Antônio Carlos da Rocha Costa
Rosa Maria Vicari
Flavio Tonidandel (Eds.)

LNAI 6404

Advances in Artificial Intelligence – SBIA 2010

20th Brazilian Symposium on Artificial Intelligence
São Bernardo do Campo, Brazil, October 2010
Proceedings

Lecture Notes in Artificial Intelligence 6404

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Antônio Carlos da Rocha Costa
Rosa Maria Vicari
Flavio Tonidandel (Eds.)

Advances in Artificial Intelligence – SBIA 2010

20th Brazilian Symposium on Artificial Intelligence
São Bernardo do Campo, Brazil, October 23-28, 2010
Proceedings

Springer

Series Editors

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

Volume Editors

Antônio Carlos da Rocha Costa
Universidade Federal do Rio Grande – FURG
Centro de Ciências Computacionais
Av. Itália, km 8 – Campus Carreiros, 96.201-900 Rio Grande, RS, Brazil
E-mail: ac.rocha.costa@gmail.com

Rosa Maria Vicari
Universidade Federal do Rio Grande do Sul – UFRGS
Instituto de Informática
Av. Bento Gonçalves 9.500, 91501-970 Porto Alegre, RS, Brazil
E-mail: rosa@inf.ufrgs.br

Flavio Tonidandel
Centro Universitário da FEI
Departamento de Ciência da Computação
Av. Humberto A. C. Branco 3972, 09850-901 São Bernardo do Campo, SP, Brazil
E-mail: flaviot@fei.edu.br

Library of Congress Control Number: 2010935023

CR Subject Classification (1998): I.2, H.3, H.4, I.4, I.5, H.5

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-16137-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-16137-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The SBIA conference series started in 1984 at the Federal University of Rio Grande do Sul (UFRGS) and through the years has benefited the Artificial Intelligence and Computer Science communities in Brazil.

After 26 years and 20 conferences SBIA is now a mature event, constituting a discussion forum for new ideas in all sub-areas of AI.

In this book you will find the full papers selected for publication in the SBIA 2010 proceedings. The papers cover the AI sub-areas in the following way:

- Ontologies, Knowledge Representation, and Reasoning: 8
- Machine Learning: 2
- Autonomous Agents and Multiagent Systems: 6
- Natural Language Processing: 2
- Planning and Scheduling: 5
- Logics for AI: 3
- Constraints and Search: 5

We would like to thank all the authors that contributed to SBIA 2010. We also thank all the members of the international Program Committee and the additional reviewers, who did an excellent job in reviewing the papers.

We are very grateful to Flavio Tonidandel, General Chair of SBIA 2010 and of the Joint SBIA/SBRN/JRI 2010 Conference, for all the support that he and his team at FEI provided.

Yoav Shoham, Jaime Sichman, and David Hogg were the keynote speakers of the event. We thank them very much for their acceptance of the invitation.

A special acknowledgement is due to Tiago Thompsen Primo, for his dedicated effort in the editing of these proceedings.

Finally, we thank the SBIA 2010 sponsors (CAPES, CNPq, FAPESP, and SBC) for their support.

August 2010

Antônio Carlos da Rocha Costa
Rosa Maria Vicari

Organization

Organizing Committee

General Chair

Flavio Tonidandel	Centro Universitário da FEI

Workshop Chair

Plínio Thomaz Aquino Jr.	Centro Universitário da FEI

Tutorial Chair

Paulo Sérgio Silva Rodrigures	Centro Universitário da FEI

Steering Committee (CEIA)

Augusto Loureiro da Costa	Universidade Federal da Bahia (Coordinator)
Solange Oliveira Rezende	Universidade de São Paulo
Fernando Santos Osório	Universidade de São Paulo
Fabio Gagliardi Cozman	Universidade de São Paulo
Marcelo Finger	Universidade de São Paulo
Fred Freitas	Universidade Federal de Pernambuco
Alexandre da Silva Simões	Universidade Estadual Paulista

Program Committee

Adina Magda Florea	Polytechnic University of Bucharest, Romania
Adolfo Arenas	Inst. Politec. Nal. Campus "Lopez Mateos", Mexico
Adolfo Neto	UTFPR, Brazil
Adriano Werhli	FURG, Brazil
Alejandro Zunino	ISISTAN-UNICEN, Argentina
Alessio Lomuscio	Imperial College, UK
Alexander Gelbukh	National Polytechnic Institute, Mexico
Alexandre Silva	UFRJ, Brazil
Aline Villavicencio	UFRGS, Brazil
Alneu Lopes	USP, Brazil
Alvaro Moreira	UFRGS, Brazil
Amilcar Cardoso	University of Coimbra, Portugal
Ana Bazzan	UFRGS, Brazil
Ana Bicharra	UFF, Brazil

