at least one replica is to be accessed, which defines $complies(G_{i,j}, \text{One})$ by $|G_{i,j}| \geq 1$. For $\text{Quorum}(q)$ we use a value q with $0 < q < 1$, and $complies(G_{i,j}, \text{Quorum}(q))$ is defined by $q \cdot |C_{i,j}| < |G_{i,j}|$.

For consistency analysis we need *appropriate combinations* of read- and write-policies. If the write-policy is $\text{Quorum}(q)$ and the read-policy is $\text{Quorum}(q')$ with $q+q' \geq 1$, then the combination is appropriate. Furthermore, a combination of the write policy (or read-policy) ALL with any other policy is also appropriate.

Proposition 2. *If the combination of the read and write policies is appropriate, then $\mathcal{CM}_1$ is view compatible with the cASM $\{(a, asm^c_a)\}_{a\in\mathcal{A}}$ and a correct refinement of $\mathcal{CM}_0$.*

Proof (sketch, a full proof is available in [17]*).* If the write-policy is $\text{Quorum}(q)$, then for each location ℓ the multiplicity of replicas considered to determine the value with the largest timestamp is at least $\lceil \frac{m+1}{2} \rceil$ with m being the total number of replicas. Consequently, each read access with a policy $\text{Quorum}(q')$ (with $q + q' \geq 1$) reads at least once this value and returns it. That is, in every state only the value with the largest timestamp for each location uniquely determines the run, which defines the equivalent concurrent run. □

4 Refinement with Internal Communication

We will now address a refinement of the memory management subsystem taking into account that data centres refer to different physical machines, whereas in $\mathcal{CM}_1$ we abstracted from any internal communication. The gist of the refinement is therefore to treat the handling of a request as a combination of direct access to local nodes, remote access via messages to the other relevant data centres, and collecting and processing return messages until the requirements for the read- or write-policies are fulfilled. That is, the validation of the policy accompanies the preparation of a response message and is no longer under control of the *home* agent.

4.1 Request Handling with Communicating Data Centres

In Sect. 2 we specified how a data centre agent d handles a request received from an agent a. Now, we first specify an abstract rule which manages external requests, i.e. coming from an agent a and received by a data centre d, where *request* is one of these read or write requests. An external request is forwarded as internal request to all other data centres $d' \in \mathcal{D}_i$, where it is handled and answered locally (see the definition of `HandleLocally` below), whereas collecting (in $answer_{a'}$) and sending the overall answer to the external agent a is delegated to a new agent a'. `SELF` denotes the data centre agent d which executes the rule.

To `Initialize` a delegate a' it is equipped with a set $answer_{a'}$, where to collect the values arriving from the asked data centres and with counters $count_{a'}(j)$ (for the number of inspected replicas of the j-th fragment). The counters are used for checking compliance with the policies. The *mediator* and *requestor* information serves to retrieve sender and receiver once the delegate completes the answer to the request.

```
DelegateExternalReq =
 IF RECEIVED($request$, from:$a$) THEN
  CONSUME($request$, from:$a$)
  LET $t_{\text{current}} = clock_{\texttt{SELF}}$, $a' = new(\text{Agent})$ IN
      Initialize($a'$)
      HandleLocally($request, a', t_{\text{current}}$)
      ForwardToOthers($request, a', t_{\text{current}}$)
WHERE
ForwardToOthers($request, a', t_{\text{current}}$)=
      FORALL $d' \in \mathcal{D}_i$ WITH $d' \neq$ SELF SEND(($request, a', t_{\text{current}}$), to:$d'$)
Initialize($a'$) =
      $answer_{a'} := \emptyset$
      FORALL $1 \leq j \leq q_i$ $count_{a'}(j) := 0$
      IF $request = \text{read}(p_i, \varphi)$
      THEN $asm_{a'} :=$ CollectRespondToRead
      ELSE $asm_{a'} :=$ CollectRespondToWrite
      $requestor_{a'} := a$
      $mediator_{a'} :=$ SELF
      $requestType_{a'} := request$
```

In this way the request handling agent d simply forwards the request to all other data centre agents and in parallel handles the request locally for all nodes associated with d. The newly created agent (a ‘delegate’) will take care of collecting all response messages and preparing the response to the issuing agent a. Request handling by any other data centre d' is simply done locally using the following rule:

```
ManageInternalReq =
  IF RECEIVED(($request, a', t$), from:$d$) THEN
      HandleLocally($request, a', t$)
      CONSUME(($request, a', t$), from:$d$)
```

Each data centre agent d is equipped with the following ASM rule, where the components `HandleLocally` and the two versions of `CollectRespond` are defined below.

$$asm_d = \texttt{DelegateExternalReq}\ \texttt{ManageInternalReq}$$

For local request handling policy checking is not performed by the data centre agent but by the delegate of the request; check the predicates *all_messages_received* and *sufficient*(policy) below. We use a predicate *alive* to check, whether a node is accessible or not. For a read request we specify `HandleLocally`(read(p_i, φ), a', t_{current}) as follows:

```
HandleLocally$(\text{read}(p_i, \varphi), a', t)$ =
    LET $d'$ = SELF IN
    LET $G_{i,j,d'} = \{j' \mid copy(i,j,d',j') \wedge alive(d',j')\}$ IN
    LET $t_{\max}(\boldsymbol{k}) = max(\{t \mid \exists \boldsymbol{v}', \bar{j}.\bar{j} \in G_{i,j,d'} \wedge p_{i,j,d',\bar{j}}(\boldsymbol{k}) = (\boldsymbol{v}', t)\})$ IN
    LET $ans_{i,j,d'} = \{(\boldsymbol{k}, \boldsymbol{v}, t_{\max}(\boldsymbol{k})) \mid \varphi(\boldsymbol{k}) \wedge h_i(\boldsymbol{k}) \in range_j \wedge$
                        $\exists j' \in G_{i,j,d'}.p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}, t_{\max}(\boldsymbol{k}))\}$ IN
    LET $ans = \bigcup_{j=1}^{q_i} ans_{i,j,d'}$, $\boldsymbol{x} = (|G_{i,1,d'}|, \dots, |G_{i,q_i,d'}|)$ IN
        SEND$(\text{answer}(ans, \boldsymbol{x}), \text{to:}a')$
```

Here we evaluate the request locally, but as the determined maximal timestamp may not be globally maximal, it is part of the returned relation. Also the number of replicas that contributed to the local result is returned, such that the delegate a' responsible for collection and final evaluation of the request can check the satisfaction of the read-policy. Therefore, the created partial result is not returned to the agent d that issued this local request, but instead to the delegate.

Rule `HandleLocally`(write(p_i, p), a', t_{current}) handles write requests:

```
HandleLocally$(\text{write}(p_i, p), a', t')$ =
    LET $d'$ = SELF IN
    IF $clock_{d'} < t'$ THEN adjust_clock$(d', t')$
    LET $G_{i,d'}(j) = \{j' \mid copy(i,j,d',j') \wedge alive(d',j')\}$ IN
    FORALL $j \in \{1, \dots, q_i\}$ FORALL $j' \in G_{i,d'}(j)$
        FORALL $(\boldsymbol{k}, \boldsymbol{v}) \in p$ WITH $h_i(\boldsymbol{k}) \in range_j$
            IF $\exists \boldsymbol{v}', t.p_{i,j,d',j'}(\boldsymbol{k}) = (\boldsymbol{v}', t) \wedge t < t'$ THEN $p_{i,j,d',j'}(\boldsymbol{k}) := (\boldsymbol{v}, t')$
    LET $\boldsymbol{x} = (|G_{i,1,d'}|, \dots, |G_{i,q_i,d'}|)$ IN SEND$(\text{ack\_write}(p_i, p, \boldsymbol{x}), \text{to:}a')$
```

Again, the partial results acknowledging the updates at the nodes associated with data centre d' are sent to the collecting agent a' to verify the compliance with the write-policy. For the delegate a' that has been created by d to collect partial responses and to create the final response to the agent a issuing the request we need predicates *sufficient*(policy) for policy checking, in which case the response to a is prepared and sent:

$$\begin{aligned} sufficient(\text{All}) &\equiv \forall j.(1 \le j \le q_i \Rightarrow count(j) = \gamma_{i,j}) \\ &\qquad\qquad \text{with } \gamma_{i,j} = |\{(d', j') \mid copy(i,j,d',j')\}| \\ sufficient(\text{One}) &\equiv \forall j.(1 \le j \le q_i \Rightarrow count(j) \ge 1) \end{aligned}$$

$$\mathit{sufficient}(\textsc{Quorum}(q)) \equiv \forall j.(1 \le j \le q_i \Rightarrow \gamma_{i,j} \cdot q < \mathit{count}(j))$$

It remains to specify the delegate rules `CollectRespondToRead` and `CollectRespondToWrite`, i.e. the programs associated with the agent a' created upon receiving a *request* from an agent a. The `CollectRespond` action is performed until all required messages have been received and splits into two rules for read and write requests, respectively.

The delegate a' collects the messages it receives from the data centres d' to which the original *request* had been forwarded to let them `HandleLocally`$(\mathit{request}, a')$. If the set of collected answers suffices to respond, the delegate sends an answer to the original requester and kills itself.

Thus each of the rules `CollectRespondToRead` and `CollectRespondToWrite` has a subrule `Collect` and a subrule `Respond` with corresponding parameter for the type of expected messages.

```
CollectRespondToRead =
  IF RECEIVED(answer($ans, \boldsymbol{x}$), from:$d'$) THEN
      CONSUME(answer($ans, \boldsymbol{x}$), from:$d'$)
      Collect(($ans, \boldsymbol{x}$), from:$d'$)
  IF $\mathit{sufficient}$(read-policy) THEN
      Respond($requestType_{\mathtt{SELF}}$)
WHERE
Respond(read($p_i, \varphi$)) =
  LET $d = mediator$(SELF),  $a = requestor$(SELF) IN
  LET $ans = \{(\boldsymbol{k}, \boldsymbol{v}) \mid \exists t.\ (\boldsymbol{k}, \boldsymbol{v}, t) \in answer_{\mathtt{SELF}}\}$ IN
      SEND(answer($ans, p_i, \varphi$), from:$d$, to:$a$)
  DELETE(SELF,Agent)
Collect(($ans, \boldsymbol{x}$), from : $d'$) =
  FORALL $\boldsymbol{k}$ WITH $\exists \boldsymbol{v}, t.\ (\boldsymbol{k}, \boldsymbol{v}, t) \in ans$
      LET $(\boldsymbol{k}, \boldsymbol{v}, t) \in ans$ IN
            IF $\exists \boldsymbol{v}', t'.\ (\boldsymbol{k}, \boldsymbol{v}', t') \in answer_{\mathtt{SELF}}$ THEN
            LET $(\boldsymbol{k}, \boldsymbol{v}', t') \in answer_{\mathtt{SELF}}$ IN
                  IF $t' < t$ THEN
                        DELETE(($\boldsymbol{k}, \boldsymbol{v}', t'$), $answer_{\mathtt{SELF}}$
                        INSERT(($\boldsymbol{k}, \boldsymbol{v}, t$), $answer_{\mathtt{SELF}}$
            ELSE  INSERT(($\boldsymbol{k}, \boldsymbol{v}, t$), $answer_{\mathtt{SELF}}$)
  LET $(x_1, \dots, x_{q_i}) = \boldsymbol{x}$ IN
      FORALL $j \in \{1, \dots, q_i\}$
            $count(j) := count(j) + x_j$
```

The analogous collection of messages for write requests is simpler, as the final response is only an acknowledgement.

```
CollectRespondToWrite=
  IF RECEIVED(ack_write($p_i, p, \boldsymbol{x}$), from:$d'$) THEN
      Collect(ack_write($p_i, \boldsymbol{x}$), from:$d'$)
      CONSUME(ack_write($p_i, p, \boldsymbol{x}$), from:$d'$)
```

```
    IF  $\mathit{sufficient}$(write-policy) THEN
        SEND(acknowledge($p_i, p$), from:$\mathit{mediator}$(SELF), to:$\mathit{requestor}$(SELF))
        DELETE(SELF,Agent)
WHERE
Collect(ack_write($p_i, \boldsymbol{x}$), from:$d'$) =
        LET $(x_1, \dots, x_{q_i}) = \boldsymbol{x}$ IN
                FORALL $j \in \{1, \dots, q_i\}$
                        $\mathit{count}(j) := \mathit{count}(j) + x_j$
```

Note that our specification does not yet deal with exception handling. We may tacitly assume that eventually all requested answers will be received by the collecting agent.

4.2 Analysis of the Refinement

Let $\mathcal{CM}_2$ denote the refined ccASM $\{(a, asm_a^c)\}_{a \in \mathcal{A}} \cup \{(d, asm'_d)\}_{d \in \mathcal{D} \cup \mathcal{E}xt}$ together with the dynamic set $\mathcal{E}xt$ of delegates. Note that the delegates are created on-the-fly by the agents $d \in \mathcal{D}$, needed for collecting partial responses for each request and preparing the final responses (a full proof is given in [17]).

Proposition 3. *$\mathcal{CM}_2$ is a complete refinement of $\mathcal{CM}_1$.*

Proof (sketch). Take a concurrent run of $\mathcal{CM}_1$ and first look at it from the perspective of a single agent $a \in \mathcal{A}$. Updates brought into the state S by a are read and write requests. If a continues in some state $S' = S_{i+x}$, then the transition from S to S' is achieved by means of a step of asm_d for $d = \mathit{home}(a)$. Therefore, there exist states $\bar{S}$, $\bar{S}'$ for $\mathcal{CM}_2$, in which the asm_a^c brings in the same read and write requests and receives the last response, respectively. In a concurrent run for $\mathcal{CM}_2$ the transition from $\bar{S}$ to $\bar{S}'$ results from several steps by the subsystem $\{(d, asm'_d)\}_{d \in \mathcal{D} \cup \mathcal{E}xt}$.

With respect to each of the requests received from a the agent $d = \mathit{home}(a)$ contributes to a state $\bar{S}_1$ with requests for each agent $d' \in \mathcal{D}$, $d' \neq d$, the creation and initialisation of a response collecting agent a', and the local handling of the request at nodes associated with data centre d. Then each agent $d' \in D$ contributes to some state $\bar{S}_k$ $(k > 1)$, in which the partial response to the request sent to d' is produced. Concurrently the collection agent a' on receipt of a partial response updates its own locations, and thus contributes to some state $\bar{S}_j$ $(j > 1)$. Finally, a' will also produce and send the response to a. This response will be the same as the one in state S', if the refined run from $\bar{S}$ to $\bar{S}'$ uses the same selection of copies for each request referring to p_i and each fragment $\mathit{Frag}_{j,i}$.

This implies that $\mathcal{CM}_{2,a} = \{(a, asm_a^c)\} \cup \{(d, asm'_d)\}_{d \in \mathcal{D} \cup \mathcal{E}xt}$ is a complete refinement of $\mathcal{CM}_{1,a} = \{(a, asm_a^c)\} \cup \{(d, asm_d)\}_{d \in \mathcal{D}}$, from which the proposition follows immediately. □

Unfortunately, view compatibility cannot be preserved, unless additional conditions are enforced in the specification. Any such condition already implies serialisability (a full proof is given in [17]).

Proposition 4. *If $\mathcal{CM}_2$ is view compatible with the cASM $\mathcal{CM}_0$, then every run $\mathcal{R}$ of $\mathcal{CM}_2$ is view serialisable. If all runs of $\mathcal{CM}_2$ are view serialisable and an appropriate combination of a read and a write policy is used, then $\mathcal{CM}_2$ is also view compatible with the concurrent ASM $\mathcal{CM}_0$.*

Proof (sketch). For a run $\mathcal{R} = S_0, S_1, \dots$ of $\mathcal{CM}_2$ view compatibility implies that there exists a subsequence of a flattening $\mathcal{R}' = S'_0, S'_1, \dots$ that is a run of $\{(a, asm^c_a)\}_{a \in \mathcal{A}} \cup \{(db, asm_{db})\}$ such that for each agent $a \in \mathcal{A}$ the agent a-view of $\mathcal{R}'$ coincides with a flat view of $\mathcal{R}$ by a. Let Δ'_ℓ be the update set defined by $S'_\ell + \Delta'_\ell = S'_{\ell+1}$, and define $\Delta_\ell =$

$$\{((p_{i,j,d,j'}, \boldsymbol{k}), (\boldsymbol{v}, t_\ell)) \mid ((p_i, \boldsymbol{k}), \boldsymbol{v}) \in \Delta'_\ell \wedge copy(i,j,d,j') \wedge h_i(\boldsymbol{k}) \in range_j\}$$

using timestamps $t_0 < t_1 < t_2 < \dots$. This defines a run $\bar{\mathcal{R}} = \bar{S}_0, \bar{S}_1, \dots$ of $\mathcal{CM}_2$ with $\bar{S}_0 = S_0$ and $\bar{S}_{\ell+1} = \bar{S}_\ell + \Delta_\ell$, which implies that $\mathcal{R}'$ is serial.

Furthermore, the runs $\mathcal{R}$ and $\bar{\mathcal{R}}$ contain exactly the same requests and responses, for each agent a the sequence of its requests and responses is identical in both runs, and hence $\mathcal{R}$ and $\bar{\mathcal{R}}$ are view equivalent.

Conversely, for a run $\mathcal{R}'$ of $\mathcal{CM}_2$ and a view equivalent serial run $\mathcal{R}' = S_0, S_1, \dots$ it suffices to show that there exists a subsequence of a flattening $\mathcal{R}' = S'_0, S'_1, S'_2, \dots$ that is a run of $\{(a, asm^c_a)\}_{a \in \mathcal{A}} \cup \{(db, asm_{db})\}$ such that for each agent $a \in \mathcal{A}$ the agent a-view of $\mathcal{R}'$ coincides with a flat view of $\mathcal{R}$ by a. For this we only have to consider states, in which a request or a response is issued to obtain the desired subsequence, and the flattening is defined by the answers to the write requests, which follows immediately from $\mathcal{R}$ being serial. □

5 Concluding Remarks

Concurrent ASMs (cASMs) have been introduced to show that truly concurrent algorithms can be captured by an extension of ASMs [4]. In particular, cASMs overcome limitations in the theory of concurrency associated with the presence of interleaving and unspecified selection agents that enable several agents to perform steps synchronously. This specification and refinement study in this paper shows that concurrent ASMs are well suited to capture all requirements in distributed, concurrent systems.

We demonstrated the application of concurrent communicating ASMs (ccASMs) [5] for the specification, refinement and consistency analysis of concurrent systems in connection with shared replicated memory. We first specified a ground model, in which all access to replicas is handled synchronously in parallel by a single agent, then refined it addressing the internal communication in the memory management subsystem. This refinement significantly changed the way requests are handled, as replicas are not selected a priori in a way that complies with the read- or write-policies, but instead the acknowledgement and return of a response depends on these policies. These refinements could be taken further to capture also the means for handling inactive nodes and for recovery. Due to space limitations for this conference version some explanations remain terse and proofs are only sketched, but details are available in an extended technical report [17].

We further showed that consistency, formalised by the notion of view compatibility, cannot be preserved by the last refinement. We could show that even such a rather weak notion of consistency can only be obtained, if view serialisability is assured. Serialisability can be achieved by adopting transactions for at least single requests. For instance, one might integrate a transactional concurrent system [6] with the specification of a replicative storage system as done in this paper.

References

1. Agha, G.: A Model of Concurrent Computation in Distributed Systems. MIT Press, Cambridge (1986)
2. Best, E.: Semantics of Sequential and Parallel Programs. Prentice Hall, Upper Saddle River (1996)
3. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms. ACM Trans. Comput. Logic **4**(4), 578–651 (2003)
4. Börger, E., Schewe, K.-D.: Concurrent abstract state machines. Acta Inf. **53**(5), 469–492 (2016)
5. Börger, E., Schewe, K.-D.: Communication in abstract state machines. J. Univ. Comp. Sci. **23**(2), 129–145 (2017)
6. Börger, E., Schewe, K.-D., Wang, Q.: Serialisable multi-level transaction control: a specification and verification. Sci. Comput. Program. **131**, 42–58 (2016)
7. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-642-18216-7
8. Ferrarotti, F., Schewe, K.-D., Tec, L., Wang, Q.: A new thesis concerning synchronised parallel computing - simplified parallel ASM thesis. Theor. Comp. Sci. **649**, 25–53 (2016)
9. Genrich, H.J., Lautenbach, K.: System modelling with high-level Petri nets. Theor. Comput. Sci. **13**, 109–136 (1981)
10. Gurevich, Y.: Sequential abstract-state machines capture sequential algorithms. ACM Trans. Comput. Logic **1**(1), 77–111 (2000)
11. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
12. Lynch, N.: Distributed Algorithms. Morgan Kaufmann, USA (1996)
13. Mazurkiewicz, Antoni: Trace theory. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) ACPN 1986. LNCS, vol. 255, pp. 278–324. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-17906-2_30
14. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Springer, New York (2011). https://doi.org/10.1007/978-1-4419-8834-8
15. Rabl, T., Sadoghi, M., Jacobsen, H.-A., Gómez-Villamor, S., Muntés-Mulero, V., Mankowskii, S.: Solving big data challenges for enterprise application performance management. PVLDB **5**(12), 1724–1735 (2012)

16. Schewe, K.-D., Ferrarotti, F., Tec, L., Wang, L., An, W.: Evolving concurrent systems - behavioural theory and logic. In: Proceedings of the Australasian Computer Science Week Multiconference (ACSW 2017), pp. 77:1–77:10. ACM (2017)
17. Schewe, K.-D., Prinz, A., Börger, E.: Concurrent computing with shared replicated memory. CoRR, abs/1902.04789 (2019)
18. Tanenbaum, A.S., Van Steen, M.: Distributed Systems. Prentice-Hall, Upper Saddle River (2007)
19. Winskel, G., Nielsen, M.: Models for concurrency. In: Abramsky, S., Gabbay, D., Maibaum, T.S.E. (eds.) Handbook of Logic and the Foundations of Computer Science: Semantic Modelling, vol. 4, pp. 1–148. Oxford University Press, Oxford (1995)

MRSLICE: Efficient RkNN Query Processing in SpatialHadoop

Francisco García-García[1], Antonio Corral[1](✉), Luis Iribarne[1], and Michael Vassilakopoulos[2]

[1] Department of Informatics, University of Almeria, Almeria, Spain
{paco.garcia,acorral,liribarn}@ual.es

[2] Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece
mvasilako@uth.gr

Abstract. Nowadays, with the continuously increasing volume of spatial data, it is difficult to execute spatial queries efficiently in spatial data-intensive applications, because of the limited computational capability and storage resources of centralized environments. Due to that, shared-nothing spatial cloud infrastructures have received increasing attention in the last years. SpatialHadoop is a full-edged MapReduce framework with native support for spatial data. SpatialHadoop also supports spatial indexing on top of Hadoop to perform efficiently spatial queries (e.g., k-Nearest Neighbor search, spatial intersection join, etc.). The Reverse k-Nearest Neighbor (RkNN) problem, i.e., finding all objects in a dataset that have a given query point among their corresponding k-nearest neighbors, has been recently studied very thoroughly. *RkNN* queries are of particular interest in a wide range of applications, such as decision support systems, resource allocation, profile-based marketing, location-based services, etc. In this paper, we present the design and implementation of an *RkNN* query MapReduce algorithm, so-called *MRSLICE*, in SpatialHadoop. We have evaluated the performance of the *MRSLICE* algorithm on SpatialHadoop with big real-world datasets. The experiments have demonstrated the efficiency and scalability of our proposal in comparison with other *RkNNQ* MapReduce algorithms in SpatialHadoop.

Keywords: RNNQ · SpatialHadoop · MapReduce · Spatial data processing

1 Introduction

Large-scale data analysis and processing is currently the core of many scientific research groups and enterprises. Nowadays, with the development of modern mobile applications, the increase of the volume of available spatial data is huge world-wide. Recent developments of big spatial data systems have motivated the emergence of novel technologies for processing large-scale spatial data on clusters of computers in a distributed environment. Parallel and distributed computing using shared-nothing clusters on extreme-scale data is becoming a dominant trend in the context of data processing and analysis. MapReduce [3] is a framework for processing and managing large-scale datasets in a distributed cluster. MapReduce was introduced with the goal of supplying a simple yet powerful parallel and distributed computing paradigm, providing scalability mechanisms.

However, MapReduce has weaknesses related to efficiency when it needs to be applied to spatial data. A main deficiency is the lack of an indexing mechanism that would allow selective access to specific regions of spatial data, which would in turn yield more efficient spatial query processing algorithms. A recent solution to this problem is an extension of Mapreduce, called *SpatialHadoop* [4], which is a mature and robust framework that inherently supports spatial indexing on top of Hadoop. Moreover, the generated spatial indexes enable the design of efficient spatial query processing algorithms that access only part of the data and still return the correct result query. That is, SpatialHadoop is an efficient MapReduce distributed spatial query processing system that supports spatial indexing and allows to work on distributed spatial data without worrying about computation distribution and fault-tolerance.

A Reverse k-Nearest Neighbor query (*RkNNQ*) [8] returns the data objects that have the query object in the set of their k-nearest neighbors. It is the complementary problem to that of finding the k-Nearest Neighbors (kNN) of a query object. The goal of a *RkNNQ* is to identify the *influence* of a query object on the whole dataset; several real examples are shown in [8]. Although the *RkNN* problem is the complement of the kNN problem, the relation between *kNNQ* and *RkNNQ* is not symmetric and the number of the RkNNs of a query object is not known in advance. A naive solution to the *RkNN* problem requires $O(n^2)$ time, since the k-nearest neighbors of all of the n objects in the dataset have to be found [8]. Obviously, more efficient algorithms are required, and thus, the *RkNN* problem has been studied extensively in the past few years [8,9,13]. As shown in a recent experimental study [14], SLICE [15] is the state-of-the art *RkNN* algorithm for two dimensional location data, since it is the best algorithm in terms of CPU cost. Most of the research works in this topic have been devoted to improve the performance of this query by proposing efficient algorithms in centralized environments [14]. But, with the fast increase in the scale of the big input datasets, processing such datasets in parallel and distributed frameworks is becoming a popular practice. For this reason, parallel and distributed algorithms for *RkNNQ* [1,6,7] have been designed and implemented in MapReduce, and a naive approach [5] has been implemented in SpatialHadoop and LocationSpark.

Motivated by the above observations, in this paper, we propose a novel MapReduce version of the SLICE algorithm (the fastest *RkNNQ* algorithm) in SpatialHadoop, called **MRSLICE**. The most important contributions of this paper are the following:

– The design and implementation of a novel *RkNNQ* MapReduce algorithm, called *MRSLICE*, in SpatialHadoop for efficient parallel and distributed *RkNNQ* processing on big real-world spatial datasets.

– The execution of a set of experiments for examining the efficiency and the scalability of *MRSLICE*, against other *RkNNQ* MapReduce algorithms in SpatialHadoop. *MRSLICE* has shown an excellent performance for all considered performance parameters.

This paper is organized as follows. In Sect. 2, we review related work on *RkNNQ* algorithms and provide the motivation of this paper. In Sect. 3, we present preliminary concepts related to *RkNNQ*, SLICE algorithm and SpatialHadoop. In Sect. 4, the *MRSLICE*, in SpatialHadoop is proposed. In Sect. 5, we present the most representative results of the experiments that we have performed, using real-world datasets. Finally, in Sect. 6, we provide the conclusions arising from our work and discuss related future work directions.

2 Related Work and Motivation

Researchers, developers and practitioners worldwide have started to take advantage of the cluster-based systems and shared-nothing cloud infrastructures to support large-scale data processing. There exist several cluster-based systems that support spatial query processing over distributed spatial datasets. One of the most representative is *SpatialHadoop* [4], which is an extension of the Hadoop-MapReduce framework, with native support for spatial data.

RkNNQ processing has been actively investigated in centralized environments, and here we review the most relevant contributions. In [8], *RkNNQ* was first introduced. Processing is based on a pre-computation process (for each data point $p \in \mathbb{P}$ the k-Nearest Neighbor, *kNN(p)*, is pre-computed and its distance is denoted by *kNNdist(p)*) and has three phases: *pruning*, *containment* and *verification*. In the *pruning* phase, for each $p \in \mathbb{P}$ a circle centered at p with radius *kNNdist(p)* is drawn, and the space that cannot contain any *RkNN* is pruned by using the query point q. In the *containment* phase, the objects that lie within the unpruned space are the *RkNN* candidates. Finally, in the *verification* phase, a *range query* is issued for each candidate to check if the query point is one of its *kNN* or not. That is, for any query point q, determine all the circles *(p, kNNdist(p))* that contain q and return their centers p. In [11], the Six-Regions algorithm is presented, and the need for any pre-computation is eliminated by utilizing some interesting properties of *RkNN* retrieval. The authors solve *RkNNQ* by dividing the space around the query point into six equal partitions of 60° each (R_1 to R_6). In each partition R_i, the *k-th* nearest neighbor of the query point defines the pruned area. In [9] the multistep *SFT* algorithm is proposed. It: (1) finds (using an R-tree) the *kNNs* of the query point q, which constitute the initial candidates; (2) eliminates the points that are closer to some other candidate than q; and (3) applies *boolean range queries* on the remaining candidates to determine the actual *RNNs*. In [12], the *TPL* algorithm which uses the property of perpendicular bisectors located between the query point for facilitating pruning the search space is presented. In the containment phase, TPL retrieves the objects that lie in the area not pruned by any combination of k bisectors. Therefore, TPL has to consider each combination of k bisectors. To overcome the shortcomings of this algorithm, a new method named *FINCH* is proposed in [13]. Instead of using bisectors to prune the objects, the authors use a convex polygon that approximates the unpruned area. *Influence Zone* [2] is a half-space based technique proposed for *RkNNQ*, which uses the concept of *influence zone* to significantly improve the verification phase. Influence zone is the area such that a point p is a *RkNN* of q if and only if p lies inside it. Once influence zone is computed, *RkNNQ* can be answered by locating the points lying inside it. In [15], the *SLICE* algorithm is proposed, which improves the filtering power of Six-Regions approach while utilizing its strength of being a cheaper filtering strategy. Recently, in [14] a comprehensive set of experiments to compare some of the most representative and efficient *RkNNQ* algorithms under various settings is presented and the authors propose an optimized version of TPL (called TPL++) for arbitrary dimensionality *RkNNQs*. SLICE is the state-of-the art *RkNNQ* algorithm, since it is the best for all considered performance parameters in terms of CPU cost.

There is not much work in developing efficient *RkNNQ* algorithms in parallel and distributed environments. The only contributions that have been implemented in MapReduce frameworks are [1,5–7]. In [1], the MRVoronoi algorithm is presented, which adopts the Voronoi diagram partitioning-based approach and applies MapReduce to answer *RNNQ* and other queries. In [6], the Basic MapReduce *RkNNQ* method based on the *inverted grid index* over large scale datasets is investigated. An optimization method, Lazy-MapReduce *RkNNQ* algorithm, that prunes the search space when all data points are discovered, is also proposed. In [7] several improvements of [6] have been presented. For instance, a novel decouple method is proposed to decomposes pruning-verification into independent steps and it can increase opportunities for parallelism. Moreover, new optimizations to minimize the network and disk input/output cost of distributed processing systems have been also investigated. Recently, in [5], parallel and distributed *RkNNQ* algorithms have been proposed for SpatialHadoop and LocationSpark. These parallel and distributed algorithms are based on the multistep *SFT* algorithm [9]. The experimental results demonstrated that LocationSpark is the overall winner for the execution time, due to the efficiency of in-memory processing provided by Spark. However, *MRSFT*, the SpatialHadoop version, shows interesting performance trends due to the nature of the proposed *RkNNQ* MapReduce algorithm, since it consists of a series of MapReduce jobs.

As we have seen above, the SLICE algorithm is the fastest algorithm for *RkNNQ* [14], and there is no MapReduce design and implementation of such an algorithm in parallel and distributed frameworks. Moreover, *RkNNQs* have received significant research attention in the past few years for centralized environments, but not for parallel and distributed data management systems. Motivated by these observations, the efficient design and implementation of *MRSLICE* (MapReduce SLICE version) in SpatialHadoop is the main objective of this research work.

3 Preliminaries and Background

In this section, we first present the basic definitions of the *kNNQ* and *RkNNQ*, followed by a brief introduction of preliminary concepts of SpatialHadoop and, next, we review the most relevant details of the SLICE algorithm for *RkNNQ*.

3.1 The Reverse *k*-Nearest Neighbor Query

We know the *RkNNQ* retrieves the data points which have the query point as one of their respective *kNNs*. We can deduce that the *RkNNQ* is based on the *kNNQ*, and we are going to define it.

Given a set of points $\mathbb{P}$, the *kNNQ* discovers the k points that are the nearest to a given query point q (i.e., it reports only the top-k points of $\mathbb{P}$ from q). It is one of the most important and studied spatial operations. The formal definition of the *kNNQ* for points is the following:

Definition 1. k**-Nearest Neighbor query,** ***kNN***
Let $\mathbb{P} = \{p_1, p_2, \cdots, p_n\}$ be a set of points in E^d (d-dimensional Euclidean space). Then, the result of the k-Nearest Neighbor query, with respect to a query point q in E^d and a number $k \in \mathbb{N}^+$, is an ordered set, $kNN(\mathbb{P}, q, k) \subseteq \mathbb{P}$, which contains the k $(1 \leq k \leq |\mathbb{P}|)$ different points of $\mathbb{P}$, with the k smallest distances from q:
$kNN(\mathbb{P}, q, k) = \{p_1, p_2, \cdots, p_k\} \subseteq \mathbb{P}$, such that $\forall p \in \mathbb{P} \setminus kNN(\mathbb{P}, q, k)$ we have $dist(p_i, q) \ \leq \ dist(p, q),\ 1 \leq i \leq k$.