Ana Casali	Universidad Nacional de Rosario, Argentina
Ana Carolina Bertoletti De Marchi	UPF, Brazil
Ana Teresa Martins	UFC, Brazil
Anarosa Brandao	USP, Brazil
André Campos	UFRN, Brazil
André Ponce de Leon F. de Carvalho	USP, Brazil
Andrea Omicini	University of Bologna, Italy
Anna Helena Reali Costa	USP, Brazil
Anne Canuto	UFRN, Brazil
Antônio Braga	UFMG, Brazil
Antonio Carlos da Rocha Costa	FURG, Brazil
Artur Garcez	City University of London, UK
Augusto Loureiro da Costa	UFBA, Brazil
Aurora Pozo	UFPR, Brazil
Barbara Hammer	Clausthal University of Technology, Germany
Benjamin Bedregal	UFRN, Brazil
Bianca Zadrozny	UFF, Brazil
Blai Bonet	Universidad Simón Bolívar, Venezuela
Carlos Reyes-Garcia	INAOE, Mexico
Carlos Ribeiro	ITA, Brazil
Cassio de Campos	Dalle Molle Institute for AI, Switzerland
Celso Kaestner	UTFPR, Brazil
Christian Lemaitre	Universidad Autónoma Metropolitana, Mexico
Clarisse de Souza	PUC-Rio, Brazil
Daniel Berrar	University of Ulster, UK
Diana Adamatti	FURG, Brazil
Donato Malerba	Università degli Studi di Bari, Italy
Edson Matsubara	UFMS, Brazil
Edward Hermann Haeusler	PUC-Rio, Brazil
Eric Matson	Purdue University, USA
Eugenio Oliveira	Universidade do Porto, Portugal
Evandro Costa	UFAL, Brazil
Fabiano Silva	UFPR, Brazil
Fariba Sadri	Imperial College, UK
Fábio Cozman	USP, Brazil
Federico Barber	Universidad Politecnica de Valencia, Spain
Fernando Osório	USP, Brazil
Filip Zelezny	Czech Techn. University in Prague, Czech Republic
Flavio Tonidandel	FEI, Brazil
Flávio Soares Corra da Silva	USP, Brazil
Francisco de Carvalho	UFPE, Brazil

Frank Dignum	Utrecht University, Netherlands
Fred Freitas	UFPE, Brazil
Gabriel Lopes	Universidade Nova de Lisboa, Portugal
Geber Ramalho	UFPE, Brazil
Gerardo Schneider	University of Gothenburg, Sweden
Gerson Zaverucha	UFRJ, Brazil
Graça Gaspar	Universidade de Lisboa, Portugal
Graçaliz Dimuro	FURG, Brazil
Guilherme Barreto	UFC, Brazil
Guillermo Simari	Universidad Nacional del Sur, Argentina
Gustavo Giménez-Lugo	UTFPR, Brazil
Hector Geffner	Universitat Pompeu Fabra, Spain
Helder Coelho	Universidade de Lisboa, Portugal
Heloisa Camargo	UFSCar, Brazil
Ines Dutra	Universidade do Porto, Portugal
Irwin King	The Chinese University of Hong Kong, China
Ivandré Paraboni	USP, Brazil
Jaime Sichman	USP, Brazil
James Cussens	University of York, UK
Jan Ramon	Katholieke Universiteit Leuven, Belgium
João Gama	Universidade do Porto, Portugal
João Marcos	UFRN, Brazil
João Balsa	Universidade de Lisboa, Portugal
João Luis Tavares da Silva	UCS, Brazil
Jomi Hübner	UFSC, Brazil
Juergen Dix	Clausthal University of Technology, Germany
Julie Dugdale	Laboratoire d'Informatique de Grenoble, France
Krysia Broda	Imperial College, UK
Laurent Perrussel	Université de Toulouse, France
Lúcia Drummond	UFF, Brazil
Leliane Nunes de Barros	USP, Brazil
Leonardo Emmendorfer	FURG, Brazil
Li Weigang	UnB, Brazil
Luis Antunes	Universidade de Lisboa, Portugal
Luis Fariñas del Cerro	Université Paul Sabatier, France
Luis Lamb	UFRGS, Brazil
Luis Otávio Alvares	UFRGS, Brazil
Luiz Satoru Ochi	UFF, Brazil
Luiza Mourelle	UERJ, Brazil
Mara Abel	UFRGS, Brazil
Marcelo Finger	USP, Brazil
Marcilio de Souto	UFRN, Brazil
Marco Gori	University of Siena, Italy

Marco Antonio Casanova	PUC-Rio, Brazil
Marcos Castilho	UFPR, Brazil
Maria Monard	USP, Brazil
Maria das Graças Volpe Nunes	USP, Brazil
Marilton Aguiar	UFPEL, Brazil
Mario Fernando Campos	UFMG, Brazil
Marley Vellasco	PUC-Rio, Brazil
Mário Benevides	UFRJ, Brazil
Michael Thielscher	TU Dresden, Germany
Nuno David	ISCTE, Portugal
Olivier Boissier	EMSE, France
Pablo Noriega	IIIA-CSIC, France
Patrícia Tedesco	UFPE, Brazil
Patricia Jaques	UNISINOS, Brazil
Paulo Ferreira Jr.	UFPEL, Brazil
Paulo Quaresma	Universidade de Évora, Portugal
Paulo Santos	FEI, Brazil
Paulo Trigo	ISEL, Portugal
Pedro Larrañaga	Polytechnic University of Madrid, Spain
Ramon de Mantaras	IIIA-CSIC, Spain
Reinaldo Bianchi	FEI, Brazil
Rejane Frozza	UNISC, Brazil
Renata de Freitas	UFF, Brazil
Renata Vieira	PUCRS, Brazil
Renata Wassermann	USP, Brazil
Ricardo Rabelo	UFSC, Brazil
Ricardo Silva	University College of London, UK
Ricardo Silveira	UFSC, Brazil
Ronaldo Prati	UFABC, Brazil
Rosa Viccari	UFRGS, Brazil
Rui Camacho	Universidade do Porto, Portugal
Sandra Sandri	IIIA, Brazil
Sheila Veloso	UERJ, Brazil
Silvia Botelho	FURG, Brazil
Silvia Schiaffino	ISISTAN, Argentina
Siome Goldenstein	UNICAMP, Brazil
Solange Rezende	USP/S.Carlos, Brazil
Soumya Ray	Oregon State University, USA
Stan Matwin	University of Ottawa, Canada
Stanley Loh	UCPEL/ULBRA, Brazil
Stefano Ferilli	Università degli Studi di Bari, Italy
Teresa Ludermir	UFPE, Brazil
Thiago Pardo	USP, Brazil

Toby Walsh	NICTA and Univ. New South Wales, Australia
Torsten Schaub	University of Potsdam, Germany
Virginia Dignum	Delft Univ. of Technology, Netherlands
Valerie Camps	Université Paul Sabatier, France
Vania Bogorny	UFSC, Brazil
Vítor Santos Costa	Universidade do Porto, Portugal
Vera Lúcia Strube de Lima	PUCRS, Brazil
Viviane Torres da Silva	UFF, Brazil
Vladik Kreinovich	University of Texas at El Paso, USA
Wagner Meira, Jr.	UFMG, Brazil
Walter Carnielli	UNICAMP, Brazil
Walter Daelemans	University of Antwerp, Belgium
Wamberto Vasconcelos	University of Aberdeen, UK
Wiebe van der Hoek	University of Liverpool, UK
Yves Demazeau	Laboratoire d'Informatique de Grenoble, France
Zhao Liang	USP, Brazil

Additional Reviewers

André Lins Aquino	UFOP, Brazil
Caecilia Zirn	Univ. Heidelberg, Germany
Cassia Santos	Evora University, Portugal
Christian Vogler	ILSP, Greece
Daniel Weingaertner	Universidade Federal do Paraná, Brazil
Eder Mateus Gonçalves	FURG, Brazil
Eugenio Silva	PUC-Rio, Brazil
Francicleber Ferreira	UFC, Brazil
Frédéric Moisan	Université de Toulouse, France
Gustavo Batista	USP, Brazil
Illya Kokshenev	UFMG, Brazil
João Alcântara	UFC, Brazil
Karla Figueiredo	PUC-Rio, Brazil
Luis Künzle	UFPR, Brazil
Martin Musicante	UFRN, Brazil
Natasha Lino	UFPb, Brazil
Nicolau Werneck	USP, Brazil
Paulo Schreiner	UFRGS, Brazil
Regivan Hugo Nunes Santiago	UFRN, Brazil
Rinaldo Lima	UFPE, Brazil
Rodrigo Wilkens	UFRGS, Brazil
Rosiane de Freitas Rodrigues	UFAM, Brazil
Sérgio Almeida	UCPel, Brazil
Thiago Noronha	UFMG, Brazil

Supporting Scientific Society

SBC - Sociedade Brasileira de Computação.

Sponsoring Institutions

CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológica
CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo

Table of Contents

Chapter 1: Ontologies, Knowledge Representation and Reasoning

Chapter 2: Machine Learning

Chapter 3: Autonomous Agents and Multiagent Systems

Chapter 4: Natural Language Processing

Chapter 5: Planning and Scheduling

Chapter 6: Constraints and Search

Chapter 7: Logics for AI

Ontological Primitives for Visual Knowledge

Alexandre Lorenzatti[1], Mara Abel[1], Sandro Rama Fiorini[1],
Ariane K. Bernardes[2], and Claiton M.S. Scherer[2]

[1] Instituto Informática – Universidade Federal do Rio Grande do Sul
{alorenzatti,marabel,srfiorini}@inf.ufrgs.br
[2] Instituto Geociências – Universidade Federal do Rio Grande do Sul
claiton.scherer@ufrgs.br

Abstract. In the last few years, we have analyzed the best alternatives for acquiring and processing visual knowledge with the goal of supporting problem solving. We call *visual knowledge* the set of mental models that support the process of reasoning over information that comes from the spatial arrangement and visual aspects of entities. Also, visual knowledge is implicit, meaning that it is difficult to be explicitly represented solely with propositional constructs. In this paper, we describe a representational approach that helps geologists in capturing and applying this kind of knowledge, in order to support software development applied to interpretation tasks in Petroleum Geology applications. Our approach combines propositional constructs with visual pictorial constructs in order to model visual knowledge of geologists. These constructs are proposed in a strong formal model, founded by Formal Ontology concepts. Based on these constructs, we develop a full ontology for stratigraphic description of sedimentary facies. The Formal Ontology background and the approach are detailed and evaluated through the paper.

1 Introduction

Imagistic domains are those in which the problem-solving process starts with a visual pattern-matching process that captures the information, which will further support the abstract inference process of interpretation. Image-based diagnosis in Medicine, biological analysis and Geological interpretation are common examples of tasks that are based on visual interpretation. Moreover, the decrease in the price of image-capturing devices has caused a quick dissemination of the use of images as complement of information register in general: reports, sites, databases. Otherwise, the indexing, searching and retrieving of content from images has shown a difficult task that is far of getting an adequate solution. Ontologies have been shown the more promising alternative to face this challenge.