For *RkNNQ*, given a set of points $\mathbb{P}$ and a query point q, a point p is called the *Reverse k Nearest Neighbor* of q, if q is one of the k closest points of p. A *RkNNQ* issued from point q returns all the points of $\mathbb{P}$ whose k nearest neighbors include q. Note that, this query is also called *Monochromatic RkNNQ* [8]. Formally:

Definition 2. Reverse k**-Nearest Neighbor query,** ***RkNN*** [13]
Let $\mathbb{P} = \{p_1, p_2, \cdots, p_n\}$ be a set of points in E^d. Then, the result of the Reverse k-Nearest Neighbor query, with respect to a query point q in E^d and a number $k \in \mathbb{N}^+$, is a set, $RkNN(\mathbb{P}, q, k) \subseteq \mathbb{P}$, which contains all the points of $\mathbb{P}$ whose k nearest neighbors include q:
$RkNN(\mathbb{P}, q, k) = \{p_i \in \mathbb{P}$, such that $q \in kNN(\mathbb{P} \cup q, p_i, k)\}$

3.2 SpatialHadoop

SpatialHadoop [4] is a fully fledged MapReduce framework with native support for spatial data. It is an efficient disk-based distributed spatial query processing system. Note that MapReduce [3] is a scalable, flexible and fault-tolerant programming framework for distributed large-scale data analysis. A task to be performed using the MapReduce framework has to be defined as two phases: the *Map* phase, which is specified by a *Map function*, takes input (typically from Hadoop Distributed File System (HDFS) files), possibly performs some computations on this input, and distributes it to worker nodes; and the *Reduce* phase which processes these results as specified by a *Reduce function*. An important aspect of MapReduce is that both the input and the output of the *Map* step are represented as *key-value pairs*, and that pairs with same key will be processed as one group by the *Reducer*. Additionally, a *Combiner function* can be used to run on the output of *Map* phase and perform some filtering or aggregation to reduce the number of keys passed to the *Reducer*.

SpatialHadoop is a comprehensive extension to Hadoop that injects spatial data awareness in each Hadoop layer, namely, the language, storage, MapReduce, and operations layers. *MapReduce* layer is the query processing layer that runs MapReduce programs, taking into account that SpatialHadoop supports spatially indexed input files. The *Operation* layer enables the efficient implementation of spatial operations, considering the combination of the spatial indexing in the storage layer with the new spatial functionality in the *MapReduce* layer. In general, a spatial query processing in SpatialHadoop consists of four steps [4]: (1) *Preprocessing*, where the dataset is partitioned according to a specific partitioning technique, generating a set of partitions. (2) *Pruning*, when the query is issued, where the master node examines all partitions and prunes (by a *Filter function*) those ones that are guaranteed not to include in any possible result of the spatial query. (3) *Local Spatial Query Processing*, where a local spatial query processing (*Map* function) is performed on each non-pruned partition in parallel on different nodes (machines). And finally, (4) *Global Processing*, where the results are collected from all nodes in the previous step and the final result of the concerned spatial query is computed. A *Combine* function can be applied in order to decrease the volume of data that is sent from the *Map* task. The *Reduce* function can be omitted when the results from the *Map* phase are final.

3.3 SLICE Algorithm

Like most of the *RkNNQ* algorithms, SLICE consists of two phases namely *filtering phase* and *verification phase*. SLICE's filtering phase dominates the total query processing cost [15].

Filtering Phase. SLICE divides the space of a set of points $\mathbb{P}$ around the query point q into multiple equally sized regions based on angle division. The experimental study in [15] demonstrated that the best performance is achieved when the space is divided into 12 equally sized regions. Given a region R and a point $p \in \mathbb{P}$, we can define the half-space that divides them as $H_{p:q}$. The intersection of this half-space with the limits of the region R allows us to obtain the *upper arc* of p w.r.t. R ($r^U_{p:R}$) and the *lower arc* of p w.r.t. R ($r^L_{p:R}$) whose radii meet the condition of $r^U > r^L$. In [15], it is shown that a point p' in the region R can be pruned by the point p if p' lies outside its upper arc, i.e., $dist(p', q) > r^U_{p:R}$. Note that a point $p' \in R$ cannot be pruned by p if p' lies inside its lower arc, i.e., $dist(p', q) < r^L_{p:R}$. The *bounding arc* of a region R, denoted as r^B_R, is the k-th smallest upper arc of that region and it is used to easily prune points or set of points. Note that any point p' that lies in R with $dist(p', q) > r^B_R$ can be pruned by at least k points. A point p is called *significant* for the region R if it can prune points inside it, i.e., only if $r^L_{p:R} < r^B_R$. Therefore, SLICE maintains a list of significant points for each region that will be used in the *verification phase*. The following lemmas are used in this phase to reduce the search space by pruning non significant points.

Lemma 1. *A point $p \in R$ cannot be a significant point of R if $dist(p, q) > 2r^B_R$.*

Proof. Shown in [15] as Lemma 4.

Lemma 2. *A point $p \notin R$ cannot be a significant point of R if $dist(M, p) > r^B_R$ and $dist(N, p) > r^B_R$ where M and N are the points where the bounding arc of R intersects the boundaries of R.*

Proof. Shown in [15] as Lemma 5.

These lemmas can be easily extended to a complex entity e (i.e., e does not contain any significant point), by comparing $mindist(q, e)$ with the bounding arc of each region that overlaps with e.

Verification Phase. First, SLICE tries to reduce the search space by using the following lemma:

Lemma 3. *A point p prunes every point $p' \in R$ for which $dist(p', q) > r^U_{p:R}$ where $0° < maxAngle(p, R) < 90°$.*

Proof. Shown in [15] as Lemma 1.

To do this, each point $p \in \mathbb{P}$ is checked against several derived pruning rules: (1) if $dist(q, p) > r^B_R$, p is not part of the *RkNNQ* answer; (2) if $dist(q, p)$ is smaller than the k-th lower arc of R, p cannot be pruned; and (3) if once the maximum and minimum angles have been calculated of p w.r.t. q, there is at least one region R with $r^B_R > dist(q, p)$, p can be part of the *RkNNQ* answer. Once the search space has been reduced, each candidate point is verified as a result of *RkNNQ* if at most there are k-1 significant points closest to the query object in the region R in which it is located.

4 *MRSLICE* Algorithm in SpatialHadoop

In this section, we present how *RkNNQ* using *SLICE* can be implemented in SpatialHadoop. In general, our parallel and distributed *MRSLICE* algorithm is based on SLICE algorithm [15] and it consists of three of MapReduce jobs:

- **Phase 1**. The *Filtering phase* of SLICE is performed on the partition in which the query object is located.
- **Phase 1.B (optional)**. The filtering process is continued on those partitions that are still part of the search space.

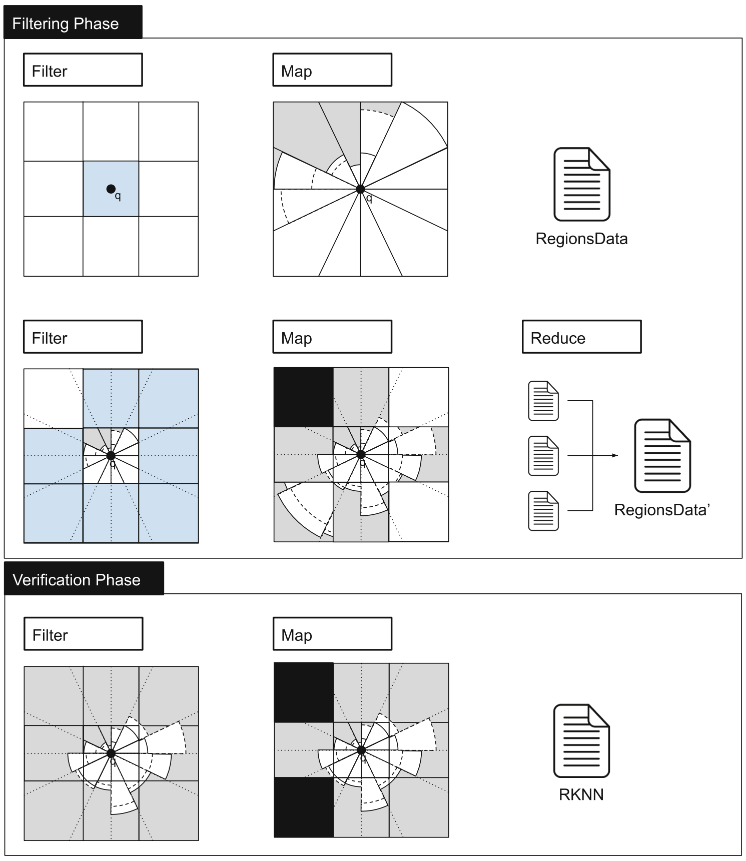


Fig. 1. Overview of *MRSLICE* in SpatialHadoop.

- **Phase 2**. The *Verification phase* is carried out with those partitions that have not been pruned as a result of applying Phases 1 and 1.B.

From Fig. 1, and assuming that $\mathbb{P}$ is the set of points to be processed and q is the query point, the basic idea is to have $\mathbb{P}$ partitioned by some method (e.g., grid) into n blocks or partitions of points ($\mathcal{P}^{\mathbb{P}}$ denotes the set of partitions from $\mathbb{P}$). The *Filtering phase* consists of two MapReduce jobs, being optional the second one, since in the case of all significant points were found by the first job, the execution of the second job is not necessary. Finally, the *Verification phase* is a MapReduce job that will check if the non pruned points are part of the *RkNNQ* answer.

4.1 *MRSLICE* Filtering Algorithm

In the first job (Algorithm 1), the *Filter* function selects the partition of $\mathbb{P}$ in which q is found. Then, in the Map phase, the *Filtering phase* is applied as

Algorithm 1. *MRSLICE* Filtering - Phase 1

```
1: function FILTER($\mathcal{P}^{\mathbb{P}}$: set of partitions from $\mathbb{P}$, $q$: query point)
2:     return FindPartition($\mathcal{P}^{\mathbb{P}}$,$q$)
3: end function

4: function MAP($MBR$: Minimum Bounding Rectangle of $\mathbb{P}$, $r$: root of R-tree of actual partition,
   $q$: query point, $k$: number of points, $t$: number of equally sized regions)
5:     $RegionsData.regions \leftarrow$ DivideSpace($MBR$,$q$,$t$)
6:     $RegionsData \leftarrow$ SliceFiltering($r$,$q$,$k$,$RegionsData$)
7:     return $RegionsData$
8: end function

9: function SliceFiltering($r$: root of R-tree, $q$: query point, $k$: number of points, $RegionsData$:
   SLICE Regions Data)
10:     Insert($Heap$,$null$,$r$)
11:     while $Heap$ is not empty do
12:         $entry \leftarrow$ Pop($Heap$)
13:         if !facilityPruned($entry$,$q$,$k$,$RegionsData$) then
14:             if isLeaf($entry$) then
15:                 pruneSpace($entry$,$q$,$k$,$RegionsData$)
16:             else
17:                 for all $child \in entry.children$ do
18:                     $key \leftarrow$ mindist($q$,$child$)
19:                     Insert($Heap$,$key$,$child$)
20:                 end for
21:             end if
22:         end if
23:     end while
24:     for all $region \in RegionsData.regions$ do
25:         $region.boundingArc \leftarrow$ FindkUpperArc($region$)
26:     end for
27:     $RegionsData.minLowerArc \leftarrow$ ComputeMinLowerArc($RegionsData.regions$)
28:     return $RegionsData$
29: end function
```

Algorithm 2. *MRSLICE* Filtering - Phase 1.B

```
1: function FILTER($\mathcal{P}^{\mathbb{P}}$: set of partitions from $\mathbb{P}$, $q$: query point, $RegionsData$: SLICE Regions
   Data)
2:     for all $p \in \mathcal{P}^{\mathbb{P}}$ do
3:         if !facilityPruned($p$,$q$,$RegionsData$) then
4:             Insert($Result$, $p$)
5:         end if
6:     end for
7:     return $Result$
8: end function

9: function MAP($r$: root of R-tree of actual partition, $q$: query point, $k$: number of points,
   $RegionsData$: SLICE Partition Data)
10:     $RegionsData' \leftarrow$ SliceFiltering($r$,$q$,$k$,$RegionsData$)
11:     return $RegionsData'$
12: end function

13: function REDUCE($RegionsDataArray$: Array of SLICE Partition Data)
14:     $RegionsData' \leftarrow RegionsDataArray[0]$
15:     for all $RegionsData \in RegionsDataArray$ do
16:         $RegionsData'.P \leftarrow$ Merge($RegionsData.regions$, $RegionsData'.P$)
17:     end for
18:     for all $partition \in RegionsData'.P$ do
19:         $partition.kUpperArc \leftarrow$ FindkUpperArc($partition$)
20:     end for
21:     $RegionsData'.minLower \leftarrow$ ComputeMinLower($RegionsData'.P$)
22:     return $RegionsData'$
23: end function
```

described in SLICE, that is, $\mathbb{P}$ is divided into t regions of equal space and the list of k smallest upper arcs is obtained for each R_i region along with its $r^B_{R_i}$ and its list of significant points, that will be returned as *RegionsData* for further use. To accelerate the *Filtering phase*, an R-tree index is used per partition and a *heap* is utilised to store the nodes based on their minimum distance to q. As the R-tree nodes are traversed, the *facilityPruned* function from [15] is used (Algorithm 1 line 13), which prunes the nodes which with the current *RegionsData* do not contain significant points. In the case of leaf nodes, the points are processed by the *pruneSpace* function from [15] (Algorithm 1 line 15), which is responsible for updating the *RegionsData* information. Finally the k-th lower arc is calculated for using in the next phase.

The second job (Algorithm 2) runs only if the function *Filter* returns some partition, that is, the *facilityPruned* [15] function is executed on each of the partitions by comparing its minimum distance to q with the bounding arc of each region R_i with which it overlaps. Note that the upper left partition of $\mathbb{P}$ in Fig. 1 is in the shaded area, and therefore can be pruned. However, the other partitions can contain significant points, and the *Filtering phase* must be applied to them during the *Map phase*. The result of each of the partitions will be merged on the *Reduce phase* to obtain the k-th upper arcs, bounding arcs and final significant points (*RegionsData'*).

Theorem 1 (Completeness). *MRSLICE Filtering Algorithm returns all the significant points.*

Proof. It suffices to show that MRSLICE Filtering does not discard significant points. A point p is discarded by MRSLICE Filtering only if it is pruned by the *facilityPruned* function by either applying Lemma 1 or 2. In any of these cases, it was shown in [15] that any point that is not inside the area defined by these lemmas is not a significant point. Points that are discarded can be split in different categories:

Phase 1. Points are pruned in this phase like in the non distributed SLICE version using Algorithm 1.

Phase 1.B - Partition granularity. Using the *FILTER* function in Algorithm 2, partitions that do not contain any significant point are pruned by applying both Lemmas 1 and 2 to the partition as a complex entity.

Phase 1.B - Point granularity. Points are discarded in the Map Phase in the same way that in *Phase 1* only on non pruned partitions.

Phase 1.B - Merging RegionsData. Finally when merging RegionsData in the Reduce Phase, both Lemmas 1 and 2 are again used to discard non significant points.

4.2 *MRSLICE* Verification Algorithm

Finally, in the *Verification phase*, a MapReduce job (Algorithm 3) is executed on the partitions that are not pruned by the *Filter* function when applying the pruning rules described in the Subsect. 3.3 in the *userPruned* function (Algorithm 3

Algorithm 3. *MRSLICE* Verification - Phase 2

```
 1: function FILTER($\mathcal{P}^{\mathbb{P}}$: set of partitions from $\mathbb{P}$, $q$: query point, $RegionsData$: SLICE partition
    data)
 2:     for all $p \in \mathcal{P}^{\mathbb{P}}$ do
 3:         if !userPruned($p$,$q$,$RegionsData$) then
 4:             Insert($Result$, $p$)
 5:         end if
 6:     end for
 7:     return $Result$
 8: end function

 9: function MAP($r$: root of R-tree of actual partition, $q$: query point, $k$: number of points,
    $RegionsData$: SLICE Partition Data)
10:     Insert($Stack$,$r$)
11:     while $Stack$ is not empty do
12:         $entry \leftarrow$ Pop($Stack$)
13:         if !userPruned($entry$,$q$,$RegionsData$) then
14:             if isLeaf($entry$) then
15:                 if isRkNN($entry$,$q$,$k$,$RegionsData$) then
16:                     Output($entry$)
17:                 end if
18:             else
19:                 for all $child \in entry.children$ do
20:                     Insert($Stack$,$child$)
21:                 end for
22:             end if
23:         end if
24:     end while
25: end function

26: function isRkNN($entry$: candidate point, $q$: query point, $k$: number of points, $RegionsData$:
    SLICE Partition Data)
27:     $region \leftarrow$ FindRegion($entry$,$q$,$RegionsData$)
28:     $counter \leftarrow 0$
29:     for all $p \in region$ do
30:         if dist($entry$,$q$) $\leq r^{L}_{p:R}$ then
31:             return true
32:         end if
33:         if dist($entry$,$p$) $<$ dist($entry$,$q$) then
34:             $counter \leftarrow counter + 1$
35:             if $counter \geq k$ then
36:                 return false
37:             end if
38:         end if
39:     end for
40:     return true
41: end function
```

line 3). That is, the algorithm is executed on those partitions that contain some white area. In the *Map phase*, the R-tree, that indexes each partition, is traversed with the help of a *stack* data structure and the search space is reduced by using the *userPruned* function again. Furthermore, the pruning rules are applied again to the points that are in the leaf nodes and, finally, they are verified if they are part of the final *RkNNQ* answer. The *isRkNN* function (Algorithm 3 line 15) verifies a candidate point p as part of the answer if there are at most k-1 significant points closer to p than q in the region R_i in which it is located.

Theorem 2 (Correctness). *MRSLICE Verification Algorithm returns the correct RkNNQ set.*

Proof. It suffices to show that *MRSLICE* Verification does not (a) discard *RkNNQ* points, and (b) return non *RkNNQ* points. First, the *MRSLICE* Verification Algorithm only prunes away those points or/and entries by using the pruning rules derived from Lemma 3, by using the information identified by the *MRSLICE* Filtering Algorithm, which guarantees no false negatives. Second, every non pruned point is verified by the *isRkNN* function, which ensures no false positives. We prove that these points are guaranteed to be *RkNNQ* points by contradiction. Assume a point p returned by *MRSLICE* Algorithm is not a *RkNNQ* point. Then, there exist k significant points closer to p than q, and p is also returned as part of the *RkNNQ* answer. But then p could not be in the *RkNNQ* answer, since it would have been evicted in line 35 of the *isRkNN* function in Algorithm 3.

5 Experimentation

In this section, we present the most representative results of our experimental evaluation. We have used real-world 2d point datasets to test our *RkNNQ* algorithms, that is, our previous *MRSFT* based algorithm [5] and the new *MRSLICE* algorithm in SpatialHadoop. We have used datasets from OpenStreetMap[1]: *LAKES* (L) which contains 8.4M records (8.6 GB) of boundaries of water areas (polygons), *PARKS* (P) which contains 10M records (9.3 GB) of boundaries of parks or green areas (polygons), *ROADS* (R) which contains 72M records (24 GB) of roads and streets around the world (line-strings), *BUILDINGS* (B) which contains 115M records (26 GB) of boundaries of all buildings (polygons), and *ROAD_NETWORKS* (RN) which contains 717M records (137 GB) of road network represented as individual road segments (line-strings) [4]. To create sets of points from these five spatial datasets, we have transformed the MBRs of line-strings into points by taking the center of each MBR. In particular, we have considered the centroid of each polygon to generate individual points for each kind of spatial object. Furthermore, all datasets have been previously partitioned by SpatialHadoop using the STR partitioning technique with a local R-tree index per partition. The main performance measure that we have used in our experiments has been the total execution time (i.e., total response time). In order to get a representative execution time, a random sample of 100 points from the smallest dataset (*LAKES*) has been obtained and the average of the execution time of the *RkNNQ* of these points has been calculated, since this query depends a lot on the location of the query point with respect to the dataset.

All experiments were conducted on a cluster of 12 nodes on an OpenStack environment. Each node has 4 vCPU with 8 GB of main memory running Linux operating systems and Hadoop 2.7.1.2.3. Each node has a capacity of 3 vCores for MapReduce2/YARN use. Finally, we used the latest code available in the repositories of SpatialHadoop[2].

[1] Available at http://spatialhadoop.cs.umn.edu/datasets.html.
[2] Available at https://github.com/aseldawy/spatialhadoop2.

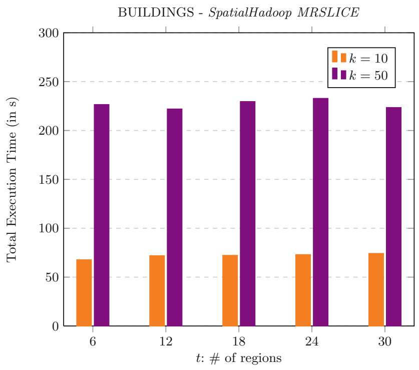


Fig. 2. *MRSLICE* execution times considering different t values.

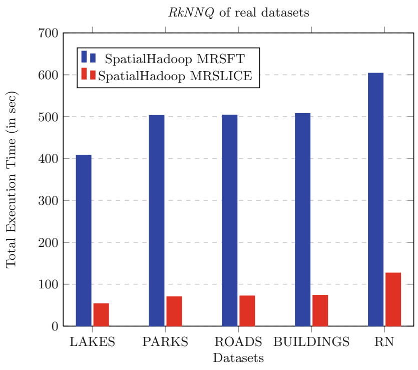
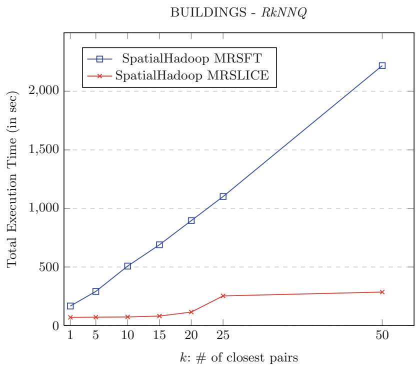


Fig. 3. *RkNNQ* execution times considering different datasets.

The first experiment aims to test the best t value (number of regions) for *MRSLICE*, using the *BUILDINGS* dataset and the k values of 10 and 50. In Fig. 2 we can see that there is a little difference in the results obtained when the t value is varied, especially for $k = 10$, being greater differences when a larger k value is used (i.e., $k = 50$). On one hand, for $k = 10$, smaller values of t get faster times (e.g., $t = 6$ has an execution time of 67 s which is 4 s faster than $t = 12$). On the other hand, for $k = 50$, $t = 12$ gets the smallest execution time (221 s) and for $t < 12$ and $t > 12$, the execution time increases. Although there are no large differences, the value of t that shows better performance for both k values is $t = 12$, reaching the same conclusion as in [15] but now in a distributed environment (from now on, we will use $t = 12$ in all our experiments).

Our second experiment studies the scalability of the *RkNNQ* MapReduce algorithms (*MRSLICE* and *MRSFT* [5]), varying the dataset sizes. As shown in Fig. 3 for the *RkNNQ* of real datasets (*LAKES*, *PARKS*, *ROADS*, *BUILDINGS* and *RN*) and a fixed $k = 10$. The execution times of *MRSLICE* are much lower than those from *MRSFT* (e.g., it is 477 s faster for the largest dataset *RN*) thanks to how the search space is reduced and the limited number of MapReduce jobs. Note that for *MRSFT* at least $k * 20 + 1$ jobs are executed while for the case of *MRSLICE*, 3 jobs are launched at most. In both algorithms, the execution times do not increase too much, showing quite stable performance, mainly for *MRSLICE*. This is due to the indexing mechanisms provided by SpatialHadoop that allow fast access to only the necessary partitions for the query processing. Furthermore, this behavior shows that the number of candidates for *MRSLICE* is almost constant (the expected number of candidates is less than $3.1 * k$ as stated in [15]), only showing a visible increment in the execution time for the *RN* dataset, due to the increase in the density of partitions and its distribution causes the need to execute the optional job (phase 1.b) of the *Filtering phase*.



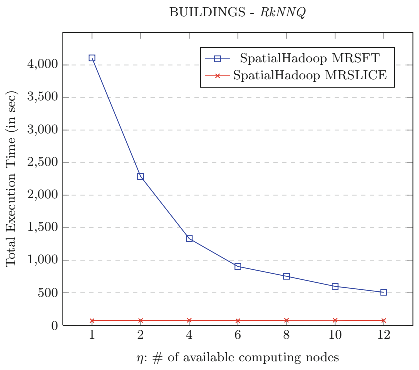


Fig. 4. *RkNNQ* cost (execution time) vs. K values (left). Query cost with respect to the number of computing nodes η (right).

The third experiment aims to measure the effect of increasing the k value for the dataset ($BUILDINGS$). The left chart of Fig. 4 shows that the total execution time grows as the value of k increases, especially for $MRSFT$. This is because as the value of k increases, the number of candidates $k * 20$ also grows and for each of them a MapReduce job is executed. On the other hand, $MRSLICE$ limits the number of MapReduce jobs to 3, obtaining a much smaller increment and more stable results since the disk accesses are reduced significantly by traversing the index of the dataset a reduced number of times. Note that the small increment in the execution times when $k = 25$, mainly due to the fact that when reaching a certain k value, the result of the first job of the *Filtering phase* is not definitive and it has been necessary to execute the optional job (phase 1.b), in this case the number of involved partitions in the query increases as well. Finally, the execution time for $k = 50$ increases slightly.

The fourth experiment studies the speedup of the *RkNNQ* MapReduce algorithms, varying the number of computing nodes (η). The right chart of Fig. 4 shows the impact of different number of computing nodes on the performance of *RkNNQ* MapReduce algorithms, for $BUILDINGS$ with a fixed value of $k = 10$. From this chart, we can deduce that for $MRSFT$, better performance would be obtained if more computing nodes are added. $MRSLICE$ is still outperforming $MRSFT$ and it is not affected despite reducing the number of available computing nodes. This is because $MRSLICE$ is an algorithm in which both the number of partitions involved in obtaining the result of the query and the number of MapReduce jobs are minimized. That is, depending on the location of the query point q and the k value, the number of partitions is usually one, and varying the number of computing nodes does not affect the execution time. However, the use of the computing resources of the cluster is quite small, which allows us the execution of several *RkNNQs* in parallel, taking advantage of the distribution of the dataset into the cluster nodes. On the other hand, $MRSFT$ executes different *kNNQs* in parallel, using all computing nodes completely for large k values.

By analyzing the previous experimental results, we can extract several conclusions that are shown below:

- We have experimentally demonstrated the *efficiency* (in terms of total execution time) and the *scalability* (in terms of k values, sizes of datasets and number of computing nodes (η)) of the proposed parallel and distributed *MRSLICE* algorithm for *RkNNQ* and we have compared it with the *MRSFT* algorithm in SpatialHadoop.
- As stated in [15], the value of t (the number of equally sized regions in which the dataset is divided) that shows the best performance is 12.
- *MRSLICE* outperforms *MRSFT* several orders of magnitude (around five times faster), thanks to its pruning capabilities and the limited number of MapReduce jobs.
- The larger the k values, the greater the number of candidates to be verified, but for *MRSLICE* the number of jobs and partitions involved are quite restricted and the total execution time increases less than for *MRSFT*.
- The use of computing nodes by *MRSLICE* is small, allowing the execution of several queries in parallel, unlike *MRSFT* that can leave the cluster busy.

6 Conclusions and Future Work

In this paper, we have proposed a novel *RkNNQ* MapReduce algorithm, called *MRSLICE*, in SpatialHadoop, to perform efficient parallel and distributed *RkNNQ* on big spatial real-world datasets. We have also compared this algorithm with our previous *MRSFT* algorithm [5] in order to test its performance. The execution of a set of experiments has demonstrated that *MRSLICE* is the clear winner for the execution time, due to the efficiency of its pruning rules and the reduced number of MapReduce jobs. Furthermore, *MRSLICE* shows interesting performance trends due to the low requirements of computing nodes that allows the execution of multiple *RkNNQs* on large spatial datasets. Our current *MRSLICE* algorithm in SpatialHadoop is an example for the study of regions-based pruning on parallel and distributed environments. Therefore, future work might include the adaptation of half-space pruning algorithms [15] to this kind of environments so as to compare them. Other future work might cover studying other types of *RkNNQs* like the Bichromatic *RkNNQ* [15]. Finally, we are planning to improve the query cost of *MRSLICE* by using the *guardian set* of a rectangular region [10], that improves the original SLICE algorithm.

Acknowledgments. Research of all authors is supported by the MINECO research project [TIN2017-83964-R].

References

1. Akdogan, A., Demiryurek, U., Kashani, F.B., Shahabi, C.: Voronoi-based geospatial query processing with MapReduce. In: CloudCom Conference, pp. 9–16 (2010)

2. Cheema, M.A., Lin, X., Zhang, W., Zhang, Y.: Influence zone: efficiently processing reverse k nearest neighbors queries. In: ICDE Conference, pp. 577–588 (2011)
3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: OSDI Conference, pp. 137–150 (2004)
4. Eldawy, A., Mokbel, M.F.: Spatialhadoop: a mapreduce framework for spatial data. In: ICDE Conference, pp. 1352–1363 (2015)
5. García-García, F., Corral, A., Iribarne, L., Vassilakopoulos, M.: RkNN query processing in distributed spatial infrastructures: a performance study. In: Ouhammou, Y., Ivanovic, M., Abelló, A., Bellatreche, L. (eds.) MEDI 2017. LNCS, vol. 10563, pp. 200–207. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66854-3_15
6. Ji, C., Hu, H., Xu, Y., Li, Y., Qu, W.: Efficient multi-dimensional spatial RkNN query processing with MapReduce. In: ChinaGrid Conference, pp. 63–68 (2013)
7. Ji, C., Qu, W., Li, Z., Xu, Y., Li, Y., Wu, J.: Scalable multi-dimensional RNN query processing. Concurrency Comput.: Pract. Experience **27**(16), 4156–4171 (2015)
8. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: SIGMOD Conference, pp. 201–212 (2000)
9. Singh, A., Ferhatosmanoglu, H., Tosun, A.S.: High dimensional reverse nearest neighbor queries. In: CIKM Conference, pp. 91–98 (2003)
10. Song, W., Qin, J., Wang, W., Cheema, M.A.: Pre-computed region guardian sets based reverse kNN queries. In: DASFAA Conference, pp. 98–112 (2016)
11. Stanoi, I., Agrawal, D., El Abbadi, A.: Reverse nearest neighbor queries for dynamic databases. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 44–53 (2000)
12. Tao, Y., Papadias, D., Lian, X.: Reverse kNN search in arbitrary dimensionality. In: VLDB Conference, pp. 744–755 (2004)
13. Wu, W., Yang, F., Chan, C.Y., Tan, K.: FINCH: evaluating reverse k-nearest-neighbor queries on location data. PVLDB **1**(1), 1056–1067 (2008)
14. Yang, S., Cheema, M.A., Lin, X., Wang, W.: Reverse K nearest neighbors query processing: Experiments and analysis. PVLDB **8**(5), 605–616 (2015)
15. Yang, S., Cheema, M.A., Lin, X., Zhang, Y.: SLICE: reviving regions-based pruning for reverse k nearest neighbors queries. In: ICDE Conference, pp. 760–771 (2014)

Should We Be Afraid of Querying Billions of Triples in a Graph-Based Centralized System?

Abdallah Khelil[1,3](✉), Amin Mesmoudi[1,2], Jorge Galicia[1], and Mohamed Senouci[3]

[1] LIAS/ISAE-ENSMA, Chasseneuil-du-Poitou, France
{abdallah.khelil,jorge.galicia}@ensma.fr

[2] University of Poitiers, Poitiers, France
amin.mesmoudi@univ-poitiers.fr

[3] University Oran 1, Oran, Algeria
m.senouci@mesrs.dz

Abstract. Data representation facilities offered by RDF (Resource Description Framework) have made it very popular. It is now considered as a standard in several fields (Web, Biology, ...). Indeed, by lightening the notion of schema, RDF allows a flexibility in the representation of data. This popularity has given rise to large datasets and has consequently led to the need for efficient processing of these data. In this paper, we propose a novel approach that we name QDAG (Querying Data as Graphs) allowing query processing on RDF data. We propose to combine RDF graph exploration with physical fragmentation of triples. Graph exploration makes possible to exploit the structure of the graph and its semantics while the fragmentation allows to group the nodes of the graph having the same properties. Compared to the state of the art (i.e., gStore, RDF3X, Virtuoso), our approach offers a compromise between efficient query processing and scalability. In this regard, we conducted an experimental study using real and synthetic datasets to validate our approach with respect to scalability and performance.

Keywords: RDF · Graph exploration · Fragmentation · Scalability · Performance

1 Introduction

Storage and data collection methods have been largely impacted by recent applications based on modern observation and simulation instruments. In several areas, the pay-as-you-go philosophy has been adopted in the production and storage environments processing data. As a result, the traditional data management approach, defining a schema (a structure) of the data before storing it accordingly, has become very constraining.

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 251–266, 2019.
https://doi.org/10.1007/978-3-030-32065-2_18

In this context, new designs of data representation have emerged. RDF is part of the efforts led by the W3C to depict web data and its linkage. RDF is based on the notion of triples, making statements about resources with expressions of the form $< subject, predicate, object >$. This feature gives flexibility in data collection and its success has resulted in many large-scale datasets, for example Freebase[1] that has 2.5 billion triples [4] and DBpedia[2] with more than 170 million triples [11]. The Linked Open Data Cloud (LOD) now connects over 10,000 datasets and currently has more than 150 billion triples[3]. The number of data sources has doubled in the last three years (2015–2018).

The development of the SPARQL language facilitated the exploitation of RDF data and enforced the growth of novel query processing approaches. Indeed, several studies have shown that conventional relational data management systems are not adapted to manage triples. In particular, they are unable to guarantee scalability and performance. The lack of an explicit RDF schema for a dataset, and the number of joins involved in a SPARQL query are the two main reasons for this limitation.

Among the solutions proposed to process RDF data, we find two types of approaches available in the literature. The first one (e.g. [14,16]) takes advantage of relational database techniques for storage and query evaluation. These works use traditional optimization strategies (e.g. fragmentation and indexing) to process queries on RDF datasets. However, they suffer from many overheads (e.g. large number of self-joins) since a relational database is meant to interrogate data stored according to a relational schema and not linked RDF data necessarily stored as a relation. The second type of approaches (e.g., gStore [17]) considers the query evaluation problem as a graph matching inquiry, maintaining the original representation of the RDF data. Unfortunately, this type of approach does not guarantee scalability and performance.

In this paper, we propose to take advantage of both types of approaches. We rely on indexing and fragmentation to minimize the I/O's and on the exploration of the graph to take into account its structure. Unlike graph matching approaches our evaluation strategy uses the Volcano [9] model, which allows to control the memory use and to avoid any possible overflow. The rest of this paper is organized as follows. We discuss in Sect. 2 the related work. Then, in Sect. 3 we give an overview and formalize of our work. In Sect. 4, we present our experimental study and finally, in Sect. 5 we conclude and discuss some future research.

2 Related Work and Background

The amount of data collected from the Web and other domains like Biology and Astronomy has exponentially grown. The schema of the data collected in these domains is frequently unknown or it is likely to change. Database Management Systems (DBMSs) have been used to store and process these new incoming data,

[1] http://www.freebase.com/.
[2] http://wiki.dbpedia.org.
[3] http://lodstats.aksw.org/.

however they do not guarantee good performances for data without a pre-defined scheme. Besides, in these domains the data are frequently represented as graphs. The RDF model, originally developed to describe Web resources, represents data as triples interconnected with directed links as shown in Fig. 1a.

SPARQL, the standard query language for RDF graphs, use graph patterns (e.g. Fig. 1c) to retrieve and manipulate data stored in an RDF graph. The solution to a SPARQL query is found traversing the graph from a node's forward or backward edges. The approaches built on top of the relational model lose the notion of graph traversal and are constrained to join many tables (that could be the same table) to find a solution. In this section we briefly overview the most relevant approaches.

$$\begin{tikzcd}
 & y_1 \arrow[r, "d"] \arrow[dr, "e"] & k_1 \\
x_1 \arrow[ur, "a"] \arrow[r, "b"] \arrow[dr, "c"] & z_1 & n_1 \\
x_2 \arrow[r, "c"] \arrow[dr, "a"] & l_1 & m_1 \arrow[d, "h"] \\
 & y_2 & p_1
\end{tikzcd}$$

(a) Logical Graph

S	**P**	**O**
x_1	a	y_1
x_1	b	z_1
x_1	c	l_1
x_2	c	l_1
x_2	a	x_1
y_1	d	k_1
y_1	e	n_1
m_1	h	p_1

(b) Triple table

$$\begin{tikzcd}
 & ?y \arrow[r, "d"] \arrow[dr, "e"] & ?k \\
?x \arrow[ur, "a"] \arrow[r, "c"] \arrow[dr, "b"] & ?z & ?n \\
 & ?l &
\end{tikzcd}$$

(c) Query

Fig. 1. Examples of RDF representation

The first approach used to treat RDF data proposed to store and query triples based on traditional DBMSs [6]. In this strategy the data are stored in a single very large table of three attributes that correspond to Subject, Predicate and Object (e.g. Fig. 1b). The problem with this approach is the number of generated self-joins. Indeed, there is a join for each graph pattern and the necessity to scan the entire table many times has a negative impact in the performance.

The second category of approaches (e.g. RDF-3x [14], Hexastore [16]) rely on excessive indexing of data. In these systems, strings of triples are mapped to ids. These ids are then stored and indexed using different orders (e.g. SPO, OPS, PSO). Each order is stored as a clustered B+-Tree. In these systems, triple patterns can be answered by a range query and the joins are very effective through the use of merge joins on indexed data. The drawbacks of this approach are: (1) the use of extra space due to excessive data replication, and (2) the exorbitant cost generated by the complete analysis of the structures used (e.g. SPO).

Another approach, called Property table, was proposed in the framework of Jena [12] and DB2-RDF [5]. A variant of this proposal is called clustered property table which groups the properties that tend to occur in the same (or similar) subjects. Another variant, called Property-class table, clusters the subjects with the same type of property into one property table. This kind of approach allows to avoid some joins for Subject-Subject queries. However, it faces some problems in the processing of other kinds of joins. Also, this approach does not support the storage of subjects with multi-valued properties.

The next approach, based on Binary tables [1], has been proposed to overcome the disadvantage of property tables approaches. Indeed, for each property, a table of two columns is built containing both subject and object, ordered by subjects. This approach is also called vertical partitioned tables. The strategy builds n two-column tables (n is the number of unique properties in the data). Column storage is used to speed up the evaluation of queries. This approach supports multi-valued properties. Only relevant data are stored, *i.e.*, no need to manage `NULL` values. Also, good performances are granted for subject-subject joins. Unfortunately, this approach is not relevant in the case of subject-object joins.

The last category of approaches represents the data as graph. The system gStore [17] for instance, uses an adjacency list to store the properties. The approach is quite similar to the property table, but it uses the adjacency list to skip the null properties of the subject. An index, based on S-Tree [7], is used to speed up the evaluation of graph matching operations. Unfortunately, this type of approach does not control the amount of memory used, which induces an overflow when processing certain queries unlike join-based approaches.

In our work, we adopt a graph storage strategy. As we have queries with constant predicates, we store only forward and backward edges as SPO and OPS. We then split SPO and OPS into many fragments storing a set of subjects (in the case of SPO) or objects (in the case of OPS) that have the same edges (predicates). Each fragment is then stored as a separated clustered B+Tree. We also offer an evaluation mechanism that allows to exploit different types of data structures. Our evaluation operations consider the amount of available memory which allows to avoid any overflow.

3 Our Approach

We begin by presenting our formal framework for Select-Project-Join (SPJ) queries. Our approach only considers queries with constant predicates. We present graph storage techniques and then we discuss the query evaluation techniques used by our system. In our research report[4], we explain how to extend this framework to other queries (*i.e.*, Group by, Order By, ...).

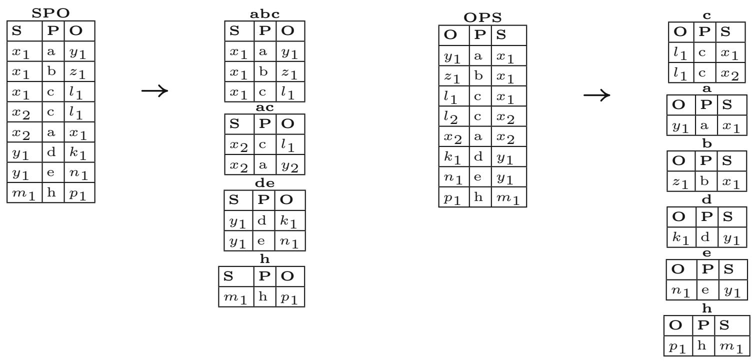
3.1 Graph Storage

Several approaches have been proposed to efficiently manage data graphs. As it was mentioned in last section, RDF-3X [14], HexaStore [16] and Virtuoso [8] propose to store graphs in the form of SPO and OPS. Seen from a graph perspective, storing the data with the SPO format implies storing a node together with

[4] https://www.lias-lab.fr/publications/32823/RapportderechercheKHELIL.pdf.

its forward edges. For example, in Fig. 2a each line represents a node's forward edge (e.g. the node x_1 and its forward edges to the nodes y_1, z_1 and l_1). On the other hand, OPS approaches store a node together with its backward edges.

The choice of the storage structure contributes at query runtime, as confirmed by the performance differences of the approaches detailed previously. In our work we consider a graph-based data representation. Since our workload only considers queries with constant predicates, we choose to use both SPO and OPS structures.



SPO

S	P	O
x_1	a	y_1
x_1	b	z_1
x_1	c	l_1
x_2	c	l_1
x_2	a	x_1
y_1	d	k_1
y_1	e	n_1
m_1	h	p_1

$\rightarrow$

abc

S	P	O
x_1	a	y_1
x_1	b	z_1
x_1	c	l_1

ac

S	P	O
x_2	c	l_1
x_2	a	y_2

de

S	P	O
y_1	d	k_1
y_1	e	n_1

h

S	P	O
m_1	h	p_1

(a) SPO splitting

OPS

O	P	S
y_1	a	x_1
z_1	b	x_1
l_1	c	x_1
l_2	c	x_2
x_2	a	x_2
k_1	d	y_1
n_1	e	y_1
p_1	h	m_1

$\rightarrow$

c

O	P	S
l_1	c	x_1
l_1	c	x_2

a

O	P	S
y_1	a	x_1

b

O	P	S
z_1	b	x_1

d

O	P	S
k_1	d	y_1

e

O	P	S
n_1	e	y_1

h

O	P	S
p_1	h	m_1

(b) OPS splitting

Fig. 2. Data splitting

SPO and OPS are generally stored as a clustered B+Tree, which allows triples retrieval in $log(n)$ disk access. Full scan is not necessary for selective queries. Storing triples in a single SPO (or OPS) file can lead to certain issues. Basically due to the heterogeneous data found in a single RDF graph. For example, one can find, in the same graph, information on soccer games and also information about stock transactions. In spite of that, SPARQL queries are usually more specific targeting data characterized by a theme defined by the query's predicates.

We propose to split SPO and OPS triples, in fragments (that we later named segments) with respect to the predicates that cover subjects in the case of SPO (or object in the case of OPS). This corresponds to the idea of characteristic sets [13], except that the characteristic sets are used only to collect statistics on data. An example of this splitting for SPO and OPS fragments for the data of Fig. 1a is given in Fig. 2a and b respectively.

Definition 1. *(Segment) A segment is a part of SPO (or OPS) that satisfies a set of predicates.*

In our work, data are physically stored as segments. In the rest of this paper we use the word segment instead of characteristic sets to design the physical split of SPO or OPS. In the case of a (non-selective) query where we have to perform a full scan, only relevant segments (which satisfy the query) are scanned.

Let us consider the following query which is a subset of the one shown in Fig. 1c:

```
SELECT ?x ?y ?z
WHERE { ?x a ?y AND ?x c ?z }
```

To avoid a full scan of all segment files, we firstly check the predicates involved in the query (a and c). Supposing that only SPO segments are available, there are two candidate segment files to be scanned according to Fig. 2a (abc and ac). To answer the query both segments must be scanned. To efficiently find relevant segments to a query, we created an index on segment (SPO or OPS) labels as a lattice [2]. Figure 3 illustrates an example of an SPO lattice for the data splitting of Fig. 2a.

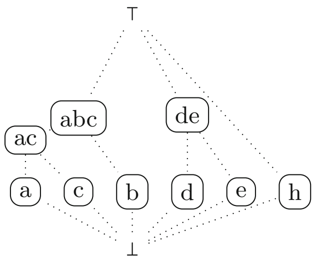


Fig. 3. Example: SPO lattice

In the following section, we propose a technique of compression of the data stored as segments.

Compression: As explained previously, data are fragmented into segments. Each segment is stored as a clustered B+Tree. For each triple, our evaluation methods need to locate the segments hosting the subject and the object of the triple.

Each triple is extended as follows:

$$< node_1, \text{sg}, \text{predicate}, node_2, sg_{in}, sg_{out} >$$

where:

- $node_1$: is the subject in the case of forward triple or object in the case of backward triple.
- sg: is the id of segment storing forward edges in the case where $node_1$ is a subject or backward edges where $node_1$ is an object.
- $node_2$: is the object in the case of forward triple or subject in the case of backward triple.
- sg_{in}: is the segment storing backward edges of $node_2$.
- sg_{out}: is the segment storing forward edges of $node_2$.

Indicator	**Predicate**
17 bits	1 bit

$Node_1$	**Segment**	$Node_2$	**Segment (in)**	**Segment (out)**
0-8 bytes	0-4 bytes	0-8 bytes	0-4 bytes	0-4 bytes

Fig. 4. Compression illustration

$node_1$ and $node_2$ are represented using 64-bit (8 bytes), whereas sg, sg_{in} and sg_{out} are represented using 32 bits (4 bytes). We use only one bit to represent the change of the predicate. Indeed, as data are sorted in the order subject, predicate and object and for all data stars of the segment we have the same set of predicates, we propose to use only one bit to indicate the change of the predicate.

Our approach to data compression is inspired by the one used in RDF-3X [14]. The number of bytes used to represent this information varies according to the number of bytes used to encode each element of the triple. For the $node_1$ we have 9 states. 0 to indicate that the $node_1$ does not change with respect to the previous triple. i $\in$ [1..8] to indicate the number we used to encode the new $node_1$. So, we need 4 bits to represent the 9 states.

For sg, sg_{in} and sg_{out}, we need 5 states. 0 to indicate that this information does not change or `NULL`. i $\in$ [1..4] to indicate the number we used to encode the new segment id. So, we need 3 bits to represent the 5 states. For $node_2$ we use i $\in$ [1..8] to indicate the number we used to encode the new $node_2$. So, we need 4 bits to represent the 8 states. For the predicate we need only 1 bit to indicate the change. We do not need to store the predicate. As shown in Fig. 4, in total we need 18 bits. Those bits represent the indicator of the number of bytes used to encode each triple and only leaf pages are compressed.

B+Tree Access: In this section, we discuss our bulk search strategy using the B+Tree. Our evaluation operators use the B+tree structure to find relevant data organized as data stars defined as:

Definition 2. *(Stars) A data star is a set of nodes and edges related to a given node in the data graph. The set of forward edges related to a node is called a "forward data star". We use the symbol: $SD_f(x)$, where $x \in V_c$, to denote the forward data star obtained from x. In this case, x is called the head of the star. Similarly, the set of incoming nodes and edges to a node in the graph is called a backward data star symbolized as $SD_b(x)$. We apply the same principle to distinguish stars in a query. We use forward query star ($SQ_f(x)$) and backward query star ($SQ_b(x)$), where $x \in V_c \cup V_p$ denote stars obtained from forward and backward triples respectively.*

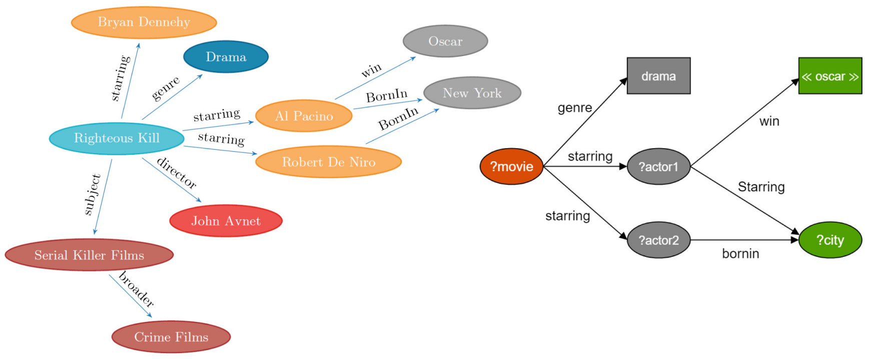
The naive algorithm consists in performing a search on B+Tree for every head we need. If we have n heads, then we need $n * log(n)$ time to find all needed data stars. We changed the searching strategy as follows. We start by finding the first data star. During this operation, we memorize the non leaf keys used. We pass them to the next data stars. We compare with the key previously memorized, and we trigger a partial find if we violate the memorized key. The input of our algorithm is a vector of data star heads and the output is a set of data stars.

3.2 Query Evalaution

In this section we present the structures used to formally describe our evaluation approach. Then, we describe the operations performed by the query execution engine of our system. Let us firstly define a graph database:

Definition 3. *(Graph Database) A Graph Database denoted as* $G = \langle V_c, L_V, E, L_E \rangle$ *where* V_c *is a collection of vertices corresponding to all subjects and objects in a data graph,* L_V *is a collection of vertex labels,* E *is a collection of directed edges that connect the corresponding subjects and objects, and* L_E *is a collection of edge labels. Given an edge* $e \in E$*, the edge's label corresponds to its property.*

Figure 5a shows an example of a graph database with ten triples describing the "Righteous Kill" film. A query of this graph is shown in Fig. 5b.



(a) RDF data graph (b) Graph query

Fig. 5. Example of data graph and query

Definition 4. *(Graph query) A query graph is denoted as* $Q = \langle V, L_V, E, L_E \rangle$*, where* $V = V_p \cup V_c$ *is a collection of vertices that correspond to all subjects and objects in a Graph query, where* V_p *is a collection of parameter vertices, and* V_c *is as defined in the previous definition. As a naming convention, we distinguish variables from elements in* V_c *through a leading question mark symbol (e.g., ?name, ?x),* L_V *is a collection of vertex labels. For a vertex* $v \in V_p$*, its vertex label is* ϕ[5]*. A vertex* $v \in V_c$*,* E *and* L_E *are discussed in Definition 3.*

[5] ϕ is used to denote an empty element.

Forward data stars extend the tuple definition from relational databases. In the relational model, we cannot represent more than one value for an attribute. We use an attribute as head if and only if the attribute is the primary key, otherwise we use the record id.

Proposition 1. *Given a set of predicates linked to a node and a set of predicates in a star SQ, a lattice node S satisfies ($\models$) the star SQ iff predicates(SQ) $\subseteq$predicates(S).*

From our query example we extract the stars shown in Table 1. The next definition explains how to evaluate a query by evaluating individually star queries.

Table 1. Query stars of Fig. 5b

$SQ_f(?movie)$	{(?movie,genre,drama),(?movie,starring,?actor1),(?movie,starring,?actor2)}
$SQ_f(?actor1)$	{(?actor1,win,“oscar”),(?actor1,bornIn,?city)}
$SQ_f(?actor2)$	{(?actor2,bornIn,?city)}
$SQ_b(\text{“}oscar\text{”})$	{(?actor1,win,“oscar”)}
$SQ_b(?city)$	{(?actor1,bornIn,?city),(?actor2,bornIn,?city)}
$SQ_b(?actor1)$	{(?movie,starring,?actor1)}
$SQ_b(?actor2)$	{(?movie,starring,?actor2)}
$SQ_b(drama)$	{(?movie,genre,drama)}

Definition 5. *(Mapping and Mapping universe) [15] A mapping is a partial function μ: $V_p \rightarrow V_c$ from a subset of variables V_p to constant nodes V_c. The domain of a mapping μ, written as dom(μ), is defined as the subset of V_p for which μ is defined. By M we denote the universe of all mappings.*

We now define the main notion of compatibility between mappings. In a straightforward way, two mappings are compatible if they do not contain contradicting variable bindings, *i.e.* if shared variables always map to the same value in both mappings:

Definition 6. *(Compatibility of Mappings) Given two mappings $\mu 1$, $\mu 2$, we say $\mu 1$ is compatible with $\mu 2$ iff $\mu 1(?x) = \mu 2(?x)$ for all $?x \in dom(\mu 1) \cap dom(\mu 2)$. We write $\mu 1 \sim \mu 2$ if $\mu 1$ and $\mu 2$ are compatible, and $\mu 1 \nsim \mu 2$ otherwise.*

We denote as $vars$ the function that returns variables of the element passed as a parameter. For instance, we denote by $vars(t)$ all variables in the triple pattern t, while $vars(SQ_f(x))$ denotes all variables of the query star $SQ_f(x)$.

We write μ(t) to denote the triple pattern obtained when replacing all variables ?x $\in vars(t)$ in t by μ(?x). In the following definition, we use $[\![x]\!]_G$ to denote the process to find relevant mappings of x with respect to G.

Definition 7. *(Triple Evaluation) We call* $[\![t]\!]_G$ *the process allowing to find the mapping related to a triple pattern* t *with respect to a graph* G. $[\![t]\!]_G$ *is formally defined as follows:* $[\![t]\!]_G := \{\mu | dom(\mu) = vars(t) \text{ and } \mu(t) \in G\}$.

From a simple viewpoint, the evaluation process consists in finding mappings such that when we replace the variable nodes by the corresponding constants, the triple obtained is in the data graph.

In the following, we use the triple evaluation definition to characterize the query star evaluation. The evaluation of query star allows to find mappings for variables in the query star with respect to a data graph. For each triple t in the star, we try to find the set of mappings satisfying t. Then, we build the mappings of a query star by joining mappings extracted from triples of the query star. Formally:

Definition 8. *(Query Star Evaluation) The evaluation of a query star* $SQ(x)$ *with respect to the data graph* G *is defined as:*

$$[\![SQ(x)]\!]_G := \{[\![t_1]\!]_G \bowtie [\![t_2]\!]_G \bowtie ... \bowtie [\![t_n]\!]_G | n = card(SQ(x))\}$$

Where:

$$[\![t_i]\!]_G \bowtie [\![t_j]\!]_G = \{\mu_l \cup \mu_r | \mu_l \in [\![t_i]\!]_G \textit{ and } \mu_r \in [\![t_j]\!]_G, \mu_l \sim \mu_r \textit{ and } \mu_l(t_i) \neq \mu_r(t_j)\}$$

From this definition, we designate the evaluation of the JOIN operator between two star queries. Indeed, JOIN will be used as basic operator of the query evaluation. The JOIN of two star queries allows to assembly the mappings obtained by evaluating two star queries. Formally:

Definition 9. *(Stars Join) The evaluation of a join between two star queries is defined as following:*

$$[\![SQ_i]\!]_G \bowtie [\![SQ_j]\!]_G = \{\mu_l \cup \mu_r | \mu_l \in [\![SQ_i]\!]_G, \mu_r \in [\![SQ_j]\!]_G \textit{ and } \mu_l \sim \mu_r\}$$

We take a mapping obtained from the left query star and another from the right query star, we check if the two mappings are compatible, the union of those mappings is a valid mapping for the JOIN of the two star queries.

Before presenting query evaluation using star queries, we will define the cover concept. Indeed, we use this concept to guarantee that a given set of star queries allows the correct evaluation of the query.

Definition 10. *(Star cover) We denote by* $Cover_q(SQ)$ *the set of query triples shared with the query star.*

$$Cover_q(SQ) = \{t | t \in Triples(q) \cup Triples(SQ)\}$$

From previous definitions, we can define query evaluation using a set of star queries as follows:

Proposition 2. *Given a set of stars* $\{SQ_1, SQ_2, ..., SQ_n\}$ *that cover the query, i.e.,* $Cover_q(SQ_1) \cup Cover_q(SQ_2) \cup ... \cup Cover_q(SQ_n) = Triples(q)$*, the evaluation of* q *using the set of star queries is defined as follows:*

$$[\![q]\!]_G = [\![SQ_1]\!]_G \bowtie [\![SQ_2]\!]_G \bowtie ... \bowtie [\![SQ_n]\!]_G$$

With respect to segments, we can define the query evaluation as follows:

$$[\![q]\!]_G = \bigcup_{sg \models SQ_1} [\![SQ_1]\!]_{sg} \bowtie \bigcup_{sg \models SQ_2} [\![SQ_2]\!]_{sg} \bowtie ... \bowtie \bigcup_{sg \models SQ_n} [\![SQ_n]\!]_{sg}$$

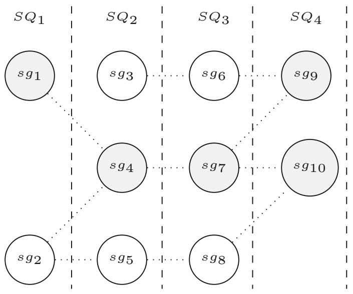


Fig. 6. Execution plan

Let us consider that one result of the evaluation of a triple, a query star and a query is a set of mappings. For the rest of this manuscript we denote by ω this set of mappings. A set of mappings, that represent all results obtained by evaluating a triple, a query star or a query is denoted by Ω (*i.e.*, $\Omega = \{\omega\}$).

Given a query and the stars obtained from backward forward edges, we can easily show that the set of star queries allowing to evaluate the query is not unique. We use the word "Plan" to refer to a set of star queries allowing to evaluate a given query. Formally, a plan is defined as follows:

Proposition 3. *A plan is a order function on a Set of Query Stars allowing to evaluate queries. We denote by* $p = [SQ_1, SQ_2, ..., SQ_n]$ *the plan formed by executing* SQ_1*, then* SQ_2*,..., and finally* SQ_n*.*

The choice of the star queries and the order of evaluation allow to optimize the query evaluation process.

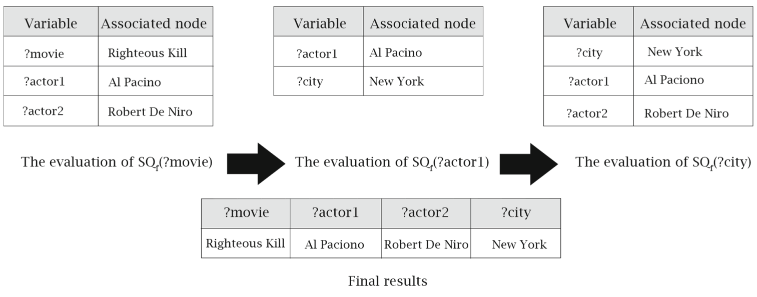
Let us illustrate all of these concepts using the data graph in Fig. 5a and the query in Fig. 5b. As explained previously, we process the query by extracting forward and backward stars. Those stars are shown in Table 1. Let us consider only one way (*i.e.*, one plan) to evaluate the query. We consider the following plan:

$$[SQ_f(?movie), SQ_f(?actor1), SQ_b(?city)]$$

As shown in Fig. 6, each plan is linked to a set of segments. We propose to prune in real time. Indeed, an intermediate segment is evaluated only if this segment is referenced by a segment that has been evaluated by another query star. By Joining the mappings obtained from the different star queries of the plan, we obtain the final result shown in Fig. 7.

4 Experimental Evaluation

We evaluated the performance of our approach using two well known RDF benchmarks: (Watdiv and LUBM). We assessed firstly the ability of the systems to load data greater than the available main memory. Then, we evaluate the performance of query processing.



Variable	Associated node
?movie	Righteous Kill
?actor1	Al Pacino
?actor2	Robert De Niro

Variable	Associated node
?actor1	Al Pacino
?city	New York

Variable	Associated node
?city	New York
?actor1	Al Paciono
?actor2	Robert De Niro

?movie	?actor1	?actor2	?city
Righteous Kill	Al Paciono	Robert De Niro	New York

The evaluation of SQ_f(?movie)
The evaluation of SQ_f(?actor1)
The evaluation of SQ_f(?city)
Final results

Fig. 7. Execution plan schema

4.1 Experimental Setup

Hardware: We conducted all experiments on a machine with an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30 GHz CPU, 1TB hard disk and 32 GB of RAM. Ubuntu server 16.04 LTS is used as an operating system.

Software: The main components of our system (i.e. fragmentation, allocation and indexing modules) are implemented in Java. The extraction of the data stars is coded with C++.

Compared Systems: Our system was compared with three state-of-the-art approaches that apply different execution paradigms. The compared systems are: gStore[6] which is an execution system based on graphs matching, Virtuoso a relational-based system[7], and RDF-3X [14] an intensive-index system.

[6] https://github.com/pkumod/gStore.
[7] https://github.com/openlink/virtuoso-opensource.

Datasets: We evaluate and compare the performance of the systems using the Watdiv [3] and LUBM [10] benchmarks. We compare the execution time to solve queries with different configurations (Linear, Star, Snowflake and Complex). The list of queries is not here for space reasons but it is found in our technical report[8]. Also, we compare the ability of the systems to deal with datasets of different sizes. We generated datasets with 100 and 1 billion triples for Watdiv and 20 and 500 million triples for LUBM. The size of each dataset is detailed in Table 2.

4.2 Pre-processing Evaluation

We tested first the ability of the systems to pre-process and load raw RDF datasets (in the N-Triples format). QDAG, Virtuoso and RDF-3X were able to load all of the tested datasets. On the contrary, gStore was able to load the 100 million triples Watdiv and 20 million LUBM. This is mainly due to the fact that gStore performs the pre-processing in main memory and is unable to load RDF graphs that do not fit into it. Virtuoso loads the dataset to a relational database, RDF-3X into the indexes (e.g. PSO, SPO) and QDAG creates files of segments (SPO and OPS). The number of SPO and OPS segments on each dataset is shown in Table 2. The number of segments SPO and OPS does not grow exponentially and their number remains reasonable to the size of the data.

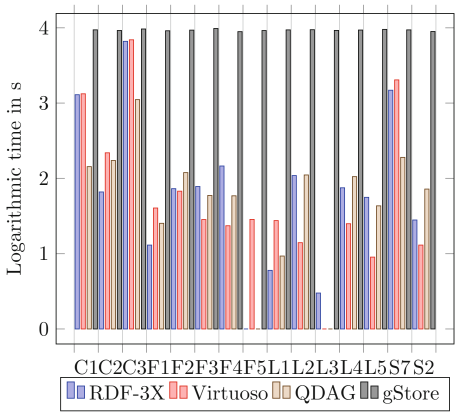
Table 2. Experimental datasets

Dataset	Size (GB)	# Segments SPO	# Segments OPS
Watdiv 100M	14.5	39,855	1,088
Watdiv 1B	149	96,344	4,724
LUBM 20M	3.22	11	13
LUBM 500M	83	18	17

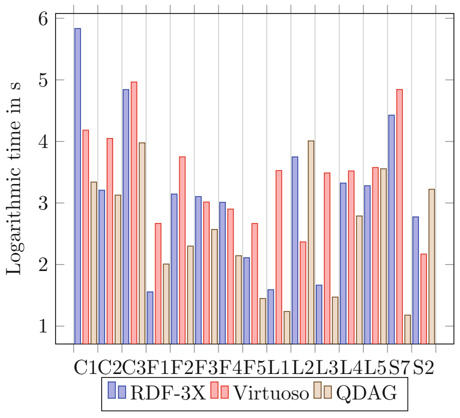
4.3 Query Performance

Watdiv. The query performance for linear (L), star (S), snowflake (F) and complex (C) queries is shown in Fig. 8. We plot using a logarithmic scale since the performance of QDAG is on average 300x better than gStore, leaving the original execution times would have led to unreadable graphs. The results for 100 million triples of Watdiv are shown in Fig. 8a, even if QDAG obtains very similar performance than Virtuoso and RDF-3X for star and linear queries, our model ensures scalability for more complex datasets and queries. This is shown with the complex and snowflakes queries in which our system is on average 1.6X times faster. The behaviour of the same queries in a 100 million dataset is very similar, QDAG is able to solve much more complex queries that are hardly transformed into SQL with a reasonable performance.

[8] https://www.lias-lab.fr/publications/32595/khelil_rdf_processing_report.pdf.

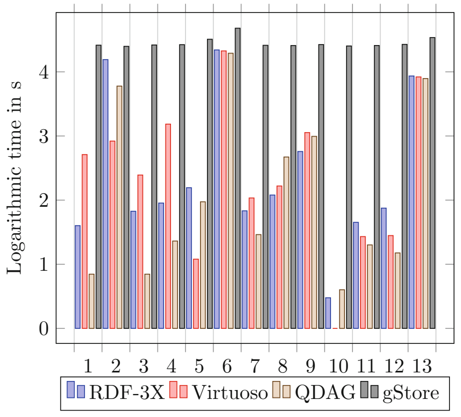


(a) 100 million

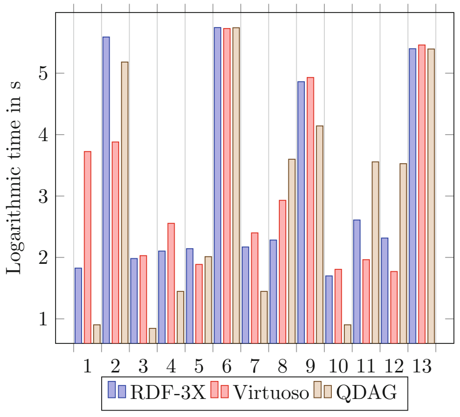


(b) 1 billion

Fig. 8. Query performance for Watdiv



(a) 20 million



(b) 500 million

Fig. 9. Query performance for LUBM benchmark

LUBM. Similarly to what was done for Watdiv, we evaluated the ability of the systems to load different sizes of datasets. gStore was unable to load the dataset of 500 million triples since it is greater than the available main memory. The query performance for the datasets of 20, 500 million triples are shown in Fig. 9. In all cases, our system outperforms gStore (up to 10x faster). The performance in the dataset of 20 million triples is quite similar to the other state-of-the-art systems. However, in the 500M dataset our system showed a better performance for most of the queries, especially complex queries joining many basic graph patterns.

5 Conclusion

In this paper, we propose a new technique for evaluating queries on RDF data. Its main particularity is that it allows combining graph exploration and fragmentation – crucial issues in RDF data repositories. Our results are encouraging and showed that our proposal outperforms gStore system considered as the state of the art in the processing of RDF data.

This work opens several research directions. Currently, we are conducting intensive experiments to evaluate the scalability of our approach. Another direction consists in studying the ordering of the star queries study, by proposing adequate cost models. Finally, we plan to parallelize our approach. Since it already includes the fragmentation process, a new module that has to be developed concerns the management of the transfer of intermediate results between fragments.

References

1. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 411–422. VLDB Endowment (2007)
2. Aït-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient implementation of lattice operations. ACM Trans. Program. Lang. Syst. (TOPLAS) **11**(1), 115–146 (1989)
3. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: The Semantic Web - ISWC 2014–13th International Semantic Web Conference, Riva del Garda, Italy, 19–23 October, pp. 197–212 (2014)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of ACM SIGMOD, pp. 1247–1250. ACM (2008)
5. Briggs, M.: DB2 NoSQL graph store what, why & overview (2012)
6. Cyganiak, R.: A relational algebra for SPARQL. Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170, p. 35 (2005)
7. Deppisch, U.: S-tree: a dynamic balanced signature index for office retrieval. In: Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 77–87. ACM (1986)
8. Erling, O.: Virtuoso, a hybrid RDBMS/graph column store. IEEE Data Eng. Bull. **35**(1), 3–8 (2012)
9. Graefe, G.: Volcano - an extensible and parallel query evaluation system. IEEE Trans. Knowl. Data Eng. **6**(1), 120–135 (1994)
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. J. Web Semant. **3**(2–3), 158–182 (2005)
11. Lehmann, J., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
12. McBride, B.: Jena: a semantic web toolkit. IEEE Internet Comput. **6**, 55–59 (2002)
13. Neumann, T., Moerkotte, G.: Characteristic sets: accurate cardinality estimation for RDF queries with multiple joins. In: Data Engineering (ICDE), pp. 984–994 (2011)
14. Neumann, T., Weikum, G.: RDF-3x: a risc-style engine for RDF. Proc. VLDB Endowment **1**(1), 647–659 (2008)

15. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_3
16. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. Proc. VLDB Endowment **1**(1), 1008–1019 (2008)
17. Zou, L., Mo, J., Chen, L., Özsu, M.T., Zhao, D.: gStore: answering SPARQL queries via subgraph matching. Proc. VLDB Endowment **4**(8), 482–493 (2011)

Extracting High-Level System Specifications from Source Code via Abstract State Machines

Flavio Ferrarotti(✉), Josef Pichler, Michael Moser, and Georg Buchgeher

Software Competence Center Hagenberg, Hagenberg, Austria
{flavio.ferrarotti,josef.pichler,michael.moser,georg.buchgeher}@scch.at

Abstract. We are interested in specifications which provide a consistent high-level view of systems. They should abstract irrelevant details and provide a precise and complete description of the behaviour of the system. This view of software specification can naturally be expressed by means of Gurevich's Abstract State Machines (ASMs). There are many known benefits of such an approach to system specifications for software engineering and testing. In practice however, such specifications are rarely generated and/or maintained during software development. Addressing this problem, we present an exploratory study on (semi) automated extraction of high-level software specifications by means of ASMs. We describe, in the form of examples, an abstraction process which starts by extracting an initial ground-level ASM specification from Java source code (with the same core functionality), and ends in a high-level ASM specification at the desired level of abstraction. We argue that this process can be done in a (semi) automated way, resulting in a valuable tool to improve the current software engineering practices.