We focus this paper on the needs of representation primitives for capturing content from images and scenes in order to build domain ontologies that provide support to storing, indexing and processing visual information. We present a revision on visual-knowledge representation approaches that are being studied around the world and propose some directives to develop ontologies in imagistic domains. Our contribution is the representational approach to capture visual knowledge and the domain ontology for sedimentary facies, which was validated with the community of Stratigraphy in Geology.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 1–10, 2010.

Several approaches for developing ontologies in visual domains described in the literature have tried to translate the visual concepts into a propositional representation and deal with the semantic gap found between some pictorial representation (such as digital files that represent pictures, maps or graphs) and the semantic content expressed in a symbolic representation. Hudelot and colleagues in [1] have tried to automatic extract objects from image and to associate them to a knowledge model for shapes and, in another level, to a domain model to diagnose diseases in rose leafs. Liu, Zhang et al. [2] defined an ontology of bird anatomy shapes (body, beak and wing shapes), which is used to recognize bird species. An image processing algorithm segments bird pictures and matches them against the shapes in the library and a domain ontology of Ornithology helps in defining the specific bird. Bertini [3] uses a domain ontology and a set of visual concepts that are used to represent the visual counterpart of abstract linguistic concepts enriching ontologies with pictorial content. It is applied for digital video libraries annotation in the soccer domain. Our approach extends the previous proposal by defining domain-independent primitives – the *PictorialConcept* and *PictorialAttribute* – that expand the expressivity of pictorial representations and are possible of being processed by knowledge systems. These primitives are instantiated here in the domain of Geology.

2 Visual Knowledge

Even though they seem intrinsically connected, *visual knowledge* and *image* are disjointed concepts. In order to understand the distinction, we will start for referring the Ullmann triangle [4] that describes the relation among an *Object* in the reality, a *Concept* in some conceptualization and a *Symbol* in some language (A modified version is shown in Figure 1). An *Object* is supposed to be a real or concrete object, although its existence only can be referred through the process of perception and abstraction by someone. *Concepts*, by their side, are the way in which the human being deals with the external world, creating and manipulating mental internal representations of it in order to reason about and to act upon the environment. *Symbols* are one of the many possibilities in which concepts can be *externalized* in the process of communication in order to share the conceptualization among the community. It will depend on the language chosen by the person that should preserve some *ontological commitment* [5] with the conceptualization. One concept may have many alternative external representations in different languages for distinct purposes. The term *image*, by its side, is usually referred as a pictorial representation of an object, and should be situated in the lower side of the Ullmann triangle.

Based on these assumptions, we define visual knowledge as follows:

"Visual knowledge is the set of mental models (concepts) of real or imaginary scenes manipulated by the brain in order to deal with image-based tasks, such as, image interpretation or pattern or shape recognition in the reality."

Consequently, when we mention *visual knowledge,* we are referring to the *conceptualization* vertex of the Ullmann triangle in the human mind scope. A pictorial representation of a *scene*, like a picture, a map or a draw, is related to the language vertex,

i.e., it refers to some external representation of the internal conceptualization. Propositional and visual concepts are both part of the conceptualization and appear to be manipulated in mind in some indistinct way. The choice of one instead of other are defined by their use, for example, spatial, location problems that demand visual representation or communication problems that require shared concepts. Pictorial representations of visual knowledge are, then, produced by people in order to express the visual knowledge. They are symbols that can not be translated to propositional representations. Thus, we add a new dimension to the Ullmann triangle (the new vertex: "Pictorial representation in a Visualization", in the base of the triangle in Figure 1) in order to separate the symbolic representations of propositional and visual concepts and their visualizations. The representation of a concept is its name in some language, while the same concept can be depicted as an image, a draw, a graphic or an icon. The correspondence between the pictorial representation and the symbol in a language, when both refer to the same concept is called *anchoring*.

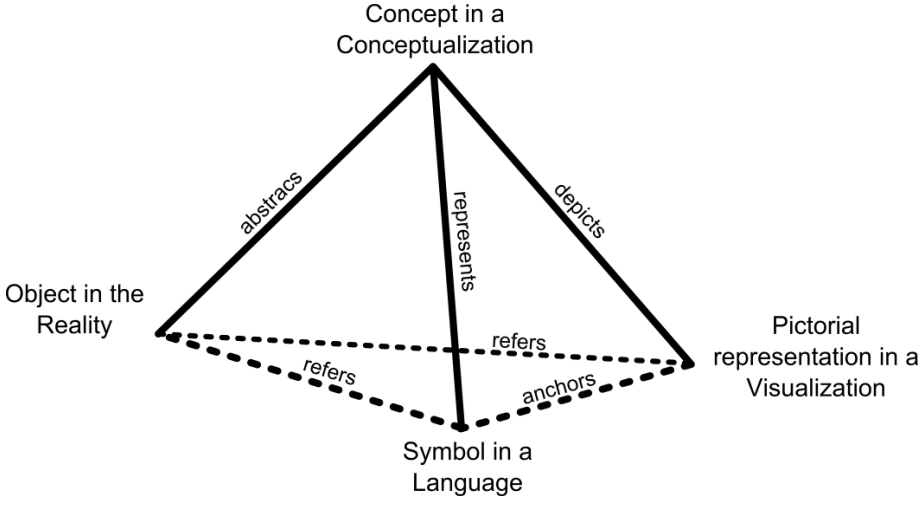


Fig. 1. The extension of the Ullmann triangle considers *visual knowledge* as a special kind of conceptualization in mind that can have either symbolic or pictorial representation

3 Conceptual Modeling Foundation

Concepts are the way in which people capture identifiable portions of reality with the purpose of understanding and knowing how to deal with them. A conceptualization is the whole set of knowledge a person has about his/her world, what was referred by Alan Newell as the *knowledge level* [6]. In order to share this conceptualization with other people or systems, someone will express it through an external representation artifact in the *symbolic level*. When a community agreed about the meaning and formalization of some representation, we call this an *ontology* [7]. More specifically, *domain ontologies* are those developed for formalizing technical vocabulary in some

domain, with the intention of making communication more efficient and support problem solving [8].