1 Introduction

We consider good software specifications to be much more than just prototypes to build systems. In our view, they should also enable us to explore, reuse, debug, document and test systems, and to explain their construction in a verifiable way. In particular if these specification (models) are meant to help practitioners to manage complex software-intensive systems.

There are many formal and semi-formal software specification methods which realize this view, have been around for many year, and have successfully been applied in practice. See [21] for an overview with practical focus of the main methods, including ASM, UML, Z, TLA+, B and Estelle among others.

The research reported in this paper results from the project *Higher-Order Logics and Structures* supported by the Austrian Science Fund (FWF: [**I2420-N31**]). It was further supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

Software specifications should serve both, the client and the developer. The client should be able to understand the specification so that she/he can validate it. The developer should have as much freedom as possible to find the best implementation that conforms to the specification. Thus, it is fundamentally important for a good specification to present a consistent high-level description of the system abstracting away irrelevant details. Let us illustrate this point with an example.

Example 1. Suppose we are given the task of specifying an algorithm for sorting a sequence of elements in-place. The algorithm must proceed sequentially, making exactly one swap in each step until the sequence is in order. The abstract state machine (ASM) in Listing 1.1 provides a formal, yet high-level specification of such algorithm. Provided we are aware that lines 3 and 4 are executed in parallel and thus there is no need to save the value of $array(i)$ into a temporary variable, this formal specification should be self-explanatory. It is not the most efficient algorithm for the task at hand, but it is the most general and gives freedom to the developer to refine it into an implementation of her/his choice such as bubble sort, insertion sort or Shell sort, among others.

```
1 rule $Sort$ =
2 choose $i, j \in indices(array)$ with $i < j$ and $array(i) > array(j)$
3           $array(i) := array(j)$
4           $array(j) := array(i)$
```

Listing 1.1. ASM High-Level Specification of Sort Algorithm.

Despite the many benefits that good high-level formal software specifications bring to software design, verification by reasoning techniques, validation by simulation and testing, and documentation, they are still a rare occurrence in the software industry, except in the case of mission critical systems. The commonly cited problems for their adoption in practice are the restrictive time and money constraints under which software is developed, and the dynamic nature of software evolution which makes difficult to keep design and documentation up to date. The need for good software specifications is further underlined by the fact that most programmers need to work on software which was not designed or developed by them, and the growing demand to document and reimplement legacy software systems.

In order to alleviate these problems, since early on [12] a considerable amount of effort have been put into reverse engineering higher level abstractions from existing systems. Nowadays the common approach in the literature (see for instance [22]) is to transform a given program P which conforms to a given grammar G into a high-level model M which conforms to a meta model MM. During this transformation P is usually represented by an abstract syntax tree or a concrete syntax tree. The extraction process relies in the specification of mappings between elements of G and MM. Independently for the specific details, this transformation is done in one big step from P to M, and the level of abstraction of M is fixed, determined by the mappings from G to MM.

In this paper we propose a different approach. Instead of relying on a big step transformation from the source code to a model at the desired level of abstraction, we propose to derive formal software specifications by a sequence of (semi) automated transformations, in which each transformation increases the level of abstraction of the previous specification in the sequence. We argue that this process can be done in a (semi) automated way and thus result in a valuable tool to improve the current software (reverse) engineering practices.

The method for high-level system design and analysis known as the ASM method [9] inspired our idea of extracting high-level specifications from software following an organic and effectively maintainable sequence of rigorous and coherent specifications at stepwise higher abstraction levels. The simple but key observation to this regard is that the process of stepwise refinement from high-level specification down to implementation provided by the ASM method, can be applied in reverse order and thus used for the (semi) automated extraction of high-level software specifications from source code.

The idea of using formal methods to reverse engineering formal specifications is of course not new. Already in the nineties, Z notation was used to extract formal specifications from COBOL [11,23]. Declarative specifications such as Z imply a fixed level of abstraction for design and verification, completely independent of any idea of computation. This is unsuitable for the multi-step abstraction approach proposed in this paper. In this sense, ASMs provide us with the required unifying view of models of computation and declarative frameworks [5].

The paper is organized as follows. In Sect. 2 we argue why ASMs provide the correct foundations for the method presented in this work for high-level software specification extraction. The actual method for stepwise abstraction of software specifications is presented in Sect. 3. In Sect. 4 we show how the method works in practice through a complete example. We conclude our work in Sect. 5.

2 Abstract State Machines

A distinctive feature of the ASM method which is not shared by other formal and semi-formal specification methods such as B, Event-B and UML is that, by the ASM thesis (first stated in [18,19] as project idea for a generalization of Turing's thesis), ASMs can step-by-step faithfully model algorithms at *any level of abstraction*. This thesis has been theoretically confirmed for most well known classes of algorithms, including sequential [20], parallel [2,3,14], concurrent [8], reflective [13], and even quantum [16] algorithms. Moreover, it has long been confirmed in practice (see [4] and Chapter 9 in [9] for a survey). This distinctive feature is a key component of the approach that we propose in this paper to extract specifications from source code. Moreover, ASMs provide simple foundations and a uniform conceptual framework (see Section 7.1 in [9]).

This paper can be understood correctly by reading our ASM rules as pseudocode over abstract data types. Nevertheless, we review some of the basic ASM features in order to make the paper self-contained. The standard reference book for this area is [9].

The *states* of ASMs are formed by a domain of elements or objects and a set of functions defined over this domain. That is, states are arbitrary universal structures. Predicates are just treated as characteristic functions. The collection of the types of the functions (and predicates) which can occur in a given ASM is called its *signature*.

In its simplest form an ASM of some signature Σ can be defined as a finite set of transition rules of the form **if** *Condition* **then** *Updates* which transforms states. The condition or guard under which a rule is applied is an arbitrary first-order logic sentence of signature Σ. *Updates* is a finite set of assignments of the form $f(t_1, \ldots, t_n) := t_0$ which are executed in parallel. The execution of $f(t_1, \ldots, t_n) := t_0$ in a given state S proceeds as follows: first all parameters $t_0, t_1, \ldots, t_n$ are assigned their values, say $a_0, a_1, \ldots, a_n$, then the value of $f(a_1, \ldots, a_n)$ is updated to a_0, which represents the value of $f(a_1, \ldots, a_n)$ in the next state. Such pairs of a function name f, which is fixed by the signature, and optional argument $(a_1, \ldots, a_n)$ of dynamic parameters values a_i, are called locations. They represent the abstract ASM concept of memory units which abstracts from particular memory addressing. Location value pairs (l, a), where l is a location and a is a value, are called updates and represent the basic units of state change.

The notion of ASM run (or equivalently computation) is an instance of the classical notion of the computation of transition systems. An ASM computation step in a given state consists in executing simultaneously all updates of all transition rules whose guard is true in the state. If these updates are consistent, the result of their execution yields a next state, otherwise it does not. A set of updates is consistent if it contains no pairs $(l, a), (l, b)$ of updates to a same location l with $a \neq b$. Simultaneous execution, as obtained in one step through the execution of a set of updates, provides a useful instrument for high-level design to locally describe a global state change. This synchronous parallelism is further enhanced by the transition rule **forall** x **with** φ **do** r which expresses the simultaneous execution of a rule r for each x satisfying a given condition φ. Similarly, non-determinism as a convenient way of abstracting from details of scheduling of rule executions can be expressed by the rule **choose** x **with** φ **do** r, which means that r should be executed with an arbitrary x chosen among those satisfying the property φ.

3 The Stepwise Abstraction Method

In this section we present a stepwise abstraction method to extract high-level specifications from source code. The method comprises the following two phases:

1. *Ground specification extraction:* This is the first step consisting on parsing the source code of the system in order to translate it into a behaviourally equivalent ASM. Here we use the term *behaviourally equivalent* in the precise sense of the ASM thesis (see [2,3,8,13,14,20] among others), i.e., in the sense that behaviourally equivalent algorithms have step-by-step exactly the

same runs. Thus the ground specification is expected to have the same core functionality as the implemented system.

2. *Iterative high-level specification extraction:* After the first phase is completed, the ground ASM specification is used as a base to extract higher-level specifications, by means of a semi-automated iterative process. The implementation of the method must at this point present the user with different options to abstract away ASM rules and/or data.

A detailed analysis of these phases follows.

3.1 Ground Specification Extraction

In this phase we focus on how to extract an ASM behaviourally equivalent to a source code implementation in a given programming language. In our research center, we have a positive industrial experience in parsing and extracting knowledge from source code[1]. We have built and applied several reverse engineering tools around the Abstract Syntax Tree Metamodel (ASTM) standard[2]. In particular, we have shown its potential for multi-language reverse engineering in practice [15].

Using this approach and adapting our previous results, we have determined that it is possible extract the desired ground specifications in the form of a behaviourally equivalent ASM by automated means. The idea is to transform the source code into an ASM model in two steps. First, by means of eKnows[3], the source code is parsed into a language-agnostic canonical AST representation. Besides concrete language syntax, this intermediate representation also abstracts from language-specific semantics with regard to control-flow. For instance, the switch statement occurs in different forms, namely with or without fall-through semantics. eKnows constructs an AST representation with a standardized semantic (e.g. explicit break statements even for non-fall-through languages) that allows homogeneous subsequent analysis/transformation steps. Furthermore, eKnows resolves unstructured control-flow (e.g. break and continue within loop statements, or goto statement) by means of refactoring resulting in well-structured control-flow, i.e. single entry/exit points of statements.

In the second step, we provide rewriting rules for AST nodes specifically related to control-flow (e.g. for loops, conditional statements) and assignment statements. Rewriting rules for control-flow nodes can be applied in a straightforward way to individual nodes independent of any context information. The subsequent examples illustrate the idea for the transformation of loop statements. The transformation of assignment statements, however, need semantic analysis of the source code due to the difference between strict sequential execution order of program code and simultaneous execution of ASM update rules. We can leverage symbolic execution (also part of eKnows) in order to eliminate intermediate variables and construct assignment statements that only contain

[1] http://codeanalytics.scch.at/.
[2] https://www.omg.org/spec/ASTM/1.0/.
[3] https://www.scch.at/de/eknows.

input/output parameters of the analyzed algorithms. In this transformed representation, the strict execution sequence becomes irrelevant and statements can be transformed into behaviorally equivalent ASM update rules executed in parallel.

Next we present a simple example of ground level ASM specifications extracted from a Java implementation of the bubble sort algorithm.

Example 2. Let us analyse the very simple and compact bubble sort algorithm implemented by the Java method in Listing 1.2.

```
1 public static void bubbleSort(int array[]) {
2     for (int n = array.length - 1; n > 0; n--) {
3         for (int i = 0; i < n; i++) {
4             if (array[i] > array[i+1]) {
5                 int temp = array[i];
6                 array[i] = array[i+1];
7                 array[i+1] = temp;} } } }
```

Listing 1.2. Bubble sort algorithm as Java method.

Let `for` be the following iterative turbo ASM rule which first executes the rule R_0, and then repeats the execution of its body rule R_2 followed by R_1 as long as they produce a non-empty update set and the condition *cond* holds.

```
for ($R_0$; $cond$; $R_1$) $R_2$ =
                    $R_0$ seq iterate (if $cond$ then $R_2$ seq
    $R_1$)
```

The turbo ASM in Listing 1.3 can easily be obtained from the Java code in Listing 1.2 by mostly simple syntactic rewriting, except for the value swap done in parallel in lines 5 and 6 which requires a simple semantic abstraction of lines 5–7 in Listing 1.2. Using the symbolic approach described above, the Java variable `temp` in the assignment in line 7 of Listing 1.2 would get substituted by the previous assignment (line 5) resulting in the ASM rule $array(i+1) := array(i)$.

```
1 rule $bubbleSort0$ =
2     for ($n := array.length - 1$; $n > 0$; $n := n - 1$)
3         for ($i := 0$; $i < n$; $i := i + 1$)
4             if $array(i) > array(i+1)$ then
5                 $array(i) := array(i+1)$
6                 $array(i+1) := array(i)$
```

Listing 1.3. Turbo ASM extracted from Java method `bubbleSort`.

Alternatively, we can extract from Listing 1.2 the control state ASM in Listing 1.4. Same as in the case of the turbo ASM, the transformation from the Java method `bubbleSort` to the control state ASM only requires simple rewriting techniques.

```
rule $bubbleSort1$ =
    if $state = s_0$ then
        $n := array.length - 1$
        $state := s_1$
    if $state = s_1$ then
        if $n > 0$ then
            $i := 0$
            $state := s_2$
    if $state = s_2$ then
        if $i < n$ then
            if $array(i) > array(i+1)$ then
                $array(i) := array(i+1)$
                $array(i+1) := array(i)$
            $i := i + 1$
        else
            $n := n - 1$
            $state := s_1$
```

Listing 1.4. Control State ASM abstracted from Java code of `bubbleSort`.

Although the control state ASM in Listing 1.4 has more lines of code than the turbo ASM in Listing 1.3 (and that the original Java code), it has certain advantages. It can for instance be represented graphically as the UML-style diagram in Fig. 1. Furthermore, the control state ASM presents a transparent white-box view of the states while the turbo ASM presents a black-box view which hides internal sub-computations. Which of these views is more useful depends on the desired specification level and the application at hand, and can be decided by the user. For instance, at low levels of abstraction, control ASMs can lead to complex UML-style diagrams which might share the usual drawbacks of UML activity diagrams, unnecessarily replacing elegant structured code by control flow based in states (a kind of "hidden goto" if viewed as a program).

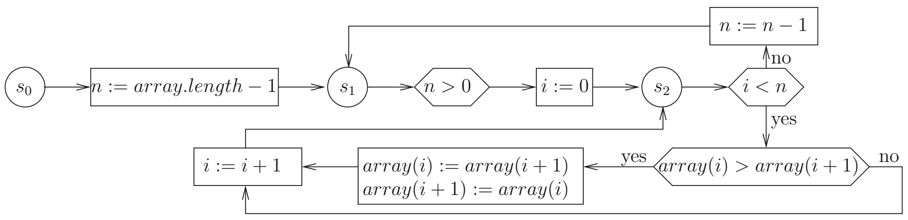


Fig. 1. bubbleSort1

3.2 Iterative High-Level Specification Extraction

Without entering into technical details, which are nevertheless well explained in Section 3.2 in [9], we note that the schema of ASM *refinement step*, can also be viewed as describing an *abstraction step* if it is used for extracting a high-level model. In this sense, the re-engineering project in [1] confirms this idea in practice and it is a source of inspiration for our proposal.

Thus, when abstracting an ASM M' from an $ASM\ M$, there is a lot of freedom. In particular, it is possible to perform diverse kinds of abstractions by combining the following items:

– The notion of an abstracted state, obtained by changing (usually reducing) the signature.
– The notion of state of interest and correspondence between states, obtained by changing (usually reducing) the states of interest in M' with respect to M, and determining the pairs of states of interest in the runs of M and M' that we want to relate through the abstraction.
– The notion of abstract computation segments, obtained by changing (usually reducing) the number of abstract steps that lead to states of interest of M' with respect to M.
– The notion of locations of interest and of corresponding locations, obtained by changing (usually reducing) the locations of interest of M', and determining the pairs of corresponding locations of interest of M and M' that we want to relate through abstraction. Recall that in ASM terminology the term location refer to abstract data container.
– The notion of equivalence of the data in the locations of interest, obtained by changing (usually reducing) the number of different classes of equivalence, and thus also changing the notion of equivalence of corresponding states of interest of M and M'.

The aim of this phase is to semi-automatically extract high-level system specifications starting from the ground ASM specification built in the previous phase. In this paper we present a proof of concept through examples. We show that very simple heuristic analyses of ASM rules can already lead to useful abstractions. Of course, much more sophisticated abstraction mechanisms such as abstract interpretations are certainly possible, opening what would constitute an interesting research project.

Example 3. Note that it is clear from the ASM ground specifications extracted in Example 2 that the order in which the values in the sequence are swapped does not really matter from a conceptual high-level perspective. That is, if we keep swapping the values of $array(i)$ and $array(i+1)$ as long as $array(i) > array(i+1)$ for some index i, then the algorithm still works correctly. We thus can abstract the ASM in Listing 1.5. After at most $2n$ steps, where n is the length of the array, we have that for every index i the condition in the choose rule no longer holds. At that point the machine stops and, for every index i, it holds that $array(i) \leq array(i+1)$, i.e., the array is in order. Clearly, the ASM in Listing 1.5 is not only an abstraction of the bubble sort algorithm, it is also an specialization of the in place sorting algorithm specified by the ASM rule in Listing 1.1.

```
rule bubbleSort2 =
    choose i with $0 \leq i < array.length - 1$ and $array(i) > array(i+1)$
        $array(i) := array(i+1)$
        $array(i+1) := array(i)$
```

Listing 1.5. Abstraction from control state ASM *bubbleSort*1.

Admittedly, high-level specifications such as the one in Listing 1.5 are not trivial to abstract following mechanical procedures, since they are not decidable in the general case. We can however analyse and classify programming patterns, applying engineering and AI techniques such as heuristics, symbolic execution, machine learning, theorem provers etc. to identify appropriate and correct abstractions of this kind.

4 Dijkstra's Shortest Path Algorithm: Extracting High-Level Specifications from a Java Implementation

In this section we showcase a step-by-step *formal* process of abstraction from a Java implementation of the famous Dijkstra's shortest path algorithm, up to a high-level ASM specification of the underpinning graph traversal algorithm. The correctness of each step of this abstraction process can be formally proven following similar arguments to those in the refinement proofs of Sect. 3.2 of the ASM book [9]. Automated proving would also be possible in some cases, but that is not the focus of this work.

We start from a Java implementation (slightly adapted from the one in https://www.baeldung.com/java-dijkstra) of the shortest path algorithm. Rather surprisingly we show that applying our method we can extract very similar high-level specifications to those in Sect. 3.2 of the ASM book [9].

In the Java implementation of the algorithm graph are represented as sets of nodes. Each node is an object of the class in Listing 1.6 which has a name, an upper bound for its distance from source, and a list of adjacent nodes.

```
public class Node {
    private String name;
    private Integer upbd = Integer.MAX_VALUE;
    private Map<Node, Integer> adj = new HashMap<>();
    public Node(String name) {
        this.name = name;
    }
    public void addDestination(Node destination, int
      weight) {
        adjacentNodes.put(destination, weight);
    }
    // getters and setters ...
}
```

Listing 1.6. Class Node.

It is not difficult to see that the Java method in Listing 1.7 actually implements Dijkstra's shortest path algorithm. This is the case mainly because: (a) the algorithm is well known, (b) the implementation is quite standard, and (c) the code is quite short. If either of (a), (b) or (c) does not hold, then the task of understanding the program would certainly be more time consuming and challenging.

```
public static Graph shortestPath(Graph graph,Node source)
   {source.setUpbd(0);
    Set<Node> visited = new HashSet<>();
    Set<Node> frontier = new HashSet<>();
    frontier.add(source);
    while (frontier.size() != 0) {
        Node u = LowestDistanceNode(frontier);
        frontier.remove(u);
        for (Entry<Node,Integer> pair :
        u.getAdj().entrySet()) {
            Node v = pair.getKey();
            Integer weight = pair.getValue();
            LowerUpbd(u,v,weight);
            if (!visited.contains(v)) {
                visited.add(v);
                frontier.add(v);}}}
    return graph;}

private static void LowerUpbd(Node u, Node v, Integer
    weight) {
    if (u.getUpbd() + weight < v.getUpbd()) {
        v.setUpbd(u.getUpbd() + weight);}}
```

Listing 1.7. Shortest path algorithm as Java program.

Let us now abstract the code of `shortestPath` by transforming it into a control state ASM. We follow a very simple procedure which consists mostly on syntactic rewriting. First we note that we can represent the omnipresent binding or instantiation of methods and operations to given objects, by means of parametrized functions [7]. The schema can be expressed by the equation $\mathit{self}.f(x) = f(\mathit{self}, x)$ or $f(x) = f(\mathit{self}, x)$. The parameter *self* is used to denote an agent, typically the one currently executing a given machine. This is similar to the object-oriented current class instance *this* with respect to which methods of that class are called (executed). In an object oriented spirit the parameter *self* is often left implicit.

The state in which the control state ASM will operate is easily abstracted from the input parameters to the method `shortestPath`, i.e., the `Graph` and `Node` classes. We omit a detailed description here since it will be clear from the context.

The rewriting of `shortestPath` into a *behavioural equivalent* ASM control state machine *ShortestPath* proceeds as follow:

1. Lines 2–4 in Listing 1.7 translate to simple updates of the current upper bound value of the node `source` to 0 (initially set to `Integer.MAX_VALUE`, or ∞ in ASM notation) and the values of `visited` and `frontier` to empty sets. These three updates can be done in parallel and thus in the initial control state s_0. The update to `frontier` in Line 5 cannot be done in parallel with that of Line 4. Nevertheless, a simple heuristic can easily discover that these two updates can be collapsed into one. Thus Lines 2–5 can be translated to the parallel updates shown in lines 3–5 in Listing 1.8.
2. The while-loop starting in line 6 requires a new control state to which the ASM can return. If we are in this state and the condition in the while-loop is satisfied, then the rule corresponding to the code inside the while-loop is applied and the machine remains in control state s_1. See lines 3–5 in Listing 1.8.
3. Lines 7–8 can be done in parallel since they require updates to different locations. See lines 9–10 in Listing 1.8.
4. Same as the while-loop, the for-loop in line 9 calls for a new control state and to keep track of the nodes adjacent to `LowestDistanceNode(frontier)` which have not been visited yet. Once the for-loop is done, i.e., there is no more adjacent nodes to visit, we need to return to the control state s_1, since this for-loop is nested in the while-loop being represented in that control state. The result is shown in lines 11–16 and 22–23 of Listing 1.8.
5. The values of v and *weight* are only defined and used locally in lines 10–15. In addition, lines 12–15 update disjoint state locations, and there is no interdependency among them. Thus, we can replace lines 10–11 by a let-rule and process lines 12–15 in parallel. See lines 17–21 in Listing 1.8.
6. Finally, the ASM rule *LowerUpbd* is almost identical to the `LowerUpbd` method, except for the trivial differences in notation. In *LowerUpbd*, parameter u is a location variable while parameters v and *weight* are logical variables. See lines 25–27 in Listing 1.8.

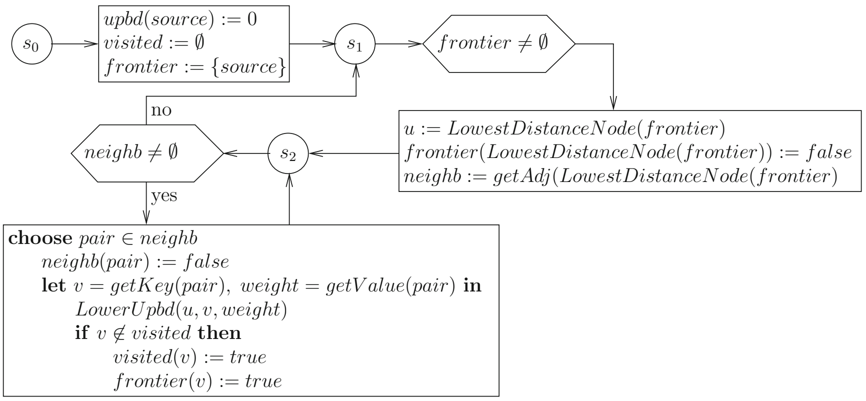
```
1  rule $ShortestPath0$ =
2      if $state = s_0$ then
3          $upbd(source) := 0$
4          $visited := \emptyset$
5          $frontier := \{source\}$
6          $state := s_1$
7      if $state = s_1$ then
8          if $frontier \neq \emptyset$ then
9              $u := LowestDistanceNode(frontier)$
10             $frontier(LowestDistanceNode(frontier)) := false$
11             $neighb := getAdj(LowestDistanceNode(frontier))$
12             $state := s_2$
13     if $state = s_2$ then
```

```
        if $neighb \neq \emptyset$ then
            choose $pair \in neighb$
                $neighb(pair) := false$
                let $v = getKey(pair),\ weight = getValue(pair)$ in
                    $LowerUpbd(u, v, weight)$
                    if $v \notin visited$ then
                        $visited(v) := true$
                        $frontier(v) := true$
        else
            $state := s_1$

rule $LowerUpbd(u, v, weight) =$
    if $upbd(u) + weight < upbd(v)$ then
        $upbd(v) := upbd(u) + weight$
```

Listing 1.8. Control State ASM extracted from the Java code of `shortestPath`.

Being *ShortestPath*0 a control state ASM, we can represent it using UML-style graphical notation. This gives us the self explanatory Fig. 2.



s_0
$upbd(source) := 0$
$visited := \emptyset$
$frontier := \{source\}$
s_1
$frontier \neq \emptyset$
no
$u := LowestDistanceNode(frontier)$
$frontier(LowestDistanceNode(frontier)) := false$
$neighb := getAdj(LowestDistanceNode(frontier))$
$neighb \neq \emptyset$
s_2
yes
choose $pair \in neighb$
$neighb(pair) := false$
let $v = getKey(pair),\ weight = getValue(pair)$ **in**
$LowerUpbd(u, v, weight)$
if $v \notin visited$ **then**
$visited(v) := true$
$frontier(v) := true$

Fig. 2. ShortestPath0

Examining the code in lines 15–21 in Listing 1.8 plus the rule *LowerUpbd*, it is not difficult to conclude that instead of extending the frontier by one neighbour of u at a time, we can extend it as a wave, i.e., in one step we can in parallel extend the frontier to all neighbours of u. This is so because the choose rule that select the neighbour to be processes in each round, implies that the order in which this is done does not affect the result. Furthermore, there is no possibility of clashes since the updated locations $visited(v)$, $frontier(v)$ and $upbd(v)$ are all disjoint for different values of v. Thus we can abstract from *ShortestPath*0 the control state ASM *ShortestPath*1 in Listing 1.9 , where we replace the choose rule by a for all rule.

```
rule $ShortestPath1$ =
    if $state = s_0$ then
        $upbd(source) := 0$
        $visited := \emptyset$
        $frontier := \{source\}$
        $state := s_1$
    if $state = s_1$ then
        if $frontier \neq \emptyset$ then
            $u := LowestDistanceNode(frontier)$
            $frontier(LowestDistanceNode(frontier)) := false$
            $neighb := getAdj(LowestDistanceNode(frontier))$
            $state := s_2$
    if $state = s_2$ then
        forall $pair \in neighb$
            let $v = getKey(pair),\ weight = getValue(pair)$ in
                $LowerUpbd(u, v, weight)$
                if $v \notin visited$ then
                    $visited(v) := true$
                    $frontier(v) := true$
        $state := s_1$
```

Listing 1.9. Abstraction of the *ShortestPath*0 ASM rule.

As a next step, we can simply eliminate control state s_2 by using a let rule, changing u and *neighb* from state locations to logical variables. The result is shown in Listing 1.10.

```
rule $ShortestPath1$ =
    if $state = s_0$ then
        $upbd(source) := 0$
        $visited := \emptyset$
        $frontier := \{source\}$
        $state := s_1$
    if $state = s_1$ then
        if $frontier \neq \emptyset$ then
            let $u = LowestDistanceNode(frontier)$ ,

   $neighb = getAdj(LowestDistanceNode(frontier))$
            in $frontier(u) := false$
                    forall $pair \in neighb$
                              let
   $v = getKey(pair),\ weight = getValue(pair)$ in
                                $LowerUpbd(u, v, weight)$
                                if $\notin visited(v)$ then
                                    $visited(v) := true$
                                    $frontier(v) := true$
```

Listing 1.10. Abstraction of the ShortestPath1 ASM rule.

At this point we have quite an abstract view of the shortest path algorithm. We can nevertheless continue this abstraction process. An interesting possibility to this regard is to eliminate the information regarding edge weights. In this way, we get the ASM in Listing 1.11. It is not difficult to see that the resulting ASM no longer calculates the shortest path from the source. It has been transformed into an ASM that specifies the graph traversal algorithm which underpins the shortest path algorithm.

```
rule $GraphrTraversal0$ =
    if $state = s_0$ then
        $upbd(source) := 0$
        $visited := \emptyset$
        $frontier := \{source\}$
        $state := s_1$
    if $state = s_1$ then
        if $frontier \neq \emptyset$ then
            choose $u \in frontier$
                $frontier(u) := false$
                forall $v \in getAdj(u)$
                    if $v \notin visited(v)$ then
                        $visited(v) := true$
                        $frontier(v) := true$
```

Listing 1.11. Abstraction of the *ShortestPath*2 ASM rule.

We can further abstract *GraphrTraversal*0 by processing all the nodes in the frontier in parallel instead of one-by-one. This is the same idea that we use to abstract *ShortestPath*1 from *ShortestPath*0. In this way, we get the ASM in Listing 1.12.

```
1  rule $GraphrTraversal1$ =
2      if $state = s_0$ then
3          $upbd(source) := 0$
4          $visited := \emptyset$
5          $frontier := \{source\}$
6          forall $u \in frontier$
7              $frontier(u) := false$
8              forall $v \in getAdj(u)$
9                  if $v \notin visited(v)$ then
10                     $visited(v) := true$
11                     $frontier(v) := true$
```

Listing 1.12. Abstraction of the *GraphTraversal*0 ASM rule.

A somehow more abstract view can still be obtained if we simply replace lines 8–10 in Listing 1.12 by a function defined by a sub-machine.

5 Conclusion

We have argued that it is possible to derive high-level formal software specifications in the form of ASMs by a sequence of (semi) automated transformations, in which each transformation increases the level of abstraction of the previous specification in the sequence. This provides a new tool to improve the current software (reverse) engineering practices as shown by the encouraging results of the small experimental examples presented in this paper. The proposed approach to software re-engineering have several advantages, including:

– Precise, yet succinct and easily understandable specifications at desired levels of abstraction.
– Each abstraction/refinement step can be proven correct if needed. This enables for instance to prove that the implementation satisfies the requirement.
– Only the first abstraction from source code to ASM rules depends from the programming language of the implementation. Subsequent abstractions only rely on general principles and transformations of ASM rules.
– The initial abstraction from source code to ASM can potentially be done entirely automatically via rewriting rules.
– Enables the exploitation of abstraction for specification reuse.
– Specifications are executable for instance in CoreASM or Asmeta.
– Can be used for reverse engineering/understanding (legacy) source code.
– Can be used to produce finite state machines for model based testing. For instance by means of refinement of the extracted high-level ASM models to finite state machines [17].
– Interactive exploration of the design on all abstraction levels, enabling the discovery of high-level bugs.

The natural next step is to confirm the observations in this paper within the context of large software implementations, in the style of [1], but using our semi-automated approach instead. For that, we aim to extend our eKnows[4] platform to extract ground ASM specification from source code and experiment with software systems of our industrial partners. In parallel, we plan to carry on a systematic study of heuristics for the automated extraction of high-level ASM specifications, starting from detailed ground ASM specifications. In this sense, references [6,10] are a good starting point.

References

1. Barnett, M., Börger, E., Gurevich, Y., Schulte, W., Veanes, M.: Using abstract state machines at microsoft: a case study. In: Gurevich, Y., Kutter, P.W., Odersky, M., Thiele, L. (eds.) ASM 2000. LNCS, vol. 1912, pp. 367–379. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44518-8_21

[4] https://www.scch.at/de/eknows.

2. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms. ACM Trans. Comput. Logic **4**(4), 578–651 (2003)
3. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms: correction and extension. ACM Trans. Comput. Logic **9**(3), 19:1–19:32 (2008)
4. Börger, E.: The origins and the development of the ASM method for high level system design and analysis. J. UCS **8**(1), 2–74 (2002)
5. Börger, E.: Abstract state machines: a unifying view of models of computation and of system design frameworks. Ann. Pure Appl. Logic **133**(1–3), 149–171 (2005)
6. Börger, E.: Design pattern abstractions and abstract state machines. In: Proceedings of the 12th International Workshop on Abstract State Machines, ASM 2005, 8–11 March 2005, Paris, France, pp. 91–100 (2005). http://www.univ-paris12.fr/lacl/dima/asm05/DesignPattern.ps
7. Börger, E., Cisternino, A., Gervasi, V.: Ambient abstract state machines with applications. J. Comput. Syst. Sci. **78**(3), 939–959 (2012)
8. Börger, E., Schewe, K.: Concurrent abstract state machines. Acta Inf. **53**(5), 469–492 (2016)
9. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-642-18216-7
10. Börger, E., Stärk, R.F.: Exploiting abstraction for specification reuse. the Java/C# case study. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2003. LNCS, vol. 3188, pp. 42–76. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30101-1_3
11. Bowen, J.P., Breuer, P.T., Lano, K.: Formal specifications in software maintenance: from code to z^{++} and back again. Inf. Softw. Technol. **35**(11–12), 679–690 (1993)
12. Chikofsky, E.J., II, J.H.C.: Reverse engineering and design recovery: a taxonomy. IEEE Softw. **7**(1), 13–17 (1990)
13. Ferrarotti, F., Schewe, K.-D., Tec, L.: A behavioural theory for reflective sequential algorithms. In: Petrenko, A.K., Voronkov, A. (eds.) PSI 2017. LNCS, vol. 10742, pp. 117–131. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74313-4_10
14. Ferrarotti, F., Schewe, K., Tec, L., Wang, Q.: A new thesis concerning synchronised parallel computing - simplified parallel ASM thesis. Theor. Comput. Sci. **649**, 25–53 (2016)
15. Fleck, G., et al.: Experience report on building ASTM based tools for multi-language reverse engineering. In: IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016, Suita, Osaka, Japan, 14–18 March 2016, vol. 1, pp. 683–687 (2016)
16. Grädel, E., Nowack, A.: Quantum computing and abstract state machines. In: Börger, E., Gargantini, A., Riccobene, E. (eds.) ASM 2003. LNCS, vol. 2589, pp. 309–323. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36498-6_18
17. Grieskamp, W., Gurevich, Y., Schulte, W., Veanes, M.: Generating finite state machines from abstract state machines. In: Proceedings of the International Symposium on Software Testing and Analysis, ISSTA 2002, Roma, Italy, 22–24 July 2002, pp. 112–122. ACM (2002)
18. Gurevich, Y.: Reconsidering turing's thesis: toward more realistic semantics of programs. Technical Report CRL-TR-36-84, January 1984
19. Gurevich, Y.: A new thesis. Technical Report 85T–68-203, abstracts, American Mathematical Society (1985)
20. Gurevich, Y.: Sequential abstract-state machines capture sequential algorithms. ACM Trans. Comput. Logic **1**(1), 77–111 (2000)

21. Habrias, H., Frappier, M.: Software Specification Methods. ISTE (2006)
22. Izquierdo, J.L.C., Molina, J.G.: Extracting models from source code in software modernization. Softw. Syst. Model. **13**(2), 713–734 (2014)
23. Lano, K., Breuer, P.T., Haughton, H.P.: Reverse-engineering COBOL via formal methods. J. Softw. Maintenance **5**(1), 13–35 (1993)

Data Warehousing

Thinking the Incorporation of LOD in Semantic Cubes as a Strategic Decision

Selma Khouri(✉), Abdessamed Réda Ghomari, and Yasmine Aouimer

Ecole Nationale Supérieure d'Informatique, BP 68M, 16309 Oued-Smar, Algeria
{s_khouri,a_ghomari,ey_aouimeur}@esi.dz

Abstract. With the advent of Linked Open Data (LOD) initiatives, organizations have seen the opportunity of augmenting their internal data cube systems with these external data. While IT actors manage technical issues of internal and external sources, a new actor has emerged "the Chief Data Officer" (CDO), the role of which is to align and prioritize data activities with key organizational priorities and goals. Existing literature managing the incorporation of LOD in internal Data cubes mainly focus on technical aspects of the LOD source and ignores the CDO role in this strategy. In this paper, we claim that technical actions should be conducted by the managerial level, which is reflected through the goals of the organization data cube and their related Key Performance Indicators (KPIs). For doing this, we first propose a metamodel aligning the three models: the data-flow model, the goal model and the KPI model. Then, we propose a process for specifying KPIs into Sparql language, the standard language for querying LOD sources. Experiments are conducted to measure the impact of the decision of integrating external LOD sources at KPI/goal level and on the technical data level. A case tool dedicated to the CDO is implemented to conduct the proposed approach.

Keywords: Cube & DW · Design approach · CDO · LOD · Goal · KPI · Managerial level · Technical level

1 Introduction

Because companies have to survive in a competitive world, they continuously look for new information and insights collected from their internal and external sources. The Data Warehouse (DW) of a company integrates and unifies data from internal sources. Different studies demonstrated that valuable information for the company DW can be found in external sources. The Linked Open Data (LOD) are an example of representative external sources for cube and DW projects [1,2,10,17]. LOD are a set of design principles for sharing machine-readable data on the Web for public administrations, business and citizens usages[1]. They use Semantic Web standards like RDF and Sparql query

[1] https://www.w3.org/DesignIssues/LinkedData.html.

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 287–302, 2019.
https://doi.org/10.1007/978-3-030-32065-2_20

languages. YAGO and DBpedia[2] are examples of most popular LOD portals. When the DW integrates data from outside the company, it can provide to decision makers a unified view of their internal and external environment which allows identifying new insights from data. To master the data of the company, a new profile has been created which is the Chief Data Officer (CDO) who is responsible of the development and execution of an overall data strategy for leveraging enterprise data to drive business value[3]. A Data Strategy aligns and prioritizes data and analytics activities with key organizational priorities and goals[4]. In this way, the CDO connects the business strategy to its IT capabilities and IT actors[5] achieving concrete tasks like cleaning of data, integration of sources, generation of reports, etc. An increasing number of tools are being developed to help technical and IT actors by new tools relying on advanced techniques (like ETL tools augmented on ML techniques[6] or DBMS advisors for the optimization and the tuning of the cube[7]). In some cases, such tools tend to replace existing IT actors. However, these technical actions are preceded by executive's strategic decisions (the CDO in the foreground) concerning the relevance of considering the chosen fragment of external LOD sources. CDO actors need methods and tools to assist them for taking the appropriate decisions.

In the context of cube design alimented by LOD sources, only technical issues have been investigated [1,2,6,9]. This is explained by the fact that existing studies assume that the CDO has decided about the relevance of external LOD source to consider for the data strategy executed. Consequently, these studies deal with the issue of LOD incorporation from a source-driven perspective, essentially because they had to deal with a new data source. However these studies neglected the managerial-driven perspective, where adding new external resources to the DW has an impact on the organization strategy, and thus should be considered carefully. Note that managerial level includes strategic & tactical levels [11]. Our proposal claims for a come-back to the essence of requirements driven approach philosophy (goal-oriented), where requirements engineering studies an organization as a whole and is not restricted to studying technical artifacts [21]. The evaluation of a strategy is achieved through the evaluation of the fulfillment of defined goals. This evaluation is crucial for measuring the success of an organization and is commonly carried out by analyzing metrics such as key performance indicators (KPIs) that must be identified from the beginning of the design process [20]. Our aim is to provide a bridge between the CDO vision connected to the company strategy with the IT vision connected to data managed. The incorporation of external data sources is seen in this paper as a trade-off between the added value at the managerial level and design complexity induced by the

[2] http://wiki.dbpedia.org/.
[3] https://www.ibm.com/downloads/cas/MQBM7GOW.
[4] https://www.dataversity.net/chief-data-officer-challenge-build-data-strategy/.
[5] https://www.ibm.com/downloads/cas/MQBM7GOW.
[6] https://www.tamr.com/blog/three-generations-of-ai/.
[7] https://aws.amazon.com/fr/blogs/machine-learning/tuning-your-dbms-automatically-with-machine-learning/.

opening of frontiers and the consideration of external resources. This complexity is present in different design tasks like source unification (e.g. graph formalism present for LOD sources), continuously checking the availability of external sources, verifying the veracity of the source, tracking its evolution, etc. If the decision of integrating external sources is connected to the managerial level, the CDO will be provided by relevant indicators that indicates if these efforts are worthy to deploy.

The problem is stated as follows: (a) considering a set of goals structured in a goal hierarchy, (b) their related KPIs, (c) and a semantic cube schema, our approach aims to answer the following question: *what is the impact of incorporating external LOD sources at the managerial and the technical levels*. Note that the existence of a semantic cube schema is an assumption made by different studies dealing with internal and external semantic sources [6]. In our vision, the estimated performance value of goals (estimated by their related KPIs) reflects the managerial level, and the technical level is reflected by the data design-artifacts of the semantic DW impacted by the introduction of external resources, i.e. concepts, instances and data-flows. For doing this, we first propose a meta-model aligning the three models: the goal model, the KPI model and the data model (concepts and data-flow) of the target cube. The performance value of KPIs is evaluated from the data sources (internal and external), and then propagated through the goal hierarchy to evaluate the degree of satisfaction of the global goal and thus to evaluate the organization strategy. From this vision, KPIs are central components linking data sources to the evaluation of defined goals reflecting the managerial situation adopted. The evaluation of KPIs performance using semantic web technologies required a thorough analysis for bridging the gap between proposed grammars for conventional KPI specification and the semantic web languages. Experiments are conducted based on QBAirbase[8], a multidimensional dataset presenting air pollution metrics. A case tool is also developed dedicated to the CDO for monitoring the incorporation of LOD through metrics illustrating both facets discussed (technical and managerial). The paper is structured as follows: Sect. 2 presents the related work. Section 3 presents the background related to goals and KPIs performance. Section 4 details the proposed meta-model. Section 5 describes the proposed approach. Section 6 presents different experiments conducted and the case tool implementing the approach. Section 7 concludes the paper.

2 Related Work

Literature related to the design of conventional DWs identified two maim approaches [18]: source-driven and requirements-driven approaches. Historically, the first category of approaches has been proposed before the second category each time a new source (Relational, XML, Tweets, LOD, etc.) emerged as a candidate for DW systems. This can be explained by the fact that designers have to investigate the source first for evaluating its worthiness. On the one hand, we

[8] http://qweb.cs.aau.dk/qboairbase/.

have proposed different studies that demonstrate the usability of requirements for different components of the DW system [7]. On the other hand, since the worthiness of LOD for DW projects has been established, we claim to a return to a requirements-driven approach when considering this external source.

Different studies have been devoted to the incorporation of LOD technology into OLAP space. Most of studies focused on solving some technical issues identified in this source, these issues are identified *during the design* or *during the querying* of the cube.

(i) During the design: some studies focus on the definition of the cube schema by managing: the unification of internal and external schemas [6] or the identification of the multidimensional role of the new external resources to integrate [1]. While other studies focus on the integration process by managing the definition of the ETL process used to unify internal and external sources. For example, [2] proposes an incremental fetching and storing of $\mathcal{LOD}$ and other external resources on-demand using an optimization process that determines a set of extractions that cover all the missing information with a minimum cost. [6] proposes a programmable semantic ETL framework from internal relational sources and linked data sources. *(ii) During querying:* these studies mainly focus on the definition of OLAP operators adapted for LOD datasets [17,19]. The result of OLAP queries defined on the DW are merged with results of Sparql queries defined on the LOD.

Because these studies are mainly source-driven, they only consider technical issues related to some defined design tasks. Contrary to existing studies, our proposal reconnects LOD sources incorporation to the goals conducting the cube design. Note that the proposed approach can be seen as complementary to existing studies cited, in the sense that it provides a complementary and managerial view of the decision of LOD incorporation in the OLAP cube of the company. For ensuring the usability of our approach with existing studies, we defined it at the conceptual level. This study is also complementary to our previous study [13] where a goal-driven approach has been defined for calculating the organizational value of LOD. This paper completes the study by managing the trade-off between technical and managerial levels, and proposes a method and a tool dedicated to the CDO.

3 Background

This section defines the main concepts related to goals and related KPIs.

Goals and Related KPIs. Goals represent a desired state-of-affairs, defined during strategic planning and pursued during business operation [3]. A Key Performance Indicator (KPI), is an industry term for a measure or metric (quantitative or qualitative) that evaluates performance with respect to some objective [3]. Indicators are used to measure the fulfillment of strategic goals. For example, the indicator "The rate of decrease/increase in annual mean of NO2 concentrations observed at traffic stations compared to year X-1" can measure the goal "Lower annual mean of NO2 concentrations observed at traffic stations in year

X". Choosing the right indicators for a given goal is a complex issue that has been treated in literature. In the current study, we assume that goals and their related KPIs are well defined.

The goals (resp. their related indicators) can be presented as hierarchies (Fig. 1) modeled using AND/OR tree hierarchies of sub-goals (resp. KPIs) to reflect a decisional situation. In the goal hierarchy, the satisfaction of a goal depends thus on the satisfaction of its subgoals. Influence relationships (also called contribution relationships) can be defined between goals, meaning that a goal's satisfaction may be affected positively or negatively by defined goals other than its subgoals.

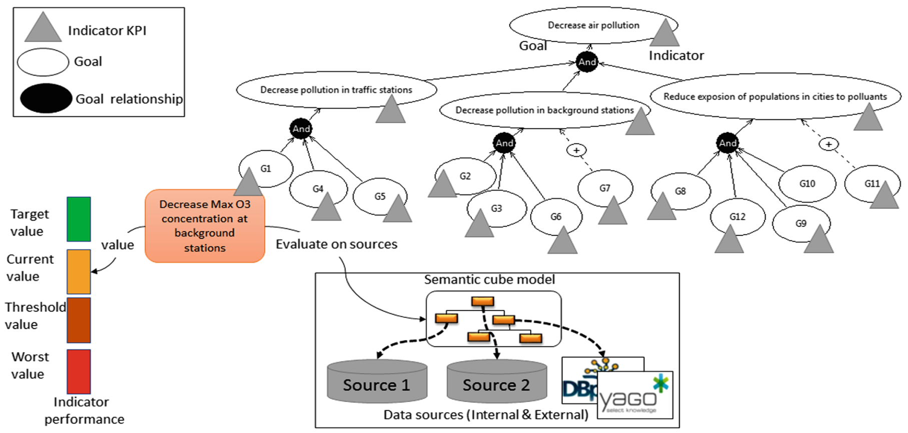
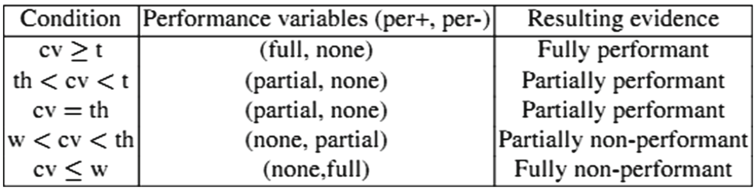


Fig. 1. Goals and related KPIs hierarchies

Indicators Values. The value of an indicator for an object depend on the values of indicators for objects one level lower in the hierarchy [3]. Values of indicators from a lower level to a higher level in a hierarchy are calculated using propagation technique, similar to label propagation in goal reasoning, where the satisfaction of a goal can be inferred from that of others in the same goal model. Each indicator, being a composite or component, has a current value which is evaluated using a set of parameters: target value, threshold value and worst value [3]. The current value is either supplied by users to conduct what-if analysis scenarios, or it can be extracted from data analysis which is the scenario considered in our approach. By this mean, indicators analysis are the bridge between data sources (internal and external) considered and the goals to achieve, which is the main statement of our study. Considering a hierarchy of indicators (Fig. 1), the current values of leaf indicators is extracted from data sources, and the values of non-leaf indicators is calculated using metric expressions (when possible) or estimated following a qualitative approach. The propagation of indicators values in the hierarchy from a lower level to higher levels (i.e. composite indicators) can be done using different quantitative and qualitative techniques depending on the availability of the domain information. Different studies present in detail three well-known techniques: conversion factors, range normalization, and qualitative reasoning [12]. In our approach, we use the qualitative reasoning because metric expressions are not always available and because this approach allows highlighting inconsistencies by identifying conflicts among goals, which is an important design indicator when dealing with external sources.

KPIs Values Propagation. Qualitative reasoning relies on two variables: positive performance (per+) and negative performance (per+) that are assigned to each indicator, and that take their values ranging in ("full", "partial", "none"). These values are assigned according to mapping rules defined in [12] and illustrated in Fig. 2. For instance, when the current value of an indicator is between its threshold value TH and the target value t, (per+) is set to "none" and (per-) is set to "partial", indicating that an evidence that the indicator is partially non-performant. The propagation of values to higher levels indicators is based on propagation rules defined in Fig. 3 [12]. For example, assuming that indicators I1, I2 and I3 are associated to goals G1, G2 and G3, and an AND relationship is defined G1: G2 AND G3, we need to identify the values of variables (per+ and per-) of the composite indicator I1 from its component indicators I2 and I3. In this case, (per+) of I1 is equal to the minimum(per+(I2), per+(I3)), and (per-) is equal to max(per-(I2), per-(I3)). Note that in this table, the (or), (+D), (-D), (++D), (- -D) cases are dual w.r.t. (and), (+S), (-S), (++S), (- -S) respectively [12].



Condition	Performance variables (per+, per-)	Resulting evidence
$cv \geq t$	(full, none)	Fully performant
$th < cv < t$	(partial, none)	Partially performant
$cv = th$	(partial, none)	Partially performant
$w < cv < th$	(none, partial)	Partially non-performant
$cv \leq w$	(none,full)	Fully non-performant

Fig. 2. Calculation of performance and result for indicators (t = target value, cv = current value, th = threshold value, w = worst value) [12]

In our approach, the set of leaf goals is considered incrementally to reflect the real situation where a company must integrate new requirements, which may require external resources in the cases in which internal resources are not sufficient to express the goal. The goal of our approach is to provide indicators reflecting the impact of considering external resources. For each goal, a KPI is defined. The performance of KPIs corresponding to leaf goals is calculated against data sources using Sparql queries, then propagated to higher level goals (see Fig. 1). The technical impact is evaluated by tracing data design artifacts impacted from the goal considered to the target cube.

	$(I_2, I_3) \xrightarrow{ET} I_1$	$I_2 \xrightarrow{+S} I_1$	$I_2 \xrightarrow{-S} I_1$	$I_2 \xrightarrow{++S} I_1$	$I_2 \xrightarrow{-S} I_1$
per+ (I_1)	min(per+(I_2), per+(I_3))	min (per+(I_2), P)	N	per+(I_2)	N
Per- (I_1)	max(per-(I_2), per-(I_3))	N	min (per+(I_2), P)	N	per+(I_2)

Fig. 3. Propagation of indicators performance values (N indicates None, P Partial) [12]

4 Proposed Metamodel for Connecting the Incorporation of LOD to the Managerial Level

The proposed metamodel is illustrated in Fig. 4. KPIs are the core component of this model since they provide the link between the decision of incorporating the LOD and the managerial level reflected by goals. The KPI model is borrowed from [15], it is mainly composed of the set of KPIs characterized by a name, status values and the Sparql query defining the schema which, when evaluated on the source, provides the current value of the KPI. Each KPI is defined to measure the performance of a goal, the KPI class is thus connected to the goal class. We have provided in previous studies [14] a detailed analysis of the DW design cycle, which allows us to have a global picture of connection between the various design artifacts used. Each goal is characterized by two coordinates (result and criteria). The goal model is connected to the data source model (internal and external) where each goal coordinates require a fragment of the data source for its definition (concept, property or expression using concepts and properties). The data flow model represents the process for integrating data sources into the target cube schema of the cube. This data flow (ETLWorkflow class) is defined by mappings identified by the designer. Each mapping is described on the basis of extraction operators, contextualization (from the external concept to

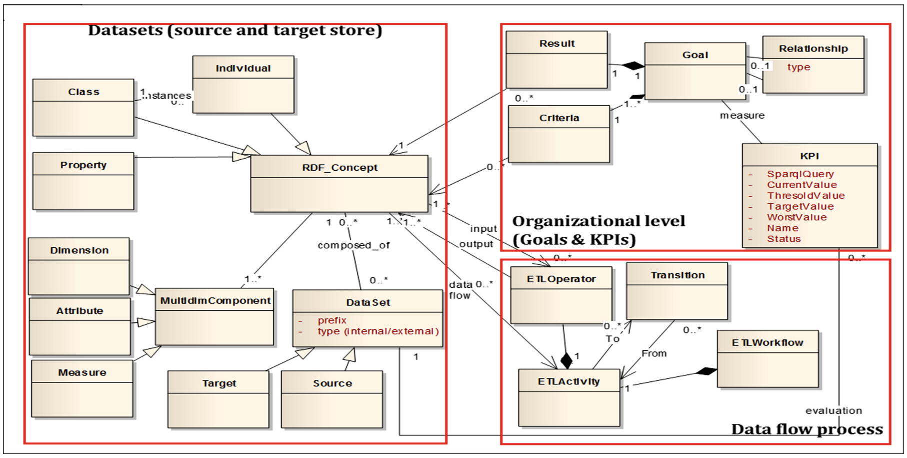


Fig. 4. Metamodel connecting Goal, KPI, Data and Data-flow models

an internal concept), transformation and loading (e.g. context, extract, retrieve, union, merge, etc.).) (class ETLOperator) ([4]).

Because the main language underlying LOD datasets is RDF, we defined data model (underlying sources and target cube) based on RDF model artifacts mainly composed of Concepts (class, property and instance), a multidimensional annotation (for the target schema), and the dataset to which the concept belongs. Each dataset is described by a unique URI. Different studies relies on the definition of an unified semantic schema (existing or constructed) to unify the sources [1,4,6]. It is this semantic scheme that we use to evaluate the actual values of the indicators. The process for constructing this schema is out of the scope of this study. Following a requirements driven approach philosophy, the set of goals are defined by decision makers and projected on the set of data sources, first internal sources and external sources when necessary. The goal model is thus connected to the data source model, where each goal coordinate (result and criteria) is defined using a data source artifact. The ETL model reflects the data flow from the set of sources to the target cube.

5 Proposed Approach for Measuring the Impact of LOD Incorporation in the Semantic Cube

Based on the proposed metamodel that aligns technical and managerial levels, we propose an approach illustrated in Fig. 5 which aim is to measure the impact of LOD incorporation in the semantic cube.

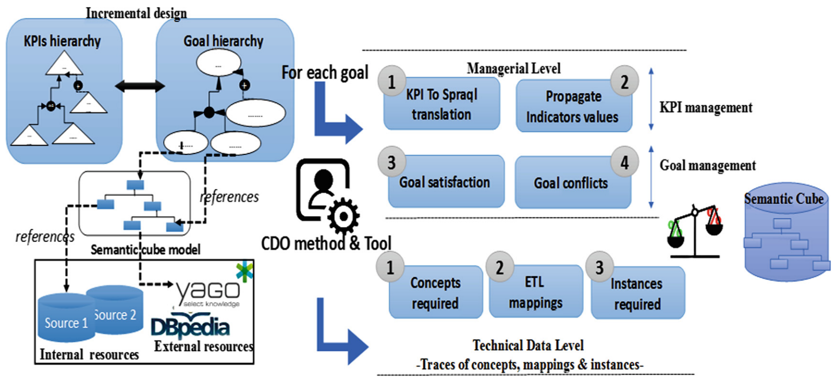


Fig. 5. General Approach to measure the impact of LOD incorporation in the semantic cube

5.1 Approach Overview

The following algorithm provides the main steps of measuring the impact of LOD incorporation at technical and managerial level. Assuming the existence of:

(1) a semantic Cube, and more precisely we assume the existence of the semantic schema of cube defined in RDF, unifying internal and required external schema. Note that this asumption is required and present in most studies defined a cube from external sources [6].
(2) the set of goals and related KPIs identifying the fragments of sources required. The algorithm starts by tracing the impact of each goal inserted on the fragments of sources using the metamodel proposed. Then, it defines the impact of each goal on the KPI level.

5.2 Impact at Technical Level

Algorithm 13 starts by tracing the impact of each goal inserted on the fragments of sources (presented by *SemCube*) using the model proposed in Fig. 4.

Using the following transitivity rule: *If object x is related to object y and object y is in turn related to object z, then object x is also related to object z*, *SemCube* can infer the different traces to consider. The rule is formalized in *SemCube* using SWRL[9] language.

Inputs: Goals and KPIs hierarchy (Set Goals and Set KPIs), semantic schema of the cube (*SemCube*) - External resources
Result: Technical & managerial Traces

```
1: for all $G_i \in$ Goals (and associated KPI) do
2:     Add leaf Goal to hierarchy
                                        ▷ Impact at technical level, cf. section 5.2
3:     Concepts := $\text{Trace}_{Concepts}$ ($G_i$)
4:     Extract $\text{Trace}_{Nodes}$ ($G_i$)  ▷ Extract Nodes (i.e. concepts and properties of the data model)
       related to each goal
5:     Identify internal and external nodes by their URI
6:     Extract $\text{Trace}_{Mappings}$ ($G_i$)     ▷ Extract mappings related to concepts of each goal
7:     Extract $\text{Trace}_{Instances}$ ($G_i$)    ▷ Extract instances related to concepts of each goal
                                        ▷ Impact at managerial level
8:     Formalize KPI                                          ▷ cf. section 5.3
9:     Translate KPI to Sparql query on SemCube               ▷ cf. section 5.3
10:    Measure positive and negative performance (Per+ and Per-) of the KPI   ▷ cf. section 5.3
11:    Values propagation in KPIs hierarchy                   ▷ cf. section 5.3
12:    Identify conflictual goals                             ▷ cf. section 5.3
13: end for
```

```
Trace(?T1, ?T2), Trace(?T2, ?T3) -> Trace(?T1, ?T3)
```

Each goal is related through its coordinates (result and criteria) to concepts and properties of the global schema (that are directly related to other concepts), these results and criteria are defined following the target schema, and they are alimented by data-flows defined in the meta-schema of *SemCube*. Following these connections, we can infer the set of traces to consider by introducing a new goal requiring internal and external resources. These traces present a relevant indicator to evaluate the design complexity of considering these resources.

[9] https://www.w3.org/Submission/SWRL/.

5.3 Impact at Managerial Level

The impact at the organization level is based on the core component linking technical and managerial levels: the KPIs.

1. KPIs Specification. An Indicator is defined as an expression that can be optionally projected using the multiple axis in the analysis cube or restricted to a certain data subspace [16]. For example, "Annual mean NO2 concentrations observed at traffic stations in year X compared to year X-1" is an indicator expression. Maté et al. [16] proposed a specification (in a structured natural language) for KPIs that we use in our approach. First, a restriction constrains the calculus of the KPI to define a data subspace (a subcube) in which the KPI value is calculated. The specification defines a KPI expression as simple or complex values that are obtained by applying successive binary operators (for example, "*minus*" in the previous example). Each KPI value is calculated by applying an optional unary operator ("mean" in the previous example) to a value across a set of dimension levels (e.g. "Year"). Values can be constrained to meet certain boolean conditions independently from the set of levels (e.g. "equal to 2008").

Note that the KPI of the new goal is evaluated on *SemCube* which is extended by new external resources. The extension is achieved by the designer (on the ontology schema). The most intuitive extension is to consider the result as a new fact and the criteria as dimensions for this fact. However, this issue is related to multidimensional annotation of semantic cube which has been extensively studied. The goal of evaluating KPIs at this semantic level is to obtain indicators about the relevance of considering these new goals (requiring external resources), before effectively considering them at the effective physical cube usually stored in a semantic DBMS and exploited by OLAP tools. Another advantage is that data-flow processes can be defined (to identify the technical impact of the new goal), but can be executed an optimized once the relevance of incorporating the new goal is effectively proven.

2. Translating KPIs to Sparql queries. The proposed specification facilitates communication between CDO and decision makers, but it is still difficult to translate such specification (in structured natural language) to a formal language like Sparql. Our goal was to find a suitable intermediate OLAP conceptual language that can support all constructs of the KPI specification defined in the previous section, and that can be easily translated into Sparql language. We chose cube query language (CQL) as an intermediary language, to achieve the translation of KPIs to Sparql queries. CQL is proposed in [5] as a conceptual language for querying data cubes for OLAP analysis, using the Semantic Web technologies, and provides a simplified way for expressing queries that can be translated into SPARQL expressions, following a nave approach. We use CQL to express KPIs at a conceptual level, KPIs are then translated into Sparql queries following study [8]. Note that in [8], CQL is expressed over QB4OLAP cubes, but can be easily adapted to RDF cubes.

We start by presenting CQL briefly. CQL provides a set of operators classified into two groups: (1) instance preserving operations (IPO) that navigate the cube instance to which the input cuboid belongs; and (2) instance generating operations (IGO) that generate a new cube instance. The Roll-up and Drill-down operations belong to the first group, while Slice, Dice and Drill-across belong to the second. Considering the following sets: C is the set of all the cuboids in a cube instance, D is the set of dimensions, M is the set of measures, L is the set of dimension levels, and B is the set of Boolean expressions over level attributes and measures. The operators are defined as follows: DICE(cube_name, Phy), ROLL-UP(cube_name, Dimension→level, (measure, aggregate_function)*), SLICE(cube_name, Dimension, ROLL-UP(cube_name, Dimension All, Aggregate_function)), DRILL-ACROSS(cube_name_1, cube_name_2) and MAP(cube_name, (measure, function)*).

For example, a dimension cube of QBOAirbase is defined in CQL as follows:
`CUBE QBOAirbase`
`DIMENSION Station LEVEL Station LEVEL City LEVEL Country`
`Station ROLL-UP to City City ROLL-UP to Country` For example, the following goal of QBOAirbase: "*maintain average concentration by type of station, city and year, expressed in g/m, under threshold X*", has as related KPI: "average concentration by type of station, city and year, expressed in g/m"

This KPI is translated into CQL language as follows: `C1 ← Slice(SemCube, Sensor) C5 ← RollUp(C1, (Station → City, Time → Year), AVG (Observation))`

This query is translated into the following Sparql query (using the algorithm we defined in [13], based on study [8]):

```
select avg(?avgno2) where {
SELECT ?type ?city ?pollutant (avg(?no2) as ?avgno2)  WHERE {
    ?s property:type ?type . ?obs schema:station ?s .
    ?obs rdf:type qb:Observation . ?obs ?p ?no2 .
    ?s schema:inCity ?city . ?obs schema:sensor ?sen .
    ?sen schema:measures ?comp . ?comp property:caption ?pollutant .}
     GROUP BY ?type ?city ?pollutant}
```

3. KPIs values propagation. In our approach, we use qualitative reasoning to propagate values of composite KPIs to component KPIs. We have detailed in Sect. 3 the details of this technique. We illustrate it in the next section through the proposed case study. Conflicts can be identified when an indicator is an evidence of an indicator is identified as "fully performant" from one path (for instance I1: I2 AND I3) and another evidence is identified as "non performant" from another path (for instance I4 influences I1, and following Fig. 2, the indicator is identified as "non performant"). Such information when the composite indicator is calculated relying on external source is important for the CDO, since she can decide to materialize these external resources until the indicator is fully performant.

6 Case Study

Our experiments are based on the following case study using QBAirbase[10], a multidimensional provenance-augmented version of the Airbase dataset. QBOAirbase represents air pollution information as an RDF data cube, which has been linked to the YAGO and DBpedia knowledge bases. QBOAirbase is based on the version 8 of the Airbase dataset[11]. In QBOAirbase an observation is associated to a measurement of a single pollutant and is defined by its coordinates in a set of dimensions. For instance, in QBOAirbase the concentration of an air pollutant is measured at a location, at certain time, and under some sensor configurations. The QBOAirbase's cube structure is defined in http://qweb.cs.aau.dk/qboairbase/QBOAirbase.ttl and its visual representation is illustrated in http://qweb.cs.aau.dk/qboairbase/. QBOAirbase proposes a set of queries that we used as the set of goals. Some queries have been adapted to rely on external sources and mainly on YAGO. Based on these queries, we have constructed the goal hierarchy illustrated in Fig. 1. Some examples of leaf goals are: "Decrease the annual mean NO2 concentrations observed at traffic stations" or "Decrease the ratio of urban population that has been exposed to more than 18 um/g3 of O3". Our experiments was conducted by considering the set of goals incrementally, to analyze the impact of adding new goals, requiring internal and/or external resources, on the cube design at both levels (managerial and technical levels). Table 1 illustrates the impact of goals consideration at the managerial level. The table contains for each leaf goal: (1) the KPIs identified for each goals, (2–5) the set of values: worst value, threshold value, target value (that we estimated randomly) and current values calculated from the internal and external sources (more precisely on the semantic schema of the cube *SemCube*). (6 & 7) Per+ and Per- values of leaf goals (calculated based on Fig. 2), (8) KPI result of leaf goals, (9–11) KPI result of intermediate goals in the hierarchy (SubGoal1, SubGoal2 & SubGoal3) calculated using Fig. 3, (13) KPI result of the global goal also calculated based on Fig. 3, (14) the identification of a conflict due to the incorporation of the goal (recall that goals are considered incrementally which is

Table 1. Calculate performance and result for indicators (t = target value, cv = current value, th = threshold value, w = worst value)

KPIs	Worst value	Threshold value	Target Value	Current value	Per+	Per-	Goal result	SubGoal1 result	SubGoal2 result	SubGoal3 result	Global goal result	new conflicts
1	-2	-3	-5	-1,77	none	full	FNP	FNP	/	/	fully non performant	no
2	-1	-2	-3	-1,02	none	partial	PNP	FNP	PNP	/	fully non performant	no
3	-1	-2	-3	-1,57	none	partial	PNP	FNP	PNP	/	fully non performant	no
4	-2	-3	-5	-2,75	none	partial	PNP	FNP	PNP	/	fully non performant	no
5	-2	-3	-5	-1,93	none	partial	PNP	FNP	PNP	/	fully non performant	no
6	-1	-2	-3	0,78	none	full	FNP	FNP	FNP	/	fully non performant	no
7	40	45	50	40,92	none	partial	PNP	FNP	Conflict	/	fully non performant	yes
8	190	170	160	173,61	partial	none	PP	FNP	FNP	PP	fully non performant	no
9	>0	0	<0	-387,00	full	none	FP	FNP	FNP	PP	fully non performant	no
10	> 0	0	<0	0	partial	none	PP	FNP	FNP	PP	fully non performant	no
11	2	0	-2	-2	full	none	FP	FNP	FNP	PP	fully non performant	no
12	300	280	270	268,85	none	partial	PNP	FNP	FNP	conflict	conflict	yes

[10] http://qweb.cs.aau.dk/qboairbase/.

[11] https://www.eea.europa.eu/data-and-maps/data/airbase-the-european-air-quality-database-8.

the most usual scenario). Note that, in the content of the table and in the next figures, PP indicates Partially Performant, PNP: Partially Non Performant, FP: Fully performant and FNP: Fully non performant.

The table illustrates to the CDO how the incremental incorporation of goals affects the design based on the KPIs results. The table indicates that the global goal becomes mainly fully non performant for all goals, meaning that the set of goals should be monitored until their fulfillment. The CDO might decide to materialize the external source fragment until the goals are fulfilled. KPIs values of intermediate goals (SubGoal1 & SubGoal2) are usually non performant whereas SubGoal3 is partially performant. The consideration of Goal7 introduces a conflict in the goal hierarchy. The conflict occurred because the intermediate goal (SubGoal2) is performant when propagating the values of leaf goals (Goal2, Goal3 and Goal6), and it is performant when propagating the influence of (Goal7) (see Fig. 1). Goal12 also introduces a conflict in the goals hierarchy. These conflicts indicate to the CDO that either her goal strategy must be reviewed (one branch cannot be satisfied), or that she must consider the goal (and its external fragment) causing the non performance of the goal and thus the conflict until the satisfaction of the goal. Figure 6 illustrates the KPI performance values for each leaf goal, and the impact on the KPIs performance of subgoals (intermediate) and of the global goals.