The main intention of our work is developing domain ontologies for imagistic domains, especially in Geology for supporting petroleum exploration. In order to achieve this goal, we have studied and proposed primitives that are capable to capture both the visual and propositional part of the concept. The demand for ontological definition in petroleum geology is mainly motivated by the need of terminological definition and development of interpretation systems based on symbolic information.

We have carried on knowledge acquisition and built our model based on the theoretical framework of meta-properties and meta-types proposed by Nicola Guarino [7, 9] and Giancarlo Guizzardi [5], in the context of Formal Ontology. Specially, we will mention here the notions of rigid sortals, properties, quality domains, partonomic relations and hierarchical relations, which are being used through this text. A rigid sortal is a concept whose definition requires that their instances cannot stop being an instance of this concept in any possible world. This means that if the *essential properties* chosen to define the concept cease to be recognized in the way they were defined, the instance will cease to exist too, because it looses its identity criterion.

A person is a rigid sortal while a student is not, since there are instances of it that can stop being a student without losing its identity. These are important constructs for ontological models, given that they allow producing trustful mappings among different domain ontologies that support interoperability.

A sortal is defined by its (essential) properties. In the framework defined by Guizzardi, each property is associated with a *quality domain*, which defines the set of possible values the property can assume. A quality domain can also be shared by more than one property. A value in a quality domain is called *quale*, having its own independent existence. Taking, for instance, the sortals *apple* and *car* and the quality domain *color*, both sortals have a property (e.g., *skin color* and *exterior color*) related to the same quality domain *color*. Eventually, two individuals, a red apple and a red car, will have the same *quale* as color. In this case, both individuals will be linked to the same *quale* in the *color* quality domain.

Relations represent the associative links between the objects of the domain and will define the organization and possibilities of inference in that domain. Further details of these definitions can be found in [5].

4 Sedimentary Facies Ontology

The studies on [10, 11] have proved that the expertise in Geology is mainly supported by visual information that cannot be described symbolically through geometric components, such as size and format, alone, which limits the application of machine learning or image processing algorithms. Stratigraphy is the study of sedimentary terrains in surface or subsurface, in order to define the geological history of their formation based on the description of well cores and outcrops. The main object of study and description is the *sedimentary structure*. Sedimentary structure is the external visual aspect of some internal spatial arrangement of the rock grains that, along with the preserved fossil content and the rock type, identifies the depositional environment in which the existent sediment has been deposited and consolidated in that rock. It is the

more striking visual object recognized in the domain and the first one to be used in raising interpretation hypotheses. The sedimentary structure concept comprises a challenge for ontology engineering, because of the incapability of the geologists in defining a sedimentary structure in a pure verbal (propositional) way, requiring a drawing or a picture to completely express the idea.

In order to represent that implicit part of knowledge we have developed some meta-constructs inspired in the idea of inferential *free-rides*. Atsushi Shimojima [12] formally defines inferential free-rides as the possibility of capturing semantic information through the direct correspondence among the properties of visual representations and the concepts' properties, i.e. the visual representations are built in order to express the semantic information. We have proposed two kinds of icons based on the idea of free-rides: one to represent *visual kinds* and other to represent values (*qualia*) in a *quality domain*. The method for defining the representation from the original structures is showed in [13]. We use, for example, the qualia to define the quality domain for the properties of the class sedimentary structures, and the visual kinds to define iconic representations for their subkinds. For each defined icon we associate a formal definition in a propositional form, intended to be used by reasoning methods, querying and also to produce exported reports. The propositional form is associated with the iconic representations through the proposed meta-constructs *PictorialConcept* and *PictorialAttribute*, described in the sequence.

The meta-construct *PictorialConcept* is used to represent sortals, i.e. visual kinds which instances must be so in any possible world. It is defined as the tuple

$$PC = \{V_{S_c}, V_{P_c}, A_c, i_{S_c}, i_{P_c}\}. \quad (1)$$

The sets V_{S_c} and V_{P_c} contains respectively the symbolic (propositional) and pictorial vocabulary to represent concepts. In our proposal, the pictorial vocabulary is constituted by icons. The grounding relation A_c maps elements in $V_{P_c}$to elements in V_{S_c}, representing the *anchor* axis from the extended Ullmann triangle. The relation is also total and injective, since every pictorial concept is grounded in one distinct symbolic concept. The interpretation functions i_{S_c} and i_{P_c} establish the mapping relation from the symbolic and pictorial vocabulary to the concept they represent (here representing respectively the *represents* and *depicts* axes from the Ullmann extended triangle).