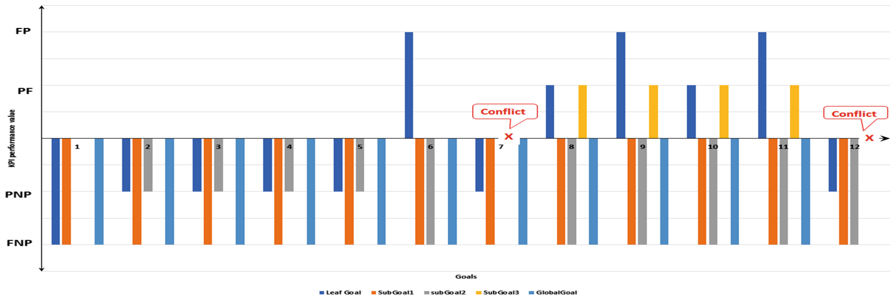


Fig. 6. KPIs performance values for leaf goals, subgoals and the global goal

We have also developed a case tool dedicated to the CDO to illustrate our approach of considering RDF cubes alimented by internal and external sources. Our goal is to provide the CDO a fully requirement-driven approach that provides indicators to the CDO about the impact of considering new requirements that require external sources, on her design process. The tool is based on the definition of SemCube schema, the set of goals and their KPIs (provided by the CDO or the designer). Figure 7 illustrates the main interfaces of the tool. Both aspects (managerial and technical) are illustrated in the tool. The first part of the tool shows the impact of goals at technical level (requirements, nodes and mappings considered), the second part of the tool illustrate the performance of

KPIs for each goal. Concerning the performance values of KPIs, we have represented the last iteration in the tool, and the goal that caused a conflict. The tool also allows the CDO to redefine design tasks by adding/modifying the set of requirement, nodes and ETL processes. The tool is developed in Java, SemCube schema is accessed using Jena API.

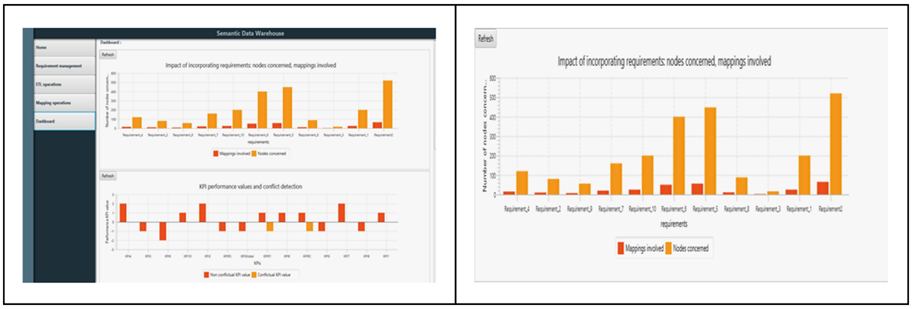


Fig. 7. Case tool interfaces

7 Conclusion

In this paper, we first proposed a metamodel aligning managerial and technical levels of cube design. Based on this metamodel, we proposed an approach and a tool dedicated to the CDO for assisting her during the process of LOD incorporation in the DW system of the company. Considering external sources like LOD is an organization decision that should be connected to the goals and strategy of the company, before considering technical aspect of LOD incorporation. Our approach identified KPIs as key components that connects the company strategy and the data sources that are considered. One key step in this approach is the transition of a KPI specification on LOD sources. Experiments and a tool for the CDO are presented that measure using different metrics the tradeoff between the technical efforts required by the consideration of a LOD fragment and its impact on the strategy represented by a set of goals. This study should be completed by a materialization strategy according to the indicators provided, the management of complex goal hierarchies including weights in the branches and by the validation of the approach on some real companies DW.

References

1. Abelló Gamazo, A., Gallinucci, E., Golfarelli, M., Rizzi Bach, S., Romero Moral, O.: Towards exploratory OLAP on linked data. In: SEBD, pp. 86–93 (2016)
2. Baldacci, L., Golfarelli, M., Graziani, S., Rizzi, S.: Qetl: an approach to on-demand etl from non-owned data sources. DKE **112**, 17–37 (2017)

3. Barone, D., Jiang, L., Amyot, D., Mylopoulos, J.: Reasoning with key performance indicators. In: Johannesson, P., Krogstie, J., Opdahl, A.L. (eds.) PoEM 2011. LNBIP, vol. 92, pp. 82–96. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24849-8_7
4. Berkani, N., Khouri, S., Bellatreche, L.: Value-driven approach for designing extended data warehouses. In: DOLAP (2019)
5. Ciferri, C., Ciferri, R., Gómez, L., Schneider, M., Vaisman, A., Zimányi, E.: Cube algebra: a generic user-centric model and query language for olap cubes. Int. J. Data Warehouse. Min. (IJDWM) **9**(2), 39–65 (2013)
6. Deb Nath, R.P., Hose, K., Pedersen, T.B.: Towards a programmable semantic extract-transform-load framework for semantic data warehouses. In: DOLAP, pp. 15–24 (2015)
7. Djilani, Z.: Donner une autre vie à vos besoins fonctionnels : une approche dirigée par l'entreposage et l'analyse en ligne. (Give Another Life to Your Functional Requirements : An Approach Drvicen by Warehousing and Online Anaysis). Ph.D. thesis, École nationale supérieure de mécanique et d'aérotechnique, France (2017)
8. Etcheverry, L., Vaisman, A.: Querying semantic web data cubes. In: AMW, pp. 11–23 (2016)
9. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10160-6_5
10. Gallinucci, E., Golfarelli, M., Rizzi, S., Abell, A., Romero, O.: Interactive multidimensional modeling of linked data for exploratory olap. Inf. Syst. **77**, 86–104 (2018)
11. Gray, C.S.: Modern Strategy, vol. 42. Oxford University Press, Oxford (1999)
12. Horkoff, J., et al.: Strategic business modeling: representation and reasoning. SSM **13**(3), 1015–1041 (2014)
13. Khouri, S., Aouimer, Y., Bellatreche, L., Ghomari, A.R.: Intgrer les LOD dans un cube de données : transformer une action technique en une dcision organisationnelle. In: To appear in Journes Entrepts de Donnes et Analyse en ligne (EDA 2019). RNTI (2019)
14. Khouri, S., Semassel, K., Bellatreche, L.: Managing data warehouse traceability: a life-cycle driven approach. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 199–213. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19069-3_13
15. Maté, A., Trujillo, J., Mylopoulos, J.: Conceptual modeling for indicator selection. Conceptual Modeling Perspectives, pp. 55–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67271-7_5
16. Maté, A., Trujillo, J., Mylopoulos, J.: Specification and derivation of key performance indicators for business analytics: a semantic approach. DKE **108**, 30–49 (2017)
17. Matei, A., Chao, K.-M., Godwin, N.: OLAP for multidimensional semantic web databases. In: Castellanos, M., Dayal, U., Pedersen, T.B., Tatbul, N. (eds.) BIRTE 2013-2014. LNBIP, vol. 206, pp. 81–96. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46839-5_6
18. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: DOLAP, pp. 3–10 (2006)
19. Saad, R., Teste, O., Trojahn, C.: OLAP manipulations on RDF data following a constellation model. In: 1st International Workshop on Semantic Statistics (2013)

20. Silva Souza, V.E., Mazn, J.N., Garrigs, I., Trujillo, J., Mylopoulos, J.: Monitoring strategic goals in data warehouses with awareness requirements. In: ACM SAC, pp. 10–75 (2012)
21. Tort, F., Teulier, R., Grosz, G., Charlet, J.: Ingénierie des besoins, ingénierie des connaissances: similarités et complémentarités des approches de modélisation. In: Journées francophones d'ingénierie des connaissances, pp. 263–275 (2000)

Implementing Window Functions in a Column-Store with Late Materialization

Nadezhda Mukhaleva[2], Valentin Grigorev[1,2], and George Chernishev[1,2,3](✉)

[1] Information Systems Engineering Lab, JetBrains Research, Saint Petersburg, Russia
valentin.d.grigorev@gmail.com, chernishev@gmail.com
[2] Saint Petersburg University, Saint Petersburg, Russia
nmukhaleva@gmail.com
[3] National Research University Higher School of Economics, Moscow, Russia
https://research.jetbrains.org/groups/information_lab, http://english.spbu.ru/, https://spb.hse.ru/en/

Abstract. A window function is a generalization of the aggregation operation. Unlike aggregation, the cardinality of its output is always the same as the cardinality of input. That is, the semantics of this operator imply computing values for extra attributes for each row, depending on its context, either expressed by a sliding window or a previously evaluated row. Window functions are a very powerful tool, which is also popular among data analysts and supported by the majority of industrial DBMSes. It allows to gracefully express quite complex use-cases, such as running sums and averages, local maximum and minimum, and different types of ranking. Since they can be expressed without self-joins and correlated subqueries, their evaluation can be performed much more efficiently.

In this paper we discuss an implementation of window functions inside a disk-based column-store with late materialization. Late materialization is a technique that aims to keep tuple reconstruction back from individual columns as long as possible. Initially popular in the late 00's, it is rarely considered nowadays. However, in case of window functions it allows to substantially lower memory footprint. Another contribution of this paper is the application of a segment tree to computing RANGE-based window functions.

Keywords: Window function · Analytical function · Aggregation · Column-store · Query processing · Late materialization · OLAP · PosDB

1 Introduction

A column-store is a type of DBMS designed specifically for handling analytic applications. Its core idea is to store each attribute separately, either on disk or in memory. This type of storage allows to implement the so-called lightweight compression schemes [9] efficiently due to the resulting data homogeneity. Column-stores both store and operate on data in columnar form. Processing style allows to classify column-stores into early and late materializing systems [2]. In the former, data is stored and processed in columnar form only on the lowest levels of the operator tree [8]. Usually, it happens as follows: each column is read, decompressed and filtered. Then, all columns corresponding to attributes of a single table are "glued" together (i.e. tuples are reconstructed), after which processing continues similarly to a row-store. In the latter, tuple reconstruction is delayed to the latest possible time. Up until this moment, the system operates on positions. The majority of existing column-stores employ early materialization, and there is only a handful of systems that support late materialization [2].

Window function (or analytic function) is a concept that was proposed in reference [3] and later became a part of the SQL:2003 standard. Its evaluation is as follows: similarly to aggregation, data is partitioned into several groups. Next, a sort may be applied to data from each group. The third step depends on the specific window function. One class of window functions uses framing, which is a process described as follows: (1) a sliding window is determined for each row, (2) an aggregation is performed for the data in the window, (3) the result is "added" to the current row. Another class operates on an entire group: it computes new values for each row using the ones of the previously processed row (e.g. `RANK`).

In this paper, we discuss the implementation of window functions inside a column-store with late materialization that supports on-demand reading of individual columns. Employing late materialization may allow to speed up processing of such queries and reduce memory footprint, which is very important for this operation. We validate our approach by implementing it inside PosDB [5,6]—a distributed disk-based column-store with late materialization—and comparing it with PostgreSQL.

Overall, the contribution of this paper is the following:

1. An adaptation of the window function processing approach for column-stores with late materialization. We present three different strategies which are sufficiently generalized to be implemented in any column-store that allows per-attribute data reading.
2. A model for estimating memory requirements for each of these strategies.
3. An enhancement of the segment tree technique for processing `RANGE`-based window functions.

2 Window Function Processing: Approaches and Algorithms

In this section, we discuss how to design the operator for window function evaluation. A comprehensive overview of existing approaches can be found in the article [11]. Overall, `Window Operator` can be implemented using two algorithms:

classic and segment tree-based. These algorithms have a significant common part which is as follows: at first, partitioning and in-group ordering is performed. Next, groups are iterated over and each of them is processed independently of others. The distinction between these two algorithms is the group processing itself. The classic algorithm goes through tuples belonging to a single group while performing the following. For each tuple it computes frame bounds and then evaluates the window function over the data belonging to the frame. There are two possible approaches to this: naive and cumulative. The naive approach is straightforward: it calculates data from scratch for every frame instance. On the other hand, the idea of cumulative approach is to store the results of processing of the previous frame and to reuse them to evaluate current frame faster. It is very efficient in case of the `SUM` window function: the result for the previous frame is saved and using only one or two (one if any border of the frame is fixed and two otherwise) arithmetic operations allows to obtain a result for the current frame[1].

In case of the `MIN` and `MAX` window functions, the cumulative approach can be implemented by preserving the previous frame using a binary search tree. This is not as promising, but still can be useful, especially for large windows. It is straightforward to find window bounds defined by the `ROWS` clause while for the `RANGE` one they can be found using binary search.

As it was already mentioned earlier, another way of window processing is based on the segment tree data structure [1]. This approach is relatively novel: it has been proposed in 2015 in the paper [11]. This approach is as follows: at first, a segment tree is created from group data, and then tuples are iterated over. However, instead of computing over the current frame, a request is issued to the segment tree. This approach allows to efficiently evaluate window functions that have frame borders depending on the current row and to implement intra-group parallelism.

In the original paper, it was considered only for the `ROWS` framing. In our paper, we propose a slightly generalized segment tree that can be utilized for `RANGE`-based window functions too. Details of this generalization are described in the Sect. 3.2. Window functions that do not require framing can be evaluated with a simplified version of the classic algorithm.

3 Proposed Approach

3.1 Adapting Classic Algorithm for PosDB

While in row-stores and column-stores with early materialization, the algorithm of the `Window Operator` is defined quite clearly, systems with late materialization can offer a variety of options. The variations are largely associated with the point of materialization inside the operator.

[1] Note that it is assumed here that frame offset does not depend on the current row value. Otherwise, the cumulative approach is still attractive, but not as dramatically.

In PosDB, every query plan consists of two parts: positional- and tuple-oriented. If `Window Operator` is located after aggregation in a query plan, then its inputs are tuple blocks, and evaluation can be performed by one of previously described algorithms without any changes. But if `Window Operator` receives positional data, we have to decide when tuples should be materialized. This task is not as simple as it may seem. There are several approaches possible.

Strategy 1. Tuples are materialized during hash table population. All of the subsequent stages of the algorithm are identical to the row-store case.

The next one has only keys materialized during hash table population, and at the same time positional data is stored as values. In some cases, this allows to significantly reduce the size of the hash table. It is important to emphasize that the ordering step ceases to be a separate step of processing and becomes a part of the evaluation step. Thus, for each group, processing should start with the ordering. Further steps can be done in a number of ways.

Strategy 2a. At the beginning of group processing, all required attributes are materialized. Afterwards, tuples are sorted and window functions evaluation is performed as usual. Tuples are materialized only for one group at a time.

Strategy 2b. At the beginning of group processing, only attributes required for ordering are materialized and ordering is performed. After this we can move through associated positions and materialize data on demand. This strategy is not implemented yet since it requires a new execution model for efficient implementation, but still, it looks quite promising for window functions over a fixed-size frame.

It is reasonable to use **Strategy 1** if positions received by the `Window Operator` are ordered, since corresponding values can be read by a sequential scan. For example, such situation occurs when window functions are evaluated on a single table, i.e. the query does not contain joins. In other cases all these strategies require an equal number of I/O operations, so there should be no significant difference between them in terms of processing time. At the same time, the amount of required memory can vary substantially.

Let us estimate the amount of memory required by all these strategies. It is necessary to introduce several variables and functions for estimation:

- A—a set of all attributes which have to be materialized in some way;
- A_k—a set of partitioning attributes;
- A_{sort}—a set of sorting attributes;
- A_{aggr}—a set of attributes for which window functions are being evaluated[2];
- N—number of logical rows in the input;
- G_{key}—group corresponding to partitioning key key as a list of logical rows;
- $|G_{key}|$—number of logical rows in the group G_{key};
- $G^{max} = \underset{G_{key}}{\arg\max}\ |G_{key}|$
- M—number of groups;
- function size_t—returns size of tuple from the corresponding set of attributes;

[2] In our implementation, several window functions can be processed at once if they are defined over the same window.

– function size_p—returns size of logical row of positions for corresponding set of attributes; actually it is determined by the amount of tables joined before `Window Operator`.

In **Strategy 1**, materialized data is stored in the hash table and processing is run directly on it. It requires

$$\overbrace{M \times \text{size}_t(A_k)}^{\text{hash table keys}} + \overbrace{N \times \text{size}_t(A_{sort} \cup A_{aggr})}^{\text{hash table data}}.$$

In **Strategy 2a**, only tuples for keys are materialized during hash table population. The data itself is stored in the positional representation. Other attributes are being materialized during group processing, while dynamically deleting already read positions. Thus, this strategy requires

$$\overbrace{M \times \text{size}_t(A_k)}^{\text{hast table keys}} + \overbrace{N \times \text{size}_p(A_{sort} \cup A_{aggr})}^{\text{hash table data}} + \\ \overbrace{G^{max} \times \Big(\text{size}_t(A_{sort} \cup A_{aggr}) - \text{size}_p(A_{sort} \cup A_{aggr})\Big)}^{\Delta \text{ for materialization with dropping processed positions}}.$$

Utilizing this strategy to process several window functions with the same window but over different attributes can result in significant performance improvement.

Strategy 2b is a further enhancement of the same idea. Here, on the group processing stage only sorting attributes are materialized and thus, even better results are obtained:

$$\overbrace{M \times \text{size}_t(A_k)}^{\text{hash table keys}} + \overbrace{N \times \text{size}_p(A_{sort} \cup A_{aggr})}^{\text{hash table data}} + \\ \overbrace{G^{max} \times \text{size}_t(A_{sort})}^{\text{materialized sorting attributes}} + \overbrace{\text{size}_t(A_{aggr}) \times \mathit{window\_size}}^{\text{other attributes materialized for the window}}$$

3.2 RANGE-Based Window Functions

At first, let us discuss implementation details of the segment tree data structure [1]. In literature, it is usually described for specific operations, such as `SUM`, `MIN` or `MAX`. But here, a general solution is required and this leads to conspicuous implications.

It is common to implement the segment tree on the base of an array with an implicit tree structure, since the segment tree is always a complete tree. For a complete tree, an array is the most space-efficient representation[3].

However, an issue arises: if the tree is not a perfect[4] binary tree (i.e. the last level is not completely filled, or, in other words, the original array size is not a

[3] In an array-based implementation, we store just data without auxiliary information such as pointers to children which are necessary to describe an arbitrary binary tree.
[4] https://xlinux.nist.gov/dads//HTML/perfectBinaryTree.html.

power of 2), then some cells in the array are left uninitialized. Usually, when a segment tree data structure is being discussed, it is considered in a form tuned for a particular operation. Here, however, it is necessary to consider a number of operations. First of all, in order to properly initialize this array, an identity element should be chosen. Obviously, it depends on the operation: 0 for `SUM`, $-\infty$ for `MAX`, $+\infty$ for `MIN`, etc.

It is reasonable to not store a chunk of the tree that consists only of identity elements. Instead, it is convenient to "overload" the access to tree elements and return a "virtual" value if an out-of-real-bounds element is requested.

Furthermore, it is easy to see that employing a segment tree leads to reordering of the sequence of operations. Thus, it is necessary to require associativity of the operation for which the segment tree is built. As the result, a segment tree requires an underlying data type with the corresponding operation to be a monoid [10].

It is quite obvious that having a tuple of monoids, we can create a new monoid whose operation works with tuples by applying corresponding operations in an element-wise manner. This approach allows to efficiently process queries with several different window functions defined over the same `OVER` clause in case of `ROWS` framing. For the `RANGE` framing, it is only reasonable to utilize this approach if several window functions over the same attribute need to be evaluated.

Next, having discussed data organization in the segment tree, we are going to consider data processing. The construction of a segment tree is quite straightforward. A detailed description of this process can be found in the reference [1].

Now, let us consider processing of queries in a segment tree. It is performed via a recursive function `evaluateSegment`, whose pseudocode and description are given later. To correctly start it, the `evaluateFrame` function shown in the Listing 1 is used. It initiates a recursive function on the root of tree that has the current segment covering the whole bottom level of the tree. Next, consider `evaluateSegment` function—the classic recursive algorithm. As input, it receives the current position in the tree `index`, a monoid (data type, identity, and operation), segment borders corresponding to the current node `cLeft` and `cRight`, and borders of the requested frame `fLeft` and `fRight`. Its output is a set of requested values over the specified frame. The algorithm itself is very simple: if current segment is correct and is not equal to the requested frame, then we split it in half and transfer control to children recursively. In this listing, monoid.op is the corresponding operation of monoid, e.g. `SUM`, `MAX`, etc.

To support `RANGE`-based window function processing with segment tree, we have to generalize the algorithm described above. The interface modification is straightforward—we have to replace integer-valued `fLeft` and `fRight` with parameters corresponding to the processed attribute type. Also, several additional functions and variables have to be defined:

– `nLeaves`—number of existing leaves on the bottom level, equal to the size of original array;
– `getLeafValue` function—get k-th element of the bottom level;

– `getLeafOrMax` function—call `getLeafValue` if index corresponds to existing value and return the last element of the bottom level if index is out-of-range.

Listing 1. Classic `evaluateSegment` algorithm

```
// Recursion initialization
function EVALUATEFRAME(monoid, fLeft, fRight)
    return EVALUATESEGMENT(0, monoid, 0, $2^{\lceil \log_2 array\_size \rceil}$ - 1, fLeaft, fRight)
end function

function EVALUATESEGMENT(index, monoid, cLeft, cRight, fLeft, fRight)
    if fLeft > fRight then
        return monoid.identity
    end if
    if fLeft = cLeft and fRight = cRight then
        return GETVALUE(index)
    end if
    m ← (cLeft + cRight)/2
    return monoid.op(
        EVALUATESEGMENT($2 \cdot$ index + 1, cLeft, m, fLeft, MIN(fRight, m)),
        EVALUATESEGMENT($2 \cdot$ index + 2, m + 1, cRight, MAX(fLeft, m + 1), fRight)
    )
end function
```

The idea of recurrent tree traversal remains largely the same, but several changes are introduced. Firstly, we return the identity element if the left bound of the frame comes out of bounds. In comparison to the classic algorithm, it is necessary to explicitly check this. Furthermore, an equality check between the current segment and the requested frame should be replaced with an inclusion check, since it is a part of `RANGE` behavior. Furthermore, all comparisons require to wrap the current segment borders in `getLeafOrMax` calls. The resulting algorithm is presented in Listing 2.

The introduced changes are very straightforward. Nevertheless, the authors of paper [11] where the segment tree based algorithm was suggested did not consider the `RANGE` case. It is rather peculiar since the `RANGE` case looks inherently more suitable for processing with the segment tree based algorithm. `ROWS` framing with “floating” borders is very rare, while `RANGE` features such borders by its definition.

Moreover, the following simple optimization can be applied to `RANGE`-based window functions. The `RANGE` case requires ordering on a working attribute, so if attribute values are not unique, then equal values are co-located and evaluation should be run only once per unique value. For low cardinality data, this optimization may result in dramatic performance improvement. This idea is quite similar to the cumulative approach, but it allows to not compute frame borders for each value.

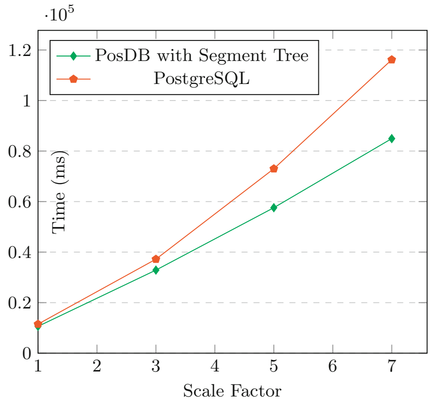
4 Experiments

Experimental evaluation was performed on a PC with the following characteristics: 4-core Intel®Core™ i5-7300HQ CPU @ 2.50 GHz, 8 GB RAM, running Ubuntu Linux 18.04.2 LTS. We have used PostgreSQL 11.3 as a baseline for comparison. For our experiment, we have constructed the query template shown below. It is based on the `LINEORDER` table from the SSB benchmark [12].

Listing 2. `evaluateSegment` algorithm for value ranges

```
function EVALUATESEGMENT(index, monoid, cLeft, cRight, fLeft, fRight)
    if fLeft > fRight or cLeft >= nLeaves then
        return monoid.identity
    end if
    if fLeft <= GETLEAFVALUE(cLeft) and
          fRight >= GETLEAFORMAX(cRight) then
        return GETVALUE(index)
    end if
    m ← (cLeft + cRight)/2
    return monoid.op(
        EVALUATESEGMENT(2 · index + 1, cLeft, m, fLeft, MIN(fRight, GETLEAFORMAXm))),
        EVALUATESEGMENT(2 · index + 2, m + 1, cRight, MAX(fLeft, GETLEAFORMAX(m + 1), fRight))
end function
```

`@offt`	DBMS	SF=1	SF=3	SF=5	SF=7
10	PosDB	10611	32888	57484	84935
	Postgres	11498	37205	73003	116160
100	PosDB	11454	35979	63658	93743
	Postgres	11536	38046	65834	111100
1K	PosDB	11543	36390	64042	95245
	Postgres	11828	38192	66061	113689
10K	PosDB	12116	38001	67239	100130
	Postgres	11909	38460	67449	113798
100K	PosDB	12713	39973	70159	105051
	Postgres	11924	38552	N/A	N/A
1M	PosDB	13272	41552	72982	107273
	Postgres	N/A	N/A	N/A	N/A
10M	PosDB	12693	39580	69677	101774
	Postgres	N/A	N/A	N/A	N/A



```
SELECT lo_orderpriority , SUM( lo_ordtotalprice ) OVER (
   PARTITION BY lo_orderpriority ORDER BY lo_ordtotalprice
   RANGE BETWEEN @offt PRECEDING AND @offt FOLLOWING) AS sum
FROM lineorder ORDER BY lo_orderpriority ASC
```

where `@offt` varies in range of [10, . . . , 10 M].

This experiment is quite simple and only demonstrates the attractiveness of a segment tree-based approach for processing of `RANGE`-based window functions. We believe that for a detailed performance analysis, a special benchmark has to be developed.

The aforementioned queries were run on PosDB and PostgreSQL with the SSB scale factors 1–7. The results of our experiments are presented in the table.

PosDB and PostgreSQL show approximately equal results on small scale factors (SF): PosDB wins for a small `@offt`, and PostgreSQL wins for large. Increasing SF leads to increasing advantage of PosDB and increasing `@offt` allows PostgreSQL to catch up, but not to outperform. At the same time, having a large `@offt` leads to unresponsive behavior of PostgreSQL (timeout was set to 10 min). Increasing SF leads to freezing on smaller window sizes. To the right of the table we present a graph comparing performances of the systems at `@offt`=10. It demonstrates the benefits of column-stores with late materialization.

Note that since this query favours a sequential scan, we implement it using **Strategy 1**. A detailed evaluation of **Strategies 1, 2a**, and **2b**, as well as assessment of performance impact of "wide" join indexes is the subject of future work. Currently, we anticipate that random data accesses spawned by **Strategies 2a, 2b** threaten to degrade query performance in some cases. However, it depends on a number of parameters: attribute sizes, selectivities of predicates, data distribution, and so on. We believe that at least in a part of cases our approach will still be beneficial, and a proper cost model will highlight it.

5 Related Work

Despite the fact that window functions were proposed almost 20 years ago, there is a surprisingly low number of works on the subject. They can be classified into two groups:

Designing the Operator Itself. Cao et al. [4] consider a case when a single query contains several window functions. The proposed approach is to reuse grouping and ordering steps. At first, the authors consider two methods of tuple ordering for a single window—hashed sort and segmented sort. They discuss their properties and applicability. Finally, they propose an optimization scheme for handling several window functions, which generates an evaluation schedule. Wesley and Xu [13] propose to reuse the internal state between adjacent frames for computing holistic windowed aggregates. A holistic function is a function that cannot be decomposed using other functions. Therefore, `MEDIAN` or `COUNT DISTINCT` are holistic and `SUM` or `MIN` are not. Speeding up the evaluation of such window aggregates is a relevant problem since their computation requires looking at all data at once. A paper by Leis et al. [11] describes an efficient algorithm for the whole window function operator. It considers existing approaches for aggregate computation, as well as proposes a novel one, based on a segment tree. Finally, an efficient parallelization of all steps is discussed.

Window Functions and External Optimization. Coelho et al. [7] addresses reshuffling in a distributed environment for efficient processing of window functions. The authors utilized histograms to assess the size of the prospective groups and their distribution between the nodes. Zuzarte et al. [14] discuss how and when it is possible to rewrite a correlated subquery using window functions. Such rewriting can significantly improve query performance.

6 Conclusion

In this study, we have discussed the implementation of window functions in a column-store with late materialization. We have proposed three different strategies, and for each of them we have provided a model for estimating the amount of required memory. We also present an enhancement of the segment tree technique for processing `RANGE`-based window functions. Experimental comparison with PostgreSQL has demonstrated the viability of this technique.

References

1. CP-Algorithms: Segment Tree. https://cp-algorithms.com/data_structures/segment_tree.html
2. Abadi, D., Boncz, P., Harizopoulos, S.: The Design and Implementation of Modern Column-Oriented Database Systems. Now Publishers Inc., Hanover (2013)
3. Bellamkonda, S., Bozkaya, T., Gupta, B.G.A., Haydu, J., Subramanian, S., Witkowski, A.: Analytic Functions in Oracle 8i. Technical report (2000). http://infolab.stanford.edu/infoseminar/archive/SpringY2000/speakers/agupta/paper.pdf
4. Cao, Y., Chan, C.Y., Li, J., Tan, K.L.: Optimization of analytic window functions. Proc. VLDB Endow. **5**(11), 1244–1255 (2012)
5. Chernishev, G.A., Galaktionov, V.A., Grigorev, V.D., Klyuchikov, E.S., Smirnov, K.K.: PosDB: an architecture overview. Programm. Comput. Softw. **44**(1), 62–74 (2018)
6. Chernishev, G., Galaktionov, V., Grigorev, V., Klyuchikov, E., Smirnov, K.: PosDB: a distributed column-store engine. In: Petrenko, A.K., Voronkov, A. (eds.) PSI 2017. LNCS, vol. 10742, pp. 88–94. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74313-4_7
7. Coelho, F., Pereira, J., Vilaça, R., Oliveira, R.: Holistic shuffler for the parallel processing of SQL window functions. In: Jelasity, M., Kalyvianaki, E. (eds.) DAIS 2016. LNCS, vol. 9687, pp. 75–81. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39577-7_6
8. Graefe, G.: Query evaluation techniques for large databases. ACM Comput. Surv. **25**(2), 73–169 (1993)
9. Harizopoulos, S., Abadi, D., Boncz, P.: Column-Oriented Database Systems, VLDB 2009 Tutorial **2**(2), 1664–1665 (2009). http://nms.csail.mit.edu/~stavros/pubs/tutorial2009-column_stores.pdf
10. Jacobson, N.: Semi-Groups and Groups, pp. 15–48. Springer, New York (1951). https://doi.org/10.1007/978-1-4684-7301-8_2

11. Leis, V., Kundhikanjana, K., Kemper, A., Neumann, T.: Efficient processing of window functions in analytical SQL queries. Proc. VLDB Endow. **8**(10), 1058–1069 (2015)
12. O'Neil, P., Chen, X.: Star Schema Benchmark, June 2009. http://www.cs.umb.edu/~poneil/StarSchemaB.PDF
13. Wesley, R., Xu, F.: Incremental computation of common windowed holistic aggregates. Proc. VLDB Endow. **9**(12), 1221–1232 (2016)
14. Zuzarte, C., Pirahesh, H., Ma, W., Cheng, Q., Liu, L., Wong, K.: Winmagic: subquery elimination using window aggregation. In: SIGMOD 2003, pp. 652–656. ACM, New York (2003)

Applications of Model and Data Engineering

A Machine Learning Model for Automation of Ligament Injury Detection Process

Cheikh Salmi[1](✉), Akram Lebcir[2], Ali Menaouer Djemmal[2], Abdelhamid Lebcir[3], and Nasserdine Boubendir[3]

[1] LIMOSE, University of M'Hamed Bougarra, Boumerdes, Algeria
c.salmi@boumerdes-univ.dz
[2] University of M'Hamed Bougarra, Boumerdes, Algeria
[3] HCA Kouba, Algiers, Algeria

Abstract. Good exploitation of medical data is very useful for patient assessment. It requires a diversity of skills and expertise since it concerns a large number of issues. Traumatic pathology is by far the most frequent problem among young athletes. Sport injuries represent a large part of these accidents, and those of the knee are the most important, dominated by meniscal and ligamentous lesions including that of the anterior cruciate ligament ($\mathcal{ACL}$). Magnetic Resonance Imaging ($\mathcal{MRI}$) is the reference for knee exploration, the number of knee MRI exams is in a perpetual increase thus of its contribution in the patient assessment and $\mathcal{MRI}$ machines availability. Therefore, radiologist's time has become a limiting factor because of the large number of images to examine, in addition to the possibility of error in the interpretation. The possibility of automating certain interpretation functions is currently possible in order to limit the amount of errors and inter-observer variability. Deep learning is useful for disease detection in clinical radiology because it maximizes the diagnostic performance and reduces subjectivity and errors due to distraction, the complexity of the case, the misapplication of rules, or lack of knowledge. The purpose of this work is to generate a model that can extract $\mathcal{ACL}$ from $\mathcal{MRI}$ input data and classify its different lesions. We developed two convolutional neural networks ($\mathcal{CNN}$) for a dual-purpose, the first is to isolate the $\mathcal{ACL}$ and the second to classify it according to the presence or absence of lesions. We investigate the possibility of automating the $\mathcal{ACL}$ tears diagnostic process by analyzing the data provided by cross sections of patient $\mathcal{MRI}$ images. The analysis and experiments based on real $\mathcal{MRI}$ data show that our approach substantially outperforms the existing deep learning models such as support vector machine and Random Forest Model, in terms of injury detection accuracy. Our model achieved an accuracy rate equal to 97.76%.

Keywords: Machine learning · Deep learning · Neural network · MRI · ACL

K.-D. Schewe and N. K. Singh (Eds.): MEDI 2019, LNCS 11815, pp. 317–332, 2019.
https://doi.org/10.1007/978-3-030-32065-2_22

1 Introduction

The management of common diseases is complex but represents a major public health issue. Computer-aided diagnostic ($\mathcal{CAD}$) are interactive computer systems designed to assist physicians or other health professionals in choosing between certain relationships or variables in order to make a diagnostic or therapeutic decision. These systems were born in the 70's through the famous MYCIN [1] expert system whose purpose was to help physicians to perform the diagnosis and care of blood infectious diseases. More efficient systems can be developed by exploiting the advances made on both technology and computer science techniques such as $\mathcal{MRI}$, molecular imaging, 4D ultrasound, flat-plate detectors, liquid metal bearings, $\mathcal{PACS}$ picture archiving and communication systems), $\mathcal{RIS}$ (radiology information systems), computing infrastructure (redundant server technology), data mining, image segmentation algorithms and artificial intelligence. By efficient systems, we mean automated system that can help to reduce the analysis time and human errors which will increase the diagnostics precision.

Machine learning ($\mathcal{ML}$) is a research field in full expansion and promised a great future. Its applications, which concern all human activities, make it possible to improve healthcare. $\mathcal{ML}$ is indeed at the heart of the medicine of the future, with robot-assisted surgery, personalized treatment and smart prostheses, etc. The alliance between $\mathcal{ML}$ and Big Data can lead to the development of smart $\mathcal{CAD}$s.

The anterior cruciate ligament ($\mathcal{ACL}$) provides stability in anterior translation and rotation. It contributes to the stability of the human knee and provides meniscus protection. Because of its frequent solicitation, the $\mathcal{ACL}$ is often subject to tear in particular among athletes practicing football, skiing, handball, judo and basketball. Typically, patients with anterior cruciate ligament (see Fig. 1) tear experience instability and knee discomfort, particularly in activities involving changes of direction or pivots on the affected leg. About 50% of anterior cruciate ligament lesions are accompanied by meniscus injury, cartilage, or other ligament injury. In such case, a surgery is necessary for the purpose of reconstructing the ligament at its exact location [2,3]. Radiologist evaluates whether the ligament is torn or not by performing different clinical tests and performing an $\mathcal{MRI}$. This examination determines also other associated meniscal and articular cartilage lesions.

The goal of this research is to build an intelligent framework to automate the ligament injury detection process based on $\mathcal{MRI}$ images data sets. The data set images will be classified into two sub-classes: injured and non-injured. Deep learning techniques are used to perform both the detection and classification tasks, as convolutional neural networks have proven to be an effective approach in complex image classification and segmentation problems in general.

The paper is organised as follows. In Sect. 2, we present a brief overview of the neuronal network. Our proposal is detailed in Sect. 3. It includes the description of the main used layers, the training process before presenting the final optimal

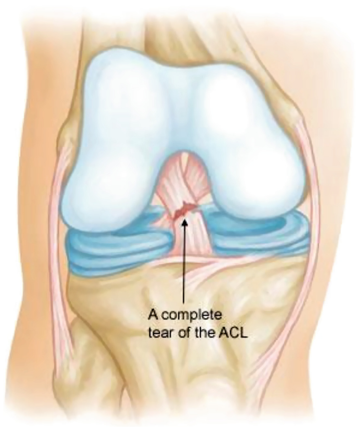


Fig. 1. Anterior cruciate ligament injury. (https://orthoinfo.aaos.org/en/diseases--conditions/anterior-cruciate-ligament-acl-injuries/)

model. Next, in Sect. 4, experimental results and discutions are presented. The last section emphasizes the conclusion and perspectives for future work.

2 Artificial Neural Network

Artificial Neural Networks ($\mathcal{ANN}$) [4] are inspired by the human brain and its central nervous system. It consists of an abstracted model of interconnected neurons called units or nodes; whose special arrangement and linking can be used to solve computer-based problems (e.g., image, voice and pattern recognition, control and simulation of complex processes, economic models, biometric systems, etc.). As shown in Fig. 2, an $\mathcal{ANN}$ is a directed network where the information travels in one direction from the input layers and modulated forward to other neurons or output as a final result. Basically, a distinction can be made between input neurons, hidden neurons and output neurons. The input neurons receive information in the form of patterns or signals from the outside world. The hidden neurons are located between the input and output neurons and map internal information patterns. Output neurons relay information and signals to the outside world as a result. The different neurons are connected to each other via the so-called edges. Thus, the output of one neuron can become the input of the next neuron. Depending on the strength and meaning of the connection, the edge has a certain weighting which play an important role in controling the operation of the network: The stronger the weighting, the greater the influence a neuron can exert on the connection to another neuron. Convolutional networks ($\mathcal{CNN}$s) are a particular form of ***multilayer neural network*** that is well suited to image processing. It is today, the most powerful class of algorithms for classifying images. There are four types of layers five a convolutional neural network: convolutional layer, pooling layer, correction layer, loss layer and the fully-connected layer (for more details see [5]). The first convolutional layers are based on the mathematical principle of convolution, and seek to identify the presence of a motif in an image (the coutour for example). Other layers that are not necessarily convolutional can be used to classify the different types of images.

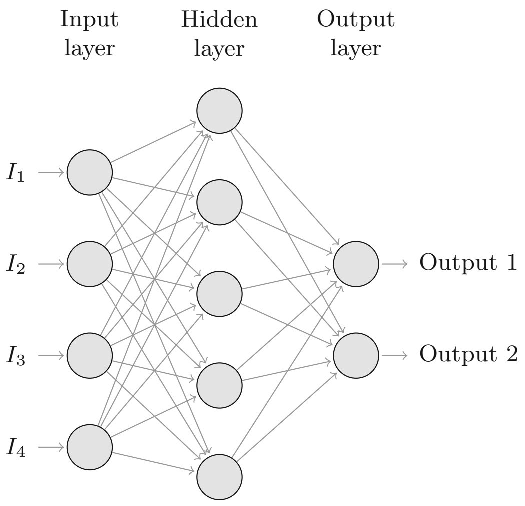


Fig. 2. Basic neural network architecture.

3 Convolutional Neural Network for ACL

In the following section we present our convolutional neuronal network model to detect anterior cruciate ligament on knee $\mathcal{MRI}$ exam. It is composed of two $\mathcal{CNN}$s that perform selection and classification tasks. The classification task allows the distinction between normal and pathological (total or partial lesion) $\mathcal{ACL}$s, while the selection task locates the image section that contains the $\mathcal{ACL}$ (see Figs. 3 and 4), a size image $100 \times 100 \times 32$ (height $\times$ width $\times$ depth) from native $320 \times 320 \times 32$ volumes.

In general, the neuron number in each hidden layer can vary according to the complexity of the problem and the dataset. Hence, $\mathcal{CNN}$ architectures vary in the number and type of layers depending on its application. For instance, the network should contain a fully connected regression layer at its end for continuous answers, whereas for decisional answers it must include a fully connected classification layer. However, there are no pre-established rules for selecting an optimal $\mathcal{CNN}$ configuration. Thus, an empirical result driven approach (trial and error) is used to select the optimal $\mathcal{CNN}$ global architecture (depth, breadth, activation

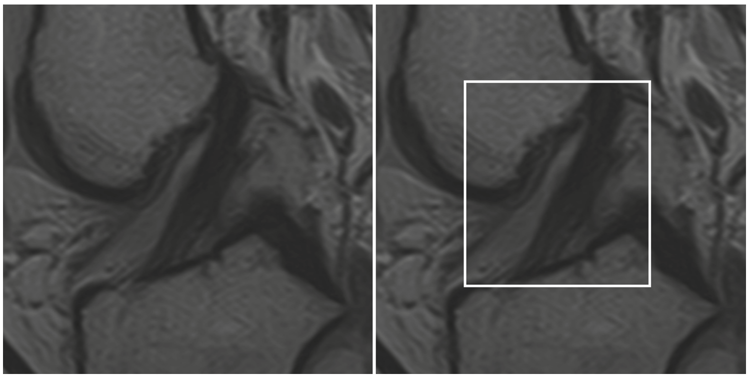


Fig. 3. ACL region selection

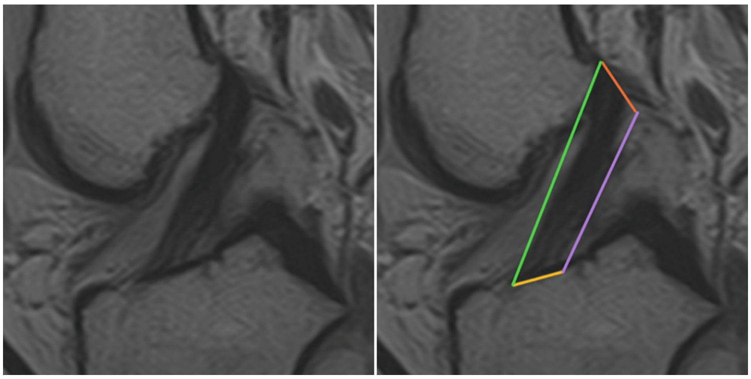


Fig. 4. ACL location

functions, connections, optimization algorithm and its parameters and loss function for a specific dataset). The main idea of our network architecture design is to minimize the number of parameters and maximize its learning ability. To this end, several techniques are used: convolution process to extract image features, non-linear activation (e.g., Relu) functions are used to deal with complexity varying parameters of image input data, pooling layers to handle the sensitivity of the output feature maps to the location of the features in the input, cross-entropy for performance measurement, softmax for classification and gradient decent combined with efficient optimization algorithms such as ADAM [6] are used for backpropagation. A basic convolutional neural network composed of 8 convolutional layers, a layer of max pooling placed after each series of 2 convolutive layers and finally a 3-layer $\mathcal{MLP}$s (Multilayer Perceptron) is learned. The network has 1195074 learnable parameters. For faster learning, ReLu [7] is used as an activation function. This basic $\mathcal{CNN}$ is enhanced by adding other hidden layers. In each layer, activation volumes are altered with the use of differentiable functions.

3.1 Loss Layer

This layer specifies some measure of error between computed outputs and the desired target outputs of the training data. Many research results suggest using different measure such as mean squared and cross entropy errors. We use categorical cross-entropy loss which is a softmax activation plus a cross-entropy loss. The softmax function calculates the probabilities of each class and the target class will have the highest probability. The Cross-Entropy cost function represents the sum of the separate loss for each class label per observation. Hence, the softmax and the cross-entropy functions can be written as:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{1}$$

$$CE = -\sum_{i=1}^{C} t_{o,c} log(p_{o,c}) \tag{2}$$

Where C is the number of classes (total injury, partial injury, no injury), s_j are the scores inferred by the net for each class $c_j \in C$, t_i is a one-hot vector meaning that the class label c is the correct classification for observation o, p is the result of the softmax function which denotes the predicted probability that observation o is of class c.

3.2 Learning Optimization

Many techniques exist to optimize the learning process speed and its prediction accuracy. This can be done by optimizing a different objective function $J(\theta)$. Gradient descent ($\mathcal{GD}$) based approaches update the parameters θ in the opposite direction of the objective function gradient. The learning rate η determines the importance of the step that will be taken to reach the local. This strategy can be described by the following equation:

$$\theta_j = \theta_j - \eta \nabla_\theta J(\theta) \tag{3}$$

An other variant of the gradient descent is the stochastic gradient descent ($\mathcal{SGD}$)[8]. As in $\mathcal{GD}$, this approach updates a set of parameters in an iterative manner to minimize an error function using only a sample of examples to correct the parameters. Hence, $\mathcal{SGD}$ converges much faster but the error function is not as well minimized as in the case of $\mathcal{GD}$. However, in all $\mathcal{GD}$ variants, choosing a proper learning rate η can be difficult. Many approaches called adaptative algorithms were proposed to improve the gradient descent by adjusting the learning rate during training. In this work, we use ADAM (Adaptive Moment Estimation) which is essentially a combination of the gradient descent with momentum [9] and RMSProp algorithms (Root Mean Square Prop is an unpublished, adaptive learning rate method proposed by Geoff Hinton). ADAM uses previous historical gradients (values calculated in past times $< t$). It uses the decaying averages of past and past squared gradients:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{4}$$

where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively. During the initialization steps the momentums are biased to zero ($m_0 = 0$ and $v_0 = 0$). Hence, bias-corrected momemnts became:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{5}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{6}$$

as recommended in [6], we used in our implementation, default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. Several other techniques have been used to improve the architecture of our model such as dropout [10] as a regularization technique to avoid overfitting due to the presence of many fully connected layers.

3.3 Network Architecture

The $\mathcal{ACL}$ injury detection is a complex problem. It is divided into subparts, and for each part, a cluster of neurons is created to study that specific portion. Hence, the network structure consists of two cascaded $\mathcal{CNN}$s layers: localization and classification (see Fig. 5).

The first $\mathcal{CNN}$ isolates the $\mathcal{ACL}$, it allows to predict a rectangular box of (100 × 100) coordinates centered on the inter-condylar notch, from the native images of (320 × 320) after their downsizing to (120 × 120) for CNN entry.

This $\mathcal{CNN}$ contains:

- 5 convolution-layers each one contains (conv2D, ReLU, Maxpool)
- A layer of global Maximum pooling
- Two fully connected layers: containing respectively 4096 and 2048 nodes each layer contains an output of 4 nodes (for the 4 coordinates) with a Dropout function inter-posed.

Based on the images produced by the first localisation $\mathcal{CNN}$, the second $\mathcal{CNN}$ allows to decide if $\mathcal{ACL}$ is pathological. This $\mathcal{CNN}$ contains:

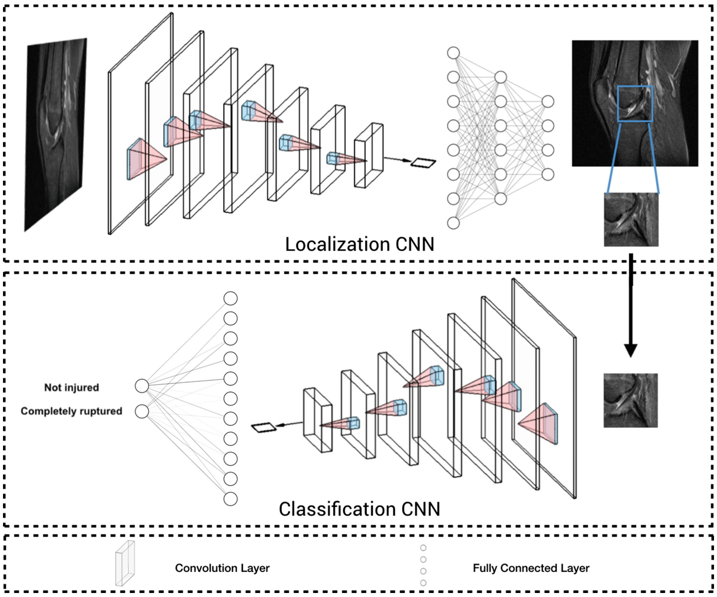


Fig. 5. Architecture model

- 8 convolution layers each contains (conv2D, BN, eLU) and Maxpool
- A layer of global Avgpooling
- A layer of neurons containing 96 nodes with an output of 2 nodes with a sofmax to calculate the probability according to the mapping of crossing of the connections of the input.

The detailed architecture of our network, the stack of layers and the different parameters of the two $\mathcal{CNN}$s are presented in the Table 1. The two $\mathcal{CNN}$s models are respectively:

- Input → Conv2D → Relu → Maxpool → Conv2D → Relu → Maxpool → Conv2D → Relu → Maxpool → Conv2D → Relu → Maxpool → Conv2D → Relu → GlobalMaxpool → Dropout → FC → Dropout → FC → FC
- Input → Conv2D → BN → elu → Maxpool → Conv2D → BN → elu → Maxpool → Conv2D → BN → elu → Maxpool → Conv2D → BN →elu → Maxpool → Conv2D → BN → elu → Maxpool → Conv2D → BN → elu → Maxpool → Conv2D → BN → elu → Maxpool → Conv2D → BN → elu → Maxpool→ GlobAvgPool → Flatten → FC → FC → Softmax

In Table 2, we give a brief description of the layers used in the proposed network architecture. For more details see [5,11].

Table 1. Detailed location and classification $\mathcal{CNN}$s

Localisation network	Classification network
Input (120 120 3 grayscale image)	Input (100 100 3 grayscale image)
Conv2D (32 3 × 3 filters), ReLU	Conv2D (32 2 × 2 filters), BN, Elu
Maxpool (window size 2 × 2)	Maxpool (window size 2 × 2, stripe size 2)
Conv2D (128 3 × 3 filters), ReLU	Conv2D (32 3 × 3 filters), BN, Elu
Maxpool (window size 2 × 2)	Conv2D (64 5 × 5 filters), BN, Elu
Conv2D (256 3 × 3 filters), ReLU	Maxpool (window size 3 × 3, stripe size 2)
Maxpool (window size 2 × 2)	Conv2D (64 3 × 3 filters), BN, Elu
Conv2D (512 3 × 3 filters), ReLU	Conv2D (64 3 × 3 filters), BN, Elu
Maxpool (window size 2 × 2)	Maxpool (window size 2 × 2, stripe size 2)
Conv2D (1024 3 × 3 filters), ReLU	Conv2D (96 3 × 3 filters), BN, Elu
GlobalMaxpool (window size 2 × 2)	Conv2D (96 3 × 3 filters), BN, Elu
Dropout	Maxpool (window size 2 × 2, stripe size 2)
FC (4096 nodes), ReLU	GlobAvgPool
Dropout	Flatten
FC (2048 nodes), ReLU	FC (96 nodes)
FC (4 nodes), ReLU	FC (2 nodes)
	Softmax (2 classes)

Table 2. Layers descriptions

Layer	Description
Input	Input layer
Conv2D	Two-dimensional convolution
%pool	Layer for image sub-sampling (compression)
ReLU	Rectified-linear unit
Dropout	Regularization layer to avoid the problem of over-learning
FC	Fully connected layer
Softmax	Multinomial logistic regression layer
Flatten	Data adaptation for inputting it to the next layers.
BN	Batch normalization layer
Elu	Exponential linear unit

4 Experimental Results and Discussion

4.1 Dataset

In the present work, the dataset is based on 12-bit grayscale volumes of either left or right knees. This dataset has been defined in [2] and used in many other research papers such as [12]. It consists of 917 exams, with sagittal T1-weighted series and labels for $\mathcal{ACL}$ injury from Clinical Hospital Centre Rijeka, Croatia. The dataset labels are extracted for 3 levels of $\mathcal{ACL}$ lesions defined as: non-injured (690 exams), partially injured (172 exams) and completely torne (55 exams). Image pre-processing is a crucial step because it makes it possible to standardize the input data and the treatment sequence. An $\mathcal{MRI}$ exam is a 320 × 320 × 32 (height × width × depth) image, each one contains 32 sagittal images. The area encompassing the $\mathcal{ACL}$ is an image portion with a size of (100 × 100 × 32). When importing the labeled dataset, the training and validation subsets are made up respectively of 80% and 20% of the entire collected dataset using stratified random sampling. This means that 80% of the data is used to train the network, and 20% of the data is used in the validation phase. After reaching a consistent validation accuracy, a 20% additional dataset is used to test the performance of the network.

4.2 Evaluation

In order to evaluate the performance of our $\mathcal{CNN}$ model we have implemented it on TensorFlow which is an open source framework developed by Google. First, to understand how well training is currently proceeding, training and validation accuracy and loss are plotted over training epochs. As shown respectively in Figs. 6 and 7 we can easily see that the accuracy of training and validation increases with the number of epochs, this reflects that in each epoch the model learns more information. Similarly, the learning and validation errors decrease with the number of epochs.

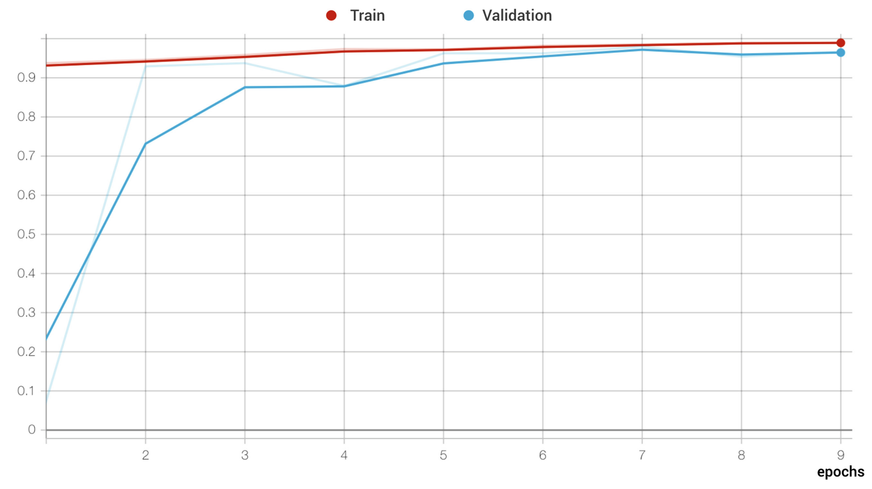


Fig. 6. Model precision

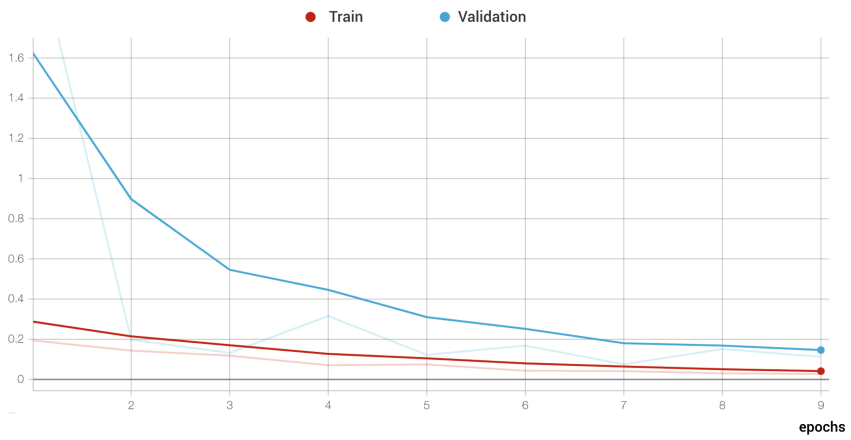


Fig. 7. Model error

We compared then, our model with 2 well known models for image recognition: Inspection-V3 [13] and Mobile-Net [14] and 2 other models in the state of the art for detection of cruciate ligament injury: MRNet [12] and tajduhar et al. [2]. The percentage of classification P is the ratio between the recognized images and the total number of images. To evaluate the performance of the models compared in the present study, three indices are adopted: sensibility, specificity and $\mathcal{AUC}$ (Area under the curve). They are listed below:

$$\begin{aligned} Sensibility &= TP/(TP + FN) \\ Specificity &= TN/(TN + FP) \end{aligned} \tag{7}$$

Where TP, TN, FP and FN denote respectively True Positive, True Negative, False Positive (type 1 error) and False Negative (type 2 Error). Recall that predicted values are described as Positive and Negative and actual values are described as True and False. The $\mathcal{AUC}$ is a measure of separability. It represents the probability that a classifier will rank a randomly chosen positive instance as positive and negative as negative. More formally:

$$\begin{aligned} AUC &= \sum_i S_i \\ S_i &= (FPR_i - FPR_{i-1}) \times \frac{TPR_i - TPR_{i-1}}{2} \end{aligned} \tag{8}$$

Where TPR is the Sensibility (true positif rate) and FPR is the false positif rate (equal to: 1 - Specificity). Estimations variability is assessed using a 95% Wilson score confidence interval (CI) [15] for all metrics (sensitivity, specificity and AUC). This interval is defined by a lower and an upper bounds which are respectively $(P - \alpha \times s_p)$ and $(P + \alpha \times s_p)$ where:

$$s_p = \sqrt{\frac{P(1-P)}{T}} \tag{9}$$

P is the percentage of classification, T is the sample size and α is an error risk (5% in our settings).

Table 3 compares the sensitivity, specificity and $\mathcal{AUC}$ of our model against Mobile-Net and Inspection-V3 to determine the presence or absence of an $\mathcal{ACL}$ tear according to the dataset taken as reference for the $\mathcal{ACL}$ tear classification system. Our model performed well with estimated sensitivity, specificity and $\mathcal{AUC}$ at the optimal threshold of the Youden index [16]. It has the highest overall diagnostic performance compared to the other models. Table 3 depicts the detailed values for each metric.

Table 3. Models sensitivity, specificity and AUC

Model	Sensitivity	Specificity	AUC
Our model	99.31 (95% CI: 98.6; 1)	93.8 (95% CI: 91.7; 95.9)	96.6 (95% CI: 95; 98.2)
Mobile-Net	92.29 (95% CI: 89.94; 94.64)	92.61 (95% CI: 90.31; 94.91)	92.5 (95% CI: 90.18; 94.82)
Inspection-V3	92.5 (95% CI: 90,18; 94, 82)	92.72 (95% CI: 89.78; 94.46)	92.6 (95% CI: 89.95; 94.65)

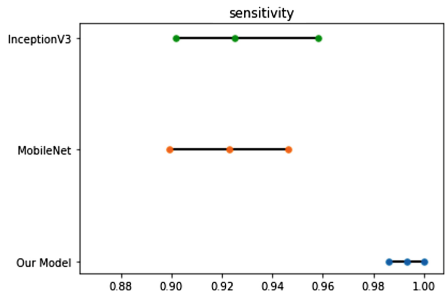


Fig. 8. Sensitivity

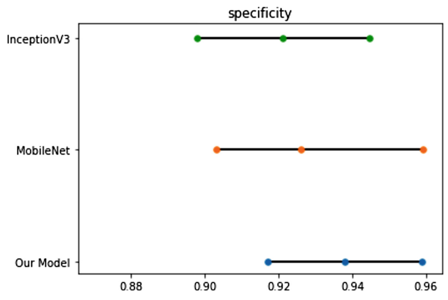


Fig. 9. Specificity

Figures 8, 9 and 10 show respectively a graphical comparison of the sensitivity, specificity and $\mathcal{AUC}$ for different models. To show the different models predective performance under various discriminative thresholds, we use instead $\mathcal{ROC}$ curve (receiver operating characteristics) which represents the plotting of TPR against FPR.

Figure 11 show the $\mathcal{ROC}$ curve giving the rate of successful classification of all the compared models used in $\mathcal{ACL}$ tear classification. According to the graphics, it is clear that the $\mathcal{AUC}$ for our model $\mathcal{ROC}$ curve is higher than other models.

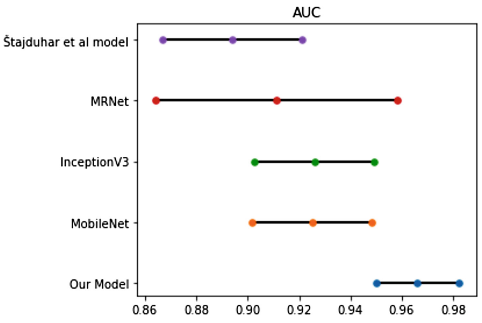


Fig. 10. Area under the curve

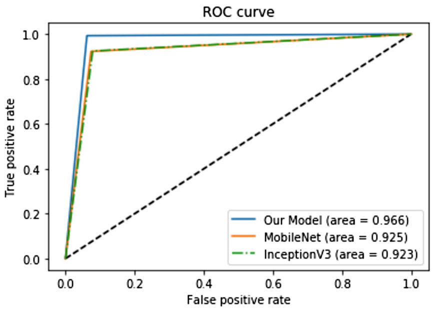


Fig. 11. Area under the curve-ROC

4.3 Discussion

In this study we compared four $\mathcal{CNN}$s, our model, and three other reference models, InceptionV3 and MobileNet. Our model provides the best diagnostic performance to classify $\mathcal{ACL}$ tears because it allows simple communication inside the network compared to the two others (MobileNet and InceptionV3). Information is propagated directely across the different layers of the network and thus reduces the number of the necessary learning parameters in spite of the small used dataset. MobileNet and InceptionV3 have a complex network structure (4.3, 23.8) million parameters; therefore, it is necessary to have a large dataset to obtain optimal performance, probably because of low diagnostic performance. The InceptionV3 model is superior to MobileNet because of the extended blocks of filters and concatenation layers. As a result, it is likely to provide better performance in classifying $\mathcal{ACL}$ tears.

5 Related Work

Artificial intelligence has been used in the $\mathcal{CAD}$s as a technique of knowledge organization and human reasoning parallelization [17]. In [18], Kulikowski et al. have used pattern-matching methods in work on a medical diagnostic task. In [19], a rule-based physiological model has been develpped as the basis for diagnostic consultations of glaucoma, which is represented as linguistic structures and explicitly displaying reasoning procedures. In [20], a fuzzy-based rule system for detecting early stage glaucoma, especially for large-scale screening is proposed.

The literature review confirmed that, there has been growing interest from academics and practitioners on machine learning as a new paradigm for $\mathcal{CAD}$s to address the diagnostic process of a wide range of deseases. This new orientation is motivated by (1) the developpement of $\mathcal{MRI}$ which is a medical imaging technique that provides two or three-dimensional views of the body's interior without radiation with relatively high contrast resolution. $\mathcal{MRI}$ provides information on lesions that are not visible on plain X-rays, ultrasound or CT Scanner

(Computed Tomography). The proliferation of $\mathcal{MR}$ units over medical centers, provides more highly detailed images which increase the ability to diagnose and to treat through the guidance of therapeutic acts by imaging but also to exploit this data images in research area (2) the developpment of good evaluation measures for image segmentation performance [21].

There already exists scientific research which aim at automating the detection of cruciate ligament injury, some of which are presented here. A semi-automated approach to detect the presence of anterior cruciate ligament injury in a human knee was proposed in [2]. Two machine-learning models were used namely, support vector machine ($\mathcal{SVM}$) and random forests model. The image area to be analyzed were selected manually and used to generate histogram of oriented gradient ($\mathcal{HOG}$) and gist descriptors. The best results obtained were 89.4% for the injury-detection problem and 94.3% for the complete-rupture-detection problem using a linear-kernel $\mathcal{SVM}$ learned from $\mathcal{HOG}$ descriptors.

In [22], Fang Liu et al. developped a model for the evaluation of cartilage lesions on knee MRI images using deep learning algorithms, which introduced a double convolutional network ($\mathcal{CNN}$, see Sects. 2 and 3) for segmentation and classification of cartilages damages. In [12], Nicholas Bien et al. developped a deep learning model for detecting anterior cruciate ligament and meniscal tears on knee $\mathcal{MRI}$ exams. The model is based on a convolutional neural network called ($\mathcal{CNN}$) called MRNet. They introduced 3D concept and used a combination of multi-planar series predictions using logistic regression. Therefore, there is a demand for the development of a new approach to automate the detection process. For this purpose, in an original way, a $\mathcal{CNN}$ approach is proposed to detect anterior cruciate ligament lesion with supervised learning.

6 Conclusion

This paper presents an application of convolutional neural networks model for the classification and detection of knee lesion based on $\mathcal{MRI}$images. Two $\mathcal{CNN}$s with different architectures are used respectively for the localisation of the $\mathcal{ACL}$ and the prediction of $\mathcal{ACL}$ injury (classification). The selection of the best model is done by trial and error approach. Six intermediate models were implemented and tested. Implementation and tests show that the perforamnce and the results accuracy depends on many parameters such as the used layers, the network depth and the number of epoch. These models results comparison are not shown due to the lack of space. We presented the different layers of both localisation and classification $\mathcal{CNN}$s: convolutional layers, rectification layers, the pooling layers and the fully connected layers. The overfitting problem is takled using dropout layers as a regularization technique. Implementation is done using TensorFlow framework. Three hardware platforms are used to implement our networks trainings: CPU, GPU and cloud computation. Results show that GPU and cloud computation improve considerably the scalability and decrease the training time. We compared our model with several reference models such as InceptionV3 and MobileNet. These two models are complex (4.3; 23.8) million parameters and therefore require to have a large dataset to get more optimal performance. Our model provides the best diagnostic performance for the $\mathcal{ACL}$ classification because it allows a simple communication compared to the latter. Our model allows a direct information propagation between the different layers of the network and thus reduces the number of parameters necessary for its learning phase despite the small size of the used dataset. As future work, the aim would be to:

– Improve the model (more performance),
– Add other structures to examine (tendons, muscles, etc.),
– Create 3D models,
– Implement dynamic models (auto-improvement during use).

References

1. Shortliffe, E.: Consultation systems for physicians: the role of artificial intelligence techniques, pp. 511–527, January 1980
2. Tajduhar, I., Mamula, M., Mileti, D., Ünal, G.: Semi-automated detection of anterior cruciate ligament injury from MRI. Comput. Methods Prog. Biomed. **140**(C), 151–164 (2017)
3. Gottlob, C., Baker, C., Pellissier, J., Colvin, L.: Cost effectiveness of anterior cruciate ligament reconstruction in young adults. Clin. Orthop. Relat. Res. (367), 272–282 (1999). https://www.ncbi.nlm.nih.gov/pubmed/10546625
4. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943)
5. Aghdam, H.H., Heravi, E.J.: Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification, 1st edn. Springer, Heidelberg (2017)
6. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
7. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 807–814 (2010)
8. Mandt, S., Hoffman, M.D., Blei, D.M.: A variational analysis of stochastic gradient algorithms. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML 2016, pp. 354–363 (2016). JMLR.org
9. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Netw. **12**(1), 145–151 (1999)
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
11. Venkatesan, R., Li, B.: Convolutional Neural Networks in Visual Computing: A Concise Guide. Data-Enabled Engineering. CRC Press, Boca Raton (2018)
12. Bien, N., et al.: Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of mrnet. PLoS Med. **15**(11), e1002699 (2018)

13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR abs/1512.00567 (2015)
14. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Inverted residuals and linear bottlenecks: mobile networks for classification, detection and segmentation. CoRR abs/1801.04381 (2018)
15. Wilson, E.B.: Probable inference, the law of succession, and statistical inference. J. Am. Stat. Assoc. **22**(158), 209–212 (1927)
16. Youden, W.: Index for rating diagnostic tests. Cancer **3**(1), 32–35 (1950)
17. Shortliffe, E.H., Axline, S.G., Buchanan, B.G., Merigan, T.C., Cohen, S.N.: An artificial intelligence program to advise physicians regarding antimicrobial therapy. Comput. Biomed. Res. **6**(6), 544–560 (1973)
18. Kulikowski, C.: Pattern recognition approach to medical diagnosis. IEEE Trans. Syst. Sci. Cybern. **SSC–6**, 173–178 (1970)
19. Weiss, S.M., Kulikowski, C.A., Amarel, S., Safir, A.: A model-based method for computer-aided medical decision-making. Artif. Intell. **11**(1), 145–172 (1978). Applications to the Sciences and Medicine
20. Song, X., Song, K., Chen, Y.: A computer-based diagnosis system for early glaucoma screening. In: 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, pp. 6608–6611, January 2005
21. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vision **42**(3), 145–175 (2001)
22. Liu, F., Zhou, Z., Samsonov, A., Blankenbaker, D., Larison, W., Kanarek, A., Lian, K., Kambhampati, S., Kijowski, R.: Deep learning approach for evaluating knee mr images: achieving high diagnostic performance for cartilage lesion detection. Radiology **289**(1), 160–169 (2018)

Robust Design of a Collaborative Platform for Model-Based System Engineering: Experience from an Industrial Deployment

Christophe Ponsard[1](✉), Robert Darimont[2], and Mounir Touzani[3]

[1] CETIC Research Center, Charleroi, Belgium
christophe.ponsard@cetic.be

[2] Respect-IT SA, Louvain-la-Neuve, Belgium
robert.darimont@respect-it.be

[3] Toulouse, France

Abstract. Model-Based System Engineering is gaining momentum in the industry. In order to be successful, it requires adequate tooling support. In addition to functional requirements related to model edition, verification and transformation, key non-functional requirements need to be carefully addressed such as versioning, usability/team work, reliability, security, ease of integration. In this paper, we first give an overview of how we dealt with such requirements in the context of the development of a real world platform for a global telecom operator, with a focus on early steps of system modelling. We then present a more detailed design of the tooling architecture and a high availability protocol for accessing a mainstream model repository. The proposed protocol is modelled and verified using the Alloy language and model-checker.