The meta-construct *PictorialAttribute* is used to represent the *qualia* from the quality domains, i.e., possible values that can be assumed by a property. It is defined as the tuple

$$PA = \{V_{S_a}, V_{P_a}, A_a, i_{S_a}, i_{P_a}\}. \quad (2)$$

The sets V_{S_a} and V_{P_a} are respectively the symbolic (propositional) and pictorial vocabularies used to *represent values inside a quality domain*. In our proposal, the elements in the pictorial vocabulary are icons as well. The grounding relation A_a maps elements in V_{P_a} to elements in V_{S_a}, representing the *anchor* axis from the extended Ullmann triangle. The relation A_a is also total and injective. The interpretation functions i_{S_a} and i_{P_a} define the mapping relation from the symbolic and pictorial representations to the concept they represent (here also representing respectively the *represents* and *depicts* axes from the Ullmann extended triangle).

In our study of stratigraphy, we elicited a set of 28 stratigraphic structures and a set of 32 qualia. They have been acquired in both propositional ($V_{S_c}$and V_{S_a}) and pictorial form. Then we defined 32 icons that describe property values (V_{P_a}) and 28 visual kinds of sedimentary structures in siliciclastic rocks (V_{P_c}). The completeness of the terminology is being tested by the geological community. Figure 2 shows a set of icons for a subset of values of the property *lamina-shape* of the concept Sedimentary structure.

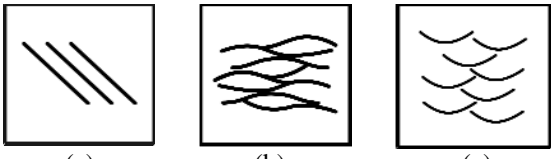


Fig. 2. Three of the seven possible values of lamina-shape property of sedimentary structure. (a) planar, (b) trough cross strata, and (c) truncated wavy lamination.

Figure 3 presents three among a hundred of defined subkind icons of sedimentary structures.

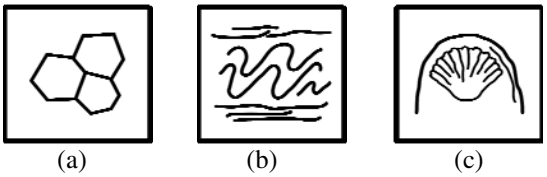


Fig. 3. Subkinds of sedimentary structure. (a) Top or base structures of mud crack, (b) deformation structure of convolute lamination, and (c) deformation structure of tool mark.

Figure 4 describes the propositional model of a subtype of the rigid sortal *sedimentary structure* defined in the ontology. Every value on the quality domain of properties (*high angle, planar and medium*) has an equivalent value in the visual qualia.

Concept Cross_stratification
Sub-type of: Traction and fallout Structure
Angularity: high angle
LaminaShape: planar
Thickness: medium
End_concept Cross_stratification

Fig. 4. Propositional definition of sedimentary structure

The knowledge acquisition carried on with the expert geologist let us to organize the sedimentary structures based on visual criteria, differently of the most common organization found in literature, which is based on environment interpretation criteria. Since the intention of the ontology is to help non-experts in the task of description, a pure descriptive organization was required to make the search of a particular sedimentary structure easy for the users.

Other important concept in our ontology is the sedimentary facies concept. Sedimentary facies group together a set of diagnostic visual aspects of sedimentary rocks strongly connected with the depositional conditions in which this rock was created. The interpretation of a facies requires the identification of the attributes Sphericity, Roundness, Geometry and Sorting, the lithotype and, if found, the fossil content. Figure 5 depicts the modeling of the sedimentary facies concept.

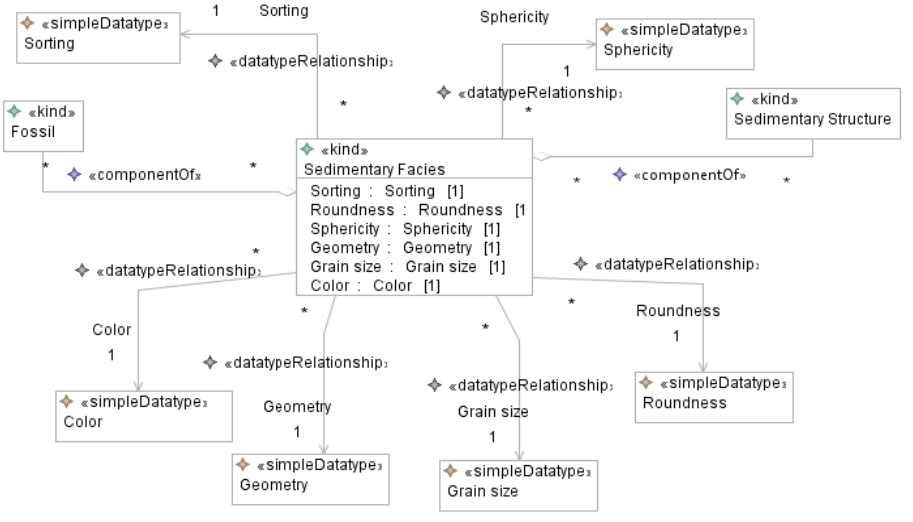


Fig. 5. Sedimentary Facies' main attributes and its relation with the Sedimentary Structure and Fossil concepts

5 Preliminary Validation of the Ontology in the Geology Community

In order to validate the method for defining icons, we carried out an empirical experiment that measured the expressiveness of the iconic representations proposed by the geologist expert. The number of participants, 21, was not statistically significant, but it was enough for a preliminary evaluation. The selected sample to be interviewed has different levels of expertise in Geology: 16 undergraduate students, 3 master students and one Ph.D. geologist. The sample was also grouped according to the amount of hours per month (in the last 12 months) that was dedicated to professional or academic activities related to the Sedimentary Stratigraphy, such as, description of cores and outcrops or interpretation.