Keywords: Model-Based System Engineering · Tool support · Modelling · Industrial transfer · High availability · Alloy · Model-checking

1 Introduction

Modelling has been used for a long time across many engineering disciplines like civil engineering, electronic systems and aeronautics. It is now increasingly applied at system level across disciplines through Model-Based System Engineering (MBSE) with the aim to rely primarily on domain models to support the exchange between engineers rather than documents. Model-Driven Engineering (MDE) is a similar trend focusing only the software development process [30]. Such approaches can rely on standardised and well adopted modelling languages like SysML [22] at system level, UML [21] for software and increasingly Domain

M. Touzani—Independent Researcher.

Specific Languages (DSLs). They provide a visual syntax enabling the design and communication activities of the engineers but also have precise semantics to enable automation of parts of the System Development Life Cycle (SDLC).

Efficient modelling across the different engineering activities can only be achieved based on reliable computer tools typically composed of a modelling environment, a model repository and a model transformation toolchain for synchronising modelling artefacts at different steps of the SDLC. Model-to-model and model-to-text transformations are respectively used to generate detailed models from abstract ones and code/documentation from models.

Designing a robust toolchain at industrial level is not an easy task. In addition to functional requirements (FR) relating to various model manipulations (edition, check, transformation, simulation,...), it is also very important to cover several non-functional requirements (NFR), in order to ensure industrial strength. Frequently cited NFR are usability, support for collaboration and versioning, scalability, highly availability, integrity, confidentiality, interoperability and maintainability [27,29,31].

In this paper, we present an industrial feedback to cope with such non-functional requirements by elaborating a MBSE platform for a global telecom operator (Huawei Ltd). Our tooling is focusing on the early steps of system development through a goal-oriented approach relying on elaborated requirements modelling. The contribution of this paper is twofold:

- First, we give a high level view about how we addressed important NFR without focusing too much on the specifics of our industrial case but rather by trying to provide adequate feedback that can be applied in a wider context.
- Second, we focus on robust operation requirements, i.e. high availability and load balancing, by describing a generic architecture composed of several redundant server nodes able to process multiple requests and reconfigure in case of node failure. This protocol also involves a master node with specific responsibilities, which must be reassigned to another node in case of failure.

The reported case was carried out over two years, with the last six months mainly devoted to enforcing the robustness of the platform. It is more extensively detailed from a requirements engineering perspective in [25].

This paper is structured as follows: Sect. 2 presents our industrial case and analyses its key requirements. Then Sect. 3 elaborates on how we dealt with non-functional requirements with a generalisation effort. Section 4 goes into details about the specific high availability requirement. It presents a multi-server architecture and a specific protocol ensuring robust operation in presence of node failures. It is modelled and verified using the Alloy language and analyser. Section 5 discusses some related work. Finally, Sect. 6 draws some conclusions and presents our future work.

2 Presentation of Our Industrial Case

This section gives a summary of the context and main requirements of the developed platform. We try to step away from too specific aspects of the industrial case that initiate the work in order to provide a more general feedback. Another reason is that the core of the resulting platform already proved applicable in other domains. An extended description of the Huawei deployment is available [25] and a demonstration version accessible at: http://demo.objectiver.cetic.be/objectiver/client.

2.1 Context and Objectives

The context of our case is quite common to many industries with requirements engineering practices mainly based on domain modelling, analysis of the current solution (i.e. existing product), use case analysis and UML/SysML modelling. The global process is still strongly document-based with different types of documents flowing across the lifecycle. Domain specific languages were already present, mainly for design and testing phases. For example, Gherkins was used to formalise specifications using "Given-When-Then" structure which can be used for testing [3,36].

A common long term objective of companies is also to evolve towards a wider use of modelling across the SDLC but also across products, i.e. by modelling their product lines and using it for better reuse through more systematic domain engineering. However, this evolution should be progressive and preserve the current flow of documents. The transition can be achieved the efficient production of documents using model-to-text [23]. Later on, some documents could become obsolete when direct model-to-model integration is achieved [13].

2.2 Key Requirements

As we focus on requirements modelling, the starting point was to obtain an adequate meta-model for capturing all the knowledge related to stakeholders goals, system properties, domain properties, assumptions on users, and information to be exchanged. As the meta-models available in standard modelling languages such as UML and SysML are mostly poor with this respect, a specialised meta-model was selected: KAOS [7,14], among other candidates like i* [37] or URN [5]. In addition to concepts, different sources and targets of the model transformations were also modelled, like diagrams or documents. Possible transformations between those artefacts were also identified and are documented in [26]. For example, requirements can be tagged in a source document and refined using decomposition inside a diagram, then selected as part of a specific subsystem and exported in a public tender or directly transferred in the development department of the company.

In order to minimise the effort to build a model and to maximise the value from the invested modelling effort, the proposed tooling needs to have the following qualities (or NFR):

- *Scalability*: efficient support for large models but also for several models and multiple concurrent users.

– *High availability*: system up and running with very reduced unplanned interrupt time, meaning service reliability and server redundancy.
– *Navigation across multiple versions* of modelling artefacts for traceability or better collaboration support.
– *Usability* (visual feedback, shortcuts,...) for productivity and adoption.
– *Flexible integration*: to exchange models or expose specific (web-)services.
– *Security* enforcement (model integrity, confidentiality, access control).
– *Long term support/portability*: to ease maintenance over a long time and to enable reuse through a knowledge base or product lines.
– *Reduced installation and maintenance* effort to minimise operation costs.

3 Dealing with Non-functional Requirements

3.1 Global Architecture

Several NFR can be addressed through an adequate tool architecture. Our architecture is depicted in Fig. 1. We give here a short summary why it is convenient. More information is available in [8].

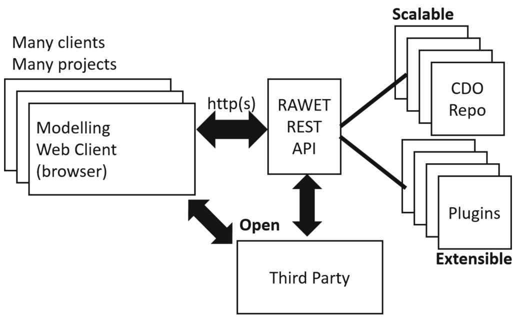


Fig. 1. Global platform architecture

Our tooling architecture is composed of:

– several clients, including a full web-based client, either standalone or embedded in third party tools.
– a RESTful API, called RAWET, providing services for model and diagram edition, history, snapshots, user authentication and project management. It also enables different kinds of integration [28].
– a back-end composed of the model repository relying on a Eclipse Modelling Framework (EMF) store [33] and a collection of plugins enabling both web-services and user interface extensions.

3.2 Scalability and High Availability

Scalability and high availability are crucial for the industrial adoption of an MBSE tooling. This are dealt with at the architecture level an more specifically the model repository which must be able to manage a large number of models, possibly large in size.

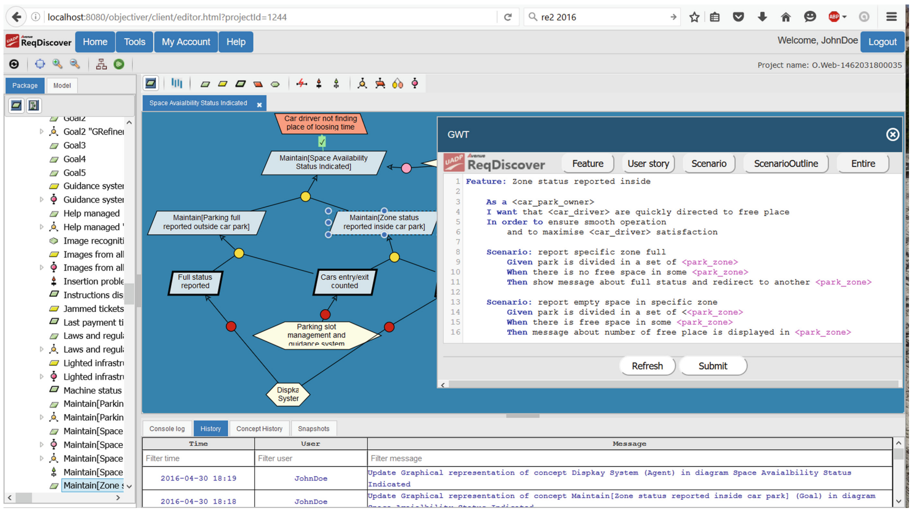
In our case the Eclipse Modelling Framework is used [32]. Different solutions to persist EMF are available and the selected one, Connected Data Object (CDO), offers different possible back-ends, including a mature and scalable relational database manager also with mirroring capabilities.

The server itself is dealing with our RAWET service API. Standard web application technologies can be used to dispatch requests on many concurrent servers and, at the same time, allow some server to be down, thus addressing both scalability and high availability of the service. However, our architecture requires that the model repository access is centralised on a single server which is thus a possible point of failure of the system. In order to cope with this problem, we designed a specific protocol, which is detailed in Sect. 4.

3.3 Ease of Integration

Toolchain integration has started from simple import/export mechanisms and evolved towards a more complex integration with specific tools such as text processors and other SDLC tools. A key decision was to shape the tooling as a series of services available for use over the company intranet. This comes at two different levels:

– *at the user interface level*, the tool provides a similar experience as other modelling tools. However, due to its modular design, web-client extensions


localhost:8080/objectiver/client/editor.html?projectId=1244
re2 2016
ReqDiscover
Home
Tools
My Account
Help
Welcome, JohnDoe
Logout
Project name: O.Web-1462031800035
Package
Model
Space Avaialbility Status Indicated
Car driver not finding place of loosing time
Maintain[Space Availability Status Indicated]
Maintain[Parking full reported outside car park]
Maintain[Zone status reported inside car park]
Full status reported
Cars entry/exit counted
Parking slot management and guidance system
Dispka Syster
GWT
Feature
User story
Scenario
ScenarioOutline
Entire
Refresh
Submit
Console log
History
Concept History
Snapshots

```
Feature: Zone status reported inside

    As a <car_park_owner>
    I want that <car_driver> are quickly directed to free place
    In order to ensure smooth operation
        and to maximise <car_driver> satisfaction

    Scenario: report specific zone full
        Given park is divided in a set of <park_zone>
        When there is no free space in some <park_zone>
        Then show message about full status and redirect to another <park_zone>

    Scenario: report empty space in specific zone
        Given park is divided in a set of <<park_zone>
        When there is free space in some <park_zone>
        Then message about number of free place is displayed in <park_zone>
```

Time	User	Message
Filter time	Filter user	Filter message
2016-04-30 18:19	JohnDoe	Update Graphical representation of concept Dispkay System (Agent) in diagram Space Avaialbility Status Indicated
2016-04-30 18:18	JohnDoe	Update Graphical representation of concept Maintain[Zone status reported inside car park] (Goal) in diagram

Fig. 2. Modular web-based user interface

can easily be embedded. For example, Fig. 2 shows the integration of an editor for the GWT Domain Specific Language. Conversely, specific components can use reused inside other tools, e.g. a dashboard or read-only view.
– *at the model level*, a clean RESTful API is directly available to perform CRUD (Create/Read/Update/Delete) operations both on the model elements and on model representations inside diagrams, baselines, etc. This allows third-party tools to directly push or query requirements inside the tool while, previously, many import and export actions had to be initiated from the tool.

3.4 Usability

Usability was largely stressed by our Chinese customer. The standard model edition features had to be enriched with extensions in order to:

– provide quick access to frequently used features, with minimal number of clicks and even keyboard shortcuts.
– support batch operation over multiple concepts (e.g. move, change type).
– tune graphical representation of concepts based on meta-model extensions (e.g. through decorations on such extended concepts).
– provide efficient default graphical layout and include filtering capabilities.

3.5 Versioning

Model versioning is required to track the model evolution. Versioning is supported by the CDO model repository [33]. However, the provided technical features had to be translated to a more intuitive user experience. Our implementation started with the support of a single baseline and was extended to multiple baselines with comparison and rollback capabilities. Access to concepts history was also made easily accessible at the user interface level to ease collaboration.

4 Analysis of the High-Availability Protocol

This section studies the robustness of server operation. The server is taking care of model manipulation initiated from the client side and implemented through a well-defined API. Its implementation can be assumed stateless because in case of crash, a server session can easily be restarted without impacting the client.

A standard solution for increasing availability and coping with high load is to use multiple servers and a load balancer/monitor front-end service, like NGINGX [34]. A typical deployment with three servers is depicted in Fig. 3(a). An important constraint relates to the access to the model repository: each server actually maintains a form of cache of modelling concepts related to its user sessions. However, all the traffic to the model repository needs to be processed by a single node which is the gateway to the model repository. This node ensures the serialisation of changes and notifies all other nodes of the relevant changes through a synchronisation mechanism. As this node has a specific role, we call it *master* in our architecture. Some other unique responsibilities may also be assigned to this node.

As the master is different from the other servers, its failure cannot be resolved by simply redirecting the traffic to another node as this is the case for non-master nodes as described in Fig. 3(b).

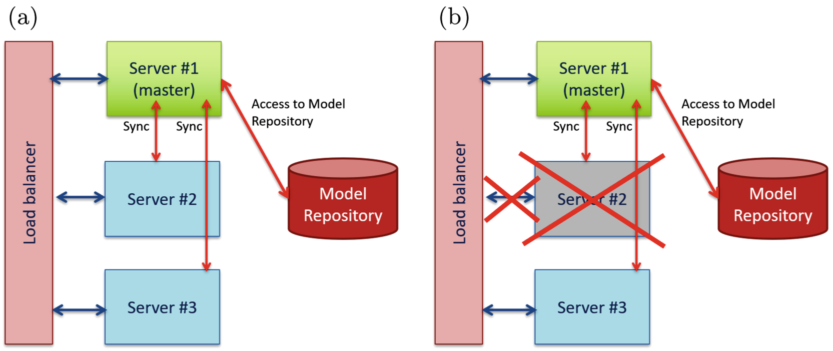


Fig. 3. (a) Fully operational system (b) Failure of a non-master server

4.1 Informal Model for Master Recovery

As depicted in Fig. 4(a), in case of crash of the master server, the access to the repository is also lost and the whole system is going to freeze until a new master is restored. Hence, it must happen quickly.

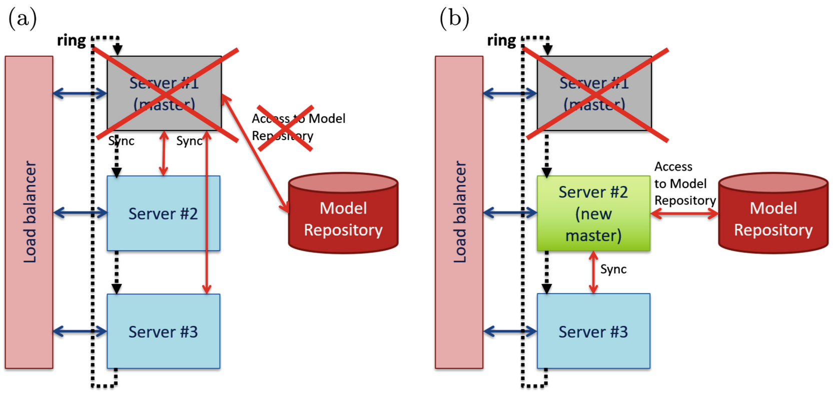


Fig. 4. (a) Failure of the master server (b) Recovery (new master election)

For finding a new master, we will consider the nodes form a logical ring, e.g. from the lower to highest IP address and then back to the lowest one. Given that setting, we could apply a standard leader election protocol such as Chang and Roberts [6]. However, we do not restrict communication to message passing between adjacent nodes. We also need to consider the sub-ring formed by the working servers because, in our algorithm, we choose to select as new master, the first available server following the crashed master. This master will then reconnect to the database and start synchronising will all remaining servers. This will result in the new operational situation depicted in Fig. 4(b) where server#2 is the new master. Note that is the crashed master is restarted, it will start acting as a normal server and will synchronise with master server#2.

In order to achieve this, the following rules are applied: a server detecting a master crash will start scanning the previous nodes in the logical ring

– until finding a running server, then this server has priority to become master and the server will wait for this new master to come up and contact him.
– or finding the crashed master, then the server is the first alive server after the master and it should become the master. It will then notify all nodes it is now the master. All other nodes will then reinitialise their synchronisation link with the new master.

Given the load-balancer only directs request to working nodes, if the master is crashed, the request must go through another server and this will trigger the change of master as soon as there is a client request. Periodic monitoring requests internal to the platform can also be used to avoid waiting for a client request.

4.2 Formal Modelling with Alloy

In order to make sure our protocol is behaving as expected, we decided to model it and verify it using a formal modelling. We selected Alloy because it is a lightweight formal method [10,12]. On the tool side, the Alloy Analyser relies on model-checking, which is fully automated contrary to theorem proving, and has a nice way for visualising solutions with many filtering and rendering capabilities [11]. This section first describes the static part of the model, then its behaviour and finally, different validation experiments.

4.3 Structure of the System

The system structure is described in Listing 1.1. It relies on *Time* and *Server* signatures which are ordered using the available ordering module. The ring structure is enforced using a circular constraint on the time-independent *succ* attribute. Three other time-dependent attributes are used: *crashed*, which records at which time a server was crashed, *master*, which records at which time a server was a master and *link*, which records who each server believes is the master at a given time. A key requirement that needs to be checked, is that at any given time only one master may exist.

Listing 1.1. Structure of the System

```
open util/ordering[Time] as TO     -- time steps is ordered
open util/ordering[Server] as SO   -- processes are ordered

sig Time {}                       -- Time steps (ordered)

sig Server {                      -- Server node (ordered)
  succ: Server,                   -- this is a static topology
  crashed: set Time,              -- captures when a server is crashed
  master: set Time,               -- captures when a server is master
  link: Time -> lone Server       -- captures knowledge of a node about master
}

fact ring {                       -- Server nodes are constrained to form a ring
  all p: Server | Server in p.^succ
}
```

4.4 Dynamic Modelling for Maintaining Master Node

In order to build a dynamic model, we use standard Alloy modelling guidelines [9], i.e. we define a trace composed of sequence of *Time*, starting with some initialisation *init* with no crashed server and the master allocated on the first one. Each pair of successive *Time* of a trace is constrained to be either a *normalOperation* when the *master* to be up and running, or a *recoverOperation* when this is not the case. The *masterAvailable* predicate is used for this test.

In *normalOperation*, the state is globally unchanged: a crashed node remains crashed (repair is considered later) but we allow new nodes to crash, so we can study server unreliability. However, we do not allow all the nodes to crash (see *notFullyCrashed* predicate) because in that case no solution is possible.

In *recoveryOperation*, the first non-crashed successor of the crashed master is selected as new master. This node is identified in a single *Time* step through a transitive closure on the *succ* function with domain and range filtering to discard crashed node. All other Servers are then informed of this new master by directly changing their *link* relationship. Listing 1.2 presents the full specification of this behavioural part.

Listing 1.2. Behaviour of the System

```
pred init [t: Time] {
  t in SO/first.master
  all s:Server |
    (t ∉ s.crashed)
    and (s≠SO/first =\textgreater t ∉ s.master)
    and (s.link[t]=SO/first)
}

pred masterAvailable(t:Time) {
  all s:Server |
    t in s.master =\textgreater t ∉ s.crashed
}

pred notFullyCrashed(t:Time) {
  some s:Server | t ∉ s.crashed
}
```

```
pred normalOperation [t, t': Time, s: Server] {
  masterAvailable[t]
  t in s.crashed =\textgreater t' in s.crashed  -- crashed stuff remains so
  --  but note that new crash may occur !
  in s.master iff t' in s.master   -- nothing changed about master routing
  s.link[t']=s.link[t]             -- nothing changed about master routing
  notFullyCrashed[t']              -- restricting fault model
}

pred recoverOperation [t,t': Time, s: Server] {
  not masterAvailable[t]
  let select=(^(crashed.t <: succ)) :> (Server-crashed.t) | -- new master !
    (t' in s.master <=> select[s.link[t]]=s)  -- new master
        and s.link[t']=master.t.(select)           -- updating links
  t' in s.crashed iff t in s.crashed           -- no crash during recovery
}

fact traces {           -- fact for constraining traces to allowed
    ↪ operations
  init [first]
  all t: Time-last | let t' = t.next | all s:Server |
     normalOperation [t, t', s] or recoverOperation [t, t', s]
}
```

4.5 Model Validation

Prior to model-checking, it is important to validate the consistency of the model, i.e. that it has instances and that those instances match the intended behaviour. In order to validate our model, we first look for *SingleMasterCrash* in cascade, i.e. each time there is a master, it should be crashed the next time, as this is allowed by our *normalOperation*. The expected behaviour is that the next server should take over as master and then crash, hence the master server will progress around the ring. Note that because crashed nodes remains crashed, no instance will be possible if there are more time steps than the double of the size of the ring. The resulting scenario is depicted in Fig. 5(a) for the three first time steps and it behaves as expected.

Listing 1.3. Behaviour of the System

```
-- find some instance with a lot of server crashing
pred singleMasterCrash { all t:Time | all s:Server |
  t in s.master => t.next in s.crashed }
run singleMasterCrash for 5 Server, 8 Time
-- find some instance with a lot of server crashing and first backup node
     ↪ too
pred MasterAndBackupCrash {  all t:Time | all s:Server |
  t in s.master => (t.next in s.crashed and t.next in s.succ.crashed) }
run MasterAndBackupCrash for 5 Server, 5 Time
```

A second validation is more naughty and involves the simultaneous crash of the master and its backup node (i.e. immediate successor). In this case, we expect the second successor server of the master to take over. The trace in Fig. 5(b) (limited to the three first time steps here) shows this is the observed behaviour.

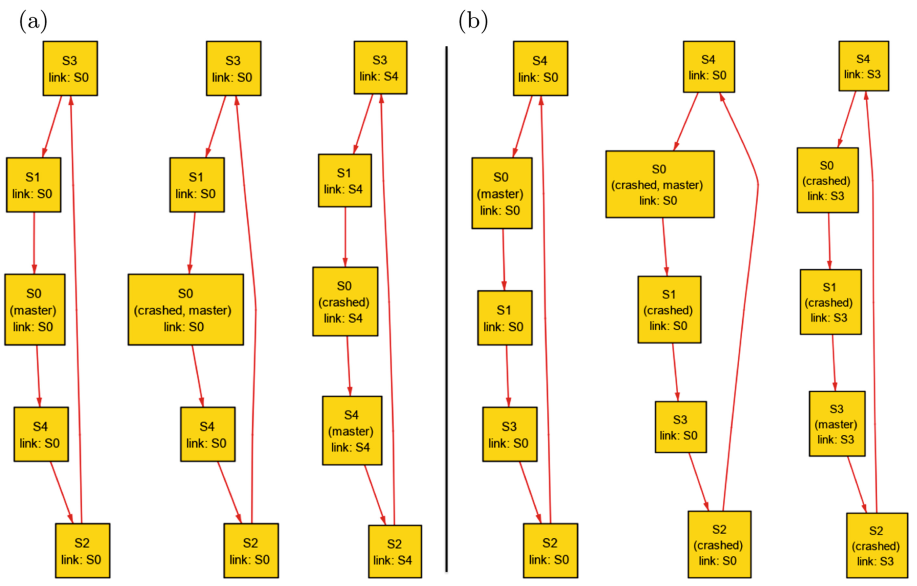


Fig. 5. Behaviour in case of (a) single failure (b) double failure

4.6 Model Checking

Finally, we can ask the analyser to verify the uniqueness of the master at all times. Rather than requiring exactly one master, actually two separate verifications are performed: at least one master and at most one master (see Listing 1.4). Their conjunction yields the wished property, but each kind of violation is more interesting to study separately.

Listing 1.4. Behaviour of the System

```
-- no multiple masters allowed
assert AtMostOneMaster { all t:Time | lone s:Server | t in s.master }
check AtMostOneMaster for 5 Server, 15 Time

-- at least one master allowed
assert AtLeastOneMaster { all t:Time | some s:Server | t in s.master  }
check AtLeastOneMaster for 5 Server, 15 Time
```

Listing 1.5 recapitulates the running time of all the checks performed on a core I7 laptop with a 64 bit Java Virtual Machine. One can see the validation are straightforward, meaning it is easy to find instances of the model, while the verification took much longer: about 5 s for *AtMostOneMaster* and about 25 s for *AtLeastOneMaster* for 5 servers and 15 units of *Time*. The verification did not find any counter-example meaning the model might be valid. Given the limited variety of scenarios, one might be confident the system is indeed correct. However, the behaviour should be studied in further details by removing some of the limitations:

- by allowing crashed servers to become operational again during operation mode. In this case, the verification is still fine but takes more time for *AtLeastOneMaster* (about 2 min)
- by allowing failures during the restoration step. In this case, there can be scenarios without any master beyond a given step after a crash of all servers. When excluding fully crashed states, the verification is fine too.

Listing 1.5. Run result

```
Executing "Run singleMasterCrash for 5 Server, 8 Time"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   7590 vars. 305 primary vars. 21359 clauses. 31ms.
   Instance found. Predicate is consistent. 32ms.
Executing "Run MasterAndBackupCrash for 5 Server, 5 Time"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4773 vars. 200 primary vars. 12892 clauses. 22ms.
   Instance found. Predicate is consistent. 31ms.
Executing "Check AtMostOneMaster for 5 Server, 15 Time"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   14562 vars. 565 primary vars. 41731 clauses. 62ms.
   No counterexample found. Assertion may be valid. 4919ms.
Executing "Check AtLeastOneMaster for 5 Server, 15 Time"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   14554 vars. 565 primary vars. 41729 clauses. 63ms.
   No counterexample found. Assertion may be valid. 25207ms.

4 commands were executed. The results are:
   #1: .singleMasterCrash is consistent.
   #2: .MasterAndBackupCrash is consistent.
   #3: No counterexample found. AtMostOneMaster may be valid.
   #4: No counterexample found. AtLeastOneMaster may be valid.
```

5 Related Work and Discussion

Several web-based tools are available to develop diagrams in different notations such as UML, BPMN and flowcharts [4,15,17,18]. They provide an easy way to draw diagrams from a web browser without requiring any installation, to save them in the Cloud and to share access with other team members. Although some may be based on Open Source [4], they all adopt a Software as a Service (SaaS) model with pay-per-use beyond a limited basic offer, e.g. to support larger models, more concurrent users, or tool integration. The majority of those tools focus on the graphical notations and do not stress the model behind them, nor the API to be able to access that model. However, some tools provide such an API, for example GenMyModel has an API to return user information, project details, execute project commands, return project tag data, and more [16]. Cacoo provides a quite similar API [19]. However when testing those tools and analysing their API, it appears that many of them have a weak notion of model, i.e. concept and their representation are not distinguished making impossible to share concept across diagrams. This also limits the ability to feed the model into a MDE toolchain. An exception is GenMyModel which also provides EMF import and export. Our approach is close the GenMyModel as we support a strong notion of model and provide an RESTFul API with all the usual CRUD operations on model concepts and representations. Beyond this, we also support project/user level operations and more advanced operations, for example to manage model versioning.

Our approach relies on the EMF Open Source modelling frameworks which is actually widespread in the research community but less in the industrial world where the majority of modelling tools are Closed Source, e.g. Rhapsody, MagicDraw and Enterprise Architect. This means such tools are missing recent advanced made by research tools. Our work aims at bridging this gap by enabling different forms of integration. Other researchers have also explored how to address this problem through mechanisms going beyond the pure exchange of models in standard formats like XMI [24] or through protocols like OSLC [20]. An attempt to bridge a proprietary UML modelling tool (PTC Integrity Modeller) with an Open Source family of languages for automated model management (Epsilon) is discussed in [38]. The question is also crucial in Cyber Physical Systems to support model integration across domains. OpenMETA was applied for the design and implementation of an experimental design automation tool suite [35]. It could provide multiple level of abstraction, correctness-by-construction in an heterogeneous context and reuse of Open Source tools assets. Our work is following the same design principles but with a bigger priority on tool reliability and availability.

6 Conclusion and Future Work

In this paper, we first investigated key non-functional requirements for building a MBSE toolchain based on our industrial experience, focusing on the early analysis steps. Although our work is driven by a specific case, the identified NFR are of general nature and are also reported by others in the literature. So we believe our feedback can be useful in other cases. Then, we focused on the specific NFR of high-availability in the context of pool of servers with a single repository access. We proposed a design to maintain a master node in a reliable way by modelling and verifying our design using the Alloy analyser. Although our solution was developed in the context of an EMF data store, we believe that the problem is more general in nature and that our solution can be reused.

In our future work, we plan to keep improving availability by also investigating problems on the repository and the load-balancing components, e.g. through mirroring or mutual monitoring. We also plan to refine our model at a finer level of operation (i.e. message level). For example, our model does not capture the behaviour when a server is crashing during the notification phase. Our intent is also to investigate another formal method supporting model refinement, such as Event-B and the Rodin toolkit [1,2]. We also plan to further analyse security requirements, which were not within the scope of our initial work because the tool was deployed within a secured intranet.

Acknowledgements. This research was partly supported by the SAMOBIGrow project (nr. 1910032). We thank Respect-IT and Huawei for their feedback in the elaboration of this tooling. We also thank the reviewers for their comments.

References

1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering, 1st edn. Cambridge University Press, New York (2010)
2. Abrial, J.R., et al.: Rodin: an open toolset for modelling and reasoning in event-B. STTT **12**(6), 447–466 (2010)
3. Adzic, G.: Specification by Example: How Successful Teams Deliver the Right Software, 1st edn. Manning Publications Co., Greenwich (2011)
4. Alder, G., Benson, D.: draw.io (2011). https://about.draw.io/integrations
5. Amyot, D., Mussbacher, G.: User requirements notation: the first ten years, the next ten years. JSW **6**(5), 747–768 (2011)
6. Chang, E., Roberts, R.: An improved algorithm for decentralized extrema-finding in circular configurations of processes. Commun. ACM **22**(5), 281–283 (1979)
7. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. **20**(1–2), 3–50 (1993)
8. Darimont, R., Zhao, W., Ponsard, C., Michot, A.: A modular requirements engineering framework for web-based toolchain integration. In: 24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, 12–16 September, pp. 405–406 (2016)
9. Dennis, G., Seater, R.: Alloy Analyzer 4 Tutorial Session 4: Dynamic Modeling Software. Design Group. MIT (2017)
10. Jackson, D.: Alloy: a lightweight object modelling notation. ACM Trans. Softw. Eng. Methodol. **11**(2), 256–290 (2002)
11. Jackson, D.: Alloy Analyser, Version 4 (2006). http://alloytools.org
12. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press, Cambridge (2012)
13. Kahani, N., Bagherzadeh, M., Cordy, J.R., Dingel, J., Varró, D.: Survey and classification of model transformation tools. Softw. Syst. Model. **18**(4), 2361–2397 (2018)
14. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, Hoboken (2009)
15. Legrand, T.: Genmymodel (2012). https://www.genmymodel.com
16. Legrand, T.: GenMyModel API Documentation (2014). https://api.genmymodel.com/doc
17. Lucid Software: Lucidchart (2008). https://www.lucidchart.com
18. Nulab Inc.: Cacoo (2009). https://cacoo.com
19. Nulab Inc.: Cacoo API Overview (2012). https://developer.nulab.com/docs/cacoo
20. OASIS: Open Services for Lifecycle Collaboration (2008). https://open-services.net
21. OMG: Unified modeling language (1997). http://www.omg.org/spec/UML
22. OMG: System modeling language (2005). http://www.omg.org/spec/SysML
23. OMG: MOF Model to Text Transformation Language (2008). http://www.omg.org/spec/MOFM2T
24. OMG: XML Metadata Interchange v2.5.1 (2015). https://www.omg.org/spec/XMI
25. Ponsard, C., Darimont, R.: Improving requirements engineering through goal-oriented models and tools: feedback from a large industrial deployment. In: Proceedings of 12th International Conference on Software Technologies, ICSOFT, Madrid, Spain, 24–26 July 2017
26. Ponsard, C., Darimont, R., Michot, A.: Combining models, diagrams and tables for efficient requirements engineering: lessons learned from the industry. In: INFORSID 2015, Biarritz, France, June 2015

27. Ponsard, C., Deprez, J.C., Delandtsheer, R.: Is my formal method tool ready for the industry? In: 11th International Workshop on Automated Verification of Critical Systems, Newcastle, UK, 12–14 September 2011
28. Ponsard, C., Michot, A., Darimont, R., Zhao, W.: A generic rest API on top of eclipse CDO for web-based modelling. EclipseCon France, Toulouse, June 2016
29. Ryan, M., Cook, S., Scott, W.: Application of MBSE to requirements engineering research challenges. In: Systems Engineering, Test and Evaluation Conference SETE2013, Canberra, Australia, April 2013
30. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006). https://doi.org/10.1109/MC.2006.58
31. Soukaina, M., Abdessamad, B., Abdelaziz, M.: Model driven engineering tools: a survey. Am. J. Sci. Eng. Technol. **3**(2), 29 (2018)
32. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional, Upper Saddle River (2009)
33. Stepper, E.: Connected data object (2006). https://www.eclipse.org/cdo
34. Sysoev, I.: Nginx (2004). https://nginx.org
35. Sztipanovits, J., et al.: Model and tool integration platforms for cyberphysical system design. Proc. IEEE **106**(9), 1501–1526 (2018)
36. Wynne, M., Hellesoy, A.: The Cucumber Book. The Pragmatic Programmers. Pragmatic Bookshelf, Dallas (2012)
37. Yu, E.S.K., Mylopoulos, J.: Enterprise modelling for business redesign: the i* framework. SIGGROUP Bull. **18**(1), 59–63 (1997)
38. Zolotas, A., et al.: Bridging proprietary modelling and open-source model management tools: the case of PTC integrity modeller and epsilon. In: Software & Systems Modeling (2019)

Author Index