The experiment consisted in showing to the participants a list of geological terms, assumed to be known by them, and the list of iconic representations defined in this work, which were unknown to them. Each geological term refers to only one iconic representation. The objective of the experiment was to verify if the iconic representations are expressive enough, to the point that a geologist were able to associate it with the right geological term, even without any previous training effort.

We applied a total of 64 geological terms, divided in two parts. The first part contained the terms related to attributes of sedimentary facies (34 terms) while the second part contained the terms related to sedimentary structures (30 terms). The geological terms (offered both in English and Brazilian Portuguese) were disposed in one column, while the iconic representations were disposed in a second column. The pictorial content was presented without any subtitles that could help the participants in identifying the corresponding geological term. Icons that represented values from a single quality domain were grouped. The objective of the participant is to associate the unknown icons with the geological known terms. The participants were allowed to let the answer blank in the case they didn't know the meaning of a geological term, reducing the chance in the choice of an icon.

The media of correct associations between geological terms and icons were 70.84% for sedimentary facies icons and 66.45% for sedimentary structure icons. Considering that it was the first contact with the proposed representation, the results confirm that the icons were quite intuitive and expressive as a representation of the geological features. The difference between the indexes can be explained by the different nature of facies and structures in terms of knowledge. Sedimentary facies are a formal knowledge, whose vocabulary is learnt through expository classes and immersion in the literature, while sedimentary structures are strongly visual knowledge that are learnt along the experience in the domain. Since the sample has more students than experienced geologists, the association between sedimentary structures and icons (that are, in the end, visual representations of the objects) was less effective.

The analyses of the results considered the level of formation of the participants that were grouped in three sets: undergraduate, Master and Ph.D. geologists. The undergraduate participants obtained 67.86% and 72%, Master participants obtained 70.59% and 86.36% and the PHD participant obtained 76.47% and 83.33% of correct associations for sedimentary facies and sedimentary structures terms, respectively. These results indicate that participants with a better formation could make a higher number of correct associations.

The relation between the amount of time applied in daily basis dealing with tasks of sedimentary Stratigraphy and the level of correct associations was also significant. The participants were grouped in three sets: the first one contains participants who spent less than 11 hours per month; the second one contains participant who spent 11 or more hours and less than 40 hours per month; the third one contains participants who spent 40 or more hours per month in professional and/or academic activities related to the domain in the last 12 months. Participants belonging to the first group obtained 70% and 61.11% of right associations for terms related to sedimentary facies and sedimentary structures, respectively. The second group of participants obtained indexes of 67.86% and 80%, while the third 85.29% and 83.33% of right associations between icons and the geological terms. The results corroborated what was expecting:

the more they work in recognizing facies, the better they identify the correct icon for the geological feature.

The evaluation (all indexes over 60%) showed that the proposed iconic representations have a high level of expressiveness of their meanings. It was detected also a convergence on the errors, showing that some of the chosen representations were not adequate to represent the geological feature. Those icons were further replaced in the ontology.

The results of our experiment corroborate those described in [10] that demonstrated that individuals with higher levels of expertise (formation and experience in the domain) capture and store their acquired knowledge through visual concepts and establish more associations within their mental models. The results observed in our experiment also show that participants with higher levels of expertise (formation and experience in the domain) could establish more associations among the proposed iconic representations and their mental models, thus giving an elevated number of correct associations among the icons and the geological terms.

6 Conclusion

Ontology, by its nature, is a propositional conceptual model, since it is basically constructed for communication and people are mainly verbal when exchange messages. The better a community understand the domain, the more effective will be the ontology in translating the conceptualization of the domain. In the frontiers of Science, however, a domain ontology may miss in providing adequate vocabulary that allows people to exchange their knowledge. This is especially true in imagistic domains, where the problem solver will drive his/her reasoning mainly supported by the visually collected knowledge over the domain.

We understand that the visual content of a conceptualization is an inherent part of it, and it is naturally used by people. The first step, showed in this work, was given in order to formally capture and represent the visual content into a domain ontology using special constructs. A second step is to develop a visual theory, exploring the characteristics of the human visual perceptual system and the visual properties of the representations. That theory will be focused in creating visual representations according to the ontological analysis of the nature from concept represented, providing and adequate language to capture the visual content.

The validation of the set of icons was carried on through a group of 21 geologists requested to associate each icon with its corresponding geological term without the support of the icons' labels. The results of the experiment showed that the proposed icons were able to express the concepts in the conceptualization and the participants were able to comprehend them in most of the associations.

References

1. Hudelot, C., Maillot, N., Thonnat, M.: Symbol Grounding for Semantic Image Interpretation: from image data to semantics. In: Tenth International Conference on Computer Vision. IEEE, Los Alamitos (2005)

2. Liu, Y., et al.: A Shape Ontology Framework for Bird Classification, in Digital Image Computing Techniques and Applications. In: 9th Biennial Conference of the Australian Pattern Recognition Society 2007, pp. 478–484 (2007)
3. Bertini, M., et al.: Dynamic pictorial ontologies for video digital libraries annotation. In: MS 2007: Workshop on Multimedia Information Retrieval on the Many Faces of Multimedia Semantics, pp. 47–56. ACM, New York (2007)
4. Ullmann, S.: Semantics: An Introduction to the Science of Meaning. Rowman & Littlefield, Oxford (1979)
5. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, p. 410. Universal Press (2005)
6. Newell, A., Simon, H.A.: Human Problem Solving. Prentice-Hall, New Jersey (1972)
7. Guarino, N.: Formal ontology, conceptual analysis and knowledge representation. International Journal Human-Computer Studies 43(2/3), 625–640 (1995)
8. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. In: Wu, X., Jain, L. (eds.) Springer, London (2004)
9. Guarino, N., Welty, C.A.: An overview of OntoClean. In: Staab, S., Studer, R. (eds.) Handbook of Ontologies, pp. 151–171. Springer, Heidelberg (2004)
10. Abel, M.: Estudo da perícia em petrografia sedimentar e sua importância para a engenharia de conhecimento. In: Programa de Pós-graduação em Computação, p. 239. Porto Alegre, UFRGS (2001)
11. Abel, M., et al.: Knowledge acquisition and interpretation problem-solving methods for visual expertise: a study of petroleum-reservoir evaluation. Journal of Petroleum Science and Engineering 47(1/2), 51–69 (2005)
12. Shimojima, A.: Operational constraints in diagrammatic reasoning. In: Logical Reasoning with Diagrams, pp. 27–48. Oxford University Press, Inc., Oxford (1996)
13. Lorenzatti, A., et al.: Ontology for Imagistic Domains: Combining Textual and Pictorial Primitives. In: Heuser, C.A., Pernul, G. (eds.) ER 2009 Workshops. LNCS, vol. 5833, pp. 169–178. Springer, Heidelberg (2009)

A Semi-automatic Method for Domain Ontology Extraction from Portuguese Language Wikipedia's Categories

Clarissa Castellã Xavier and Vera Lúcia Strube de Lima

Faculdade de Informática – PUCRS, Av. Ipiranga, 6681 – Prédiof 32, Porto Alegre, Brazil
{clarissa.xavier,vera.strube}@pucrs.br

Abstract. The increasing need for ontologies and the difficulties of manual construction give place to initiatives proposing methods for automatic and semi-automatic ontology learning. In this work we present a semi-automatic method for domain ontologies extraction from Wikipedia's categories. In order to validate the method, we have conducted a case study in which we implemented a prototype generating a Tourism ontology. The results are evaluated against a manually built Golden Standard reporting 79.51% Precision and 91.95% Recall, comparable to those found in the literature for other languages.

Keywords: ontologies, Wikipedia, semi-automatic ontology extraction.

1 Introduction

According to Wikipedia's project maintainer, Wikipedia is a free multilingual online encyclopedia, non-profit, collaborative, created on January 15, 2001. In April 2010, Wikipedia had more than 555,000 articles in Portuguese language and more than 3,235,000 articles in English.

Wikipedia's documents are organized in a hierarchy of categories that can be understood as a structure of terms, not strictly a tree structure, but a richer representation. This structure allows multiple simultaneous categorization of topics. Some categories may have more than one super-category [1], forming a graph that represents a conceptual network with unspecified semantic relations [2].

In Computer Science ontologies are understood as "an explicit specification of a conceptualization"[3]. There are even simpler definitions for ontologies, such as the W3C[1] consortium featuring ontology as "the definition of terms used to describe and represent an area of knowledge", as well as more complex definitions, such as proposed by Guarino [4] and Smith and Welty [5], who consider classes, properties, instances, axioms and logic to build ontological structures[2].

[1] http://www.w3.org/TR/2003/ WD-webont-req-20030203

[2] We will use the terms "ontology" and "ontological structure" interchangeably, and we will adopt, to ontology, an open approach.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 11–20, 2010.

Krötzsch et al. [6] introduce the concept of class in ontologies with "Classes can be compared to Wikipedia's categories: they describe collections of objects and can be organized in a hierarchy". For instance, actor is a subclass of person. As in Wikipedia, the multiple inheritance and even cycles are allowed in the class hierarchy.

Building ontologies is a costly, tedious and error-prone process [7], and the number of domain ontologies currently available remains extremely small [8], scenario that is even worse in Portuguese language [9]. One alternative option for ontologies extraction is to use Wikipedia as data source.

In this context, we present a method for extracting domain ontologies in Portuguese language from Wikipedia's category structure. In order to validate this method we have developed a case study. We have built a prototype that extracts a Tourism ontology containing classes, instances and relations (is-a, located-in) from the Wikipedia database in Portuguese. The results were evaluated by comparing the obtained ontology with a Golden Standard manually built from Wikipedia's Tourism category. These results (79.51% Precision and 91.95% Recall) were promising, demonstrating the feasibility of the proposed method.

The article is organized as follows: Section 2 reviews related work on ontology extraction from Wikipedia. In Section 3 we present our method for domain ontology extraction. The case study, its evaluation and results are presented in Section 4. We end the article with a brief conclusion.

2 Related Work

Several studies, as [1, 2, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] support our research on ontologies extraction from Wikipedia, especially regarding the feasibility of this task. In this section, we present a brief review of those which provided relevant ideas for our method.

YAGO: A Large Ontology from Wikipedia and WordNet [16] presents an ontology derived from Wikipedia and WordNet. The ontology is populated by facts derived from Wikipedia's category system and infoboxes[3], combined with taxonomic relationships from WordNet. This is made in two stages: first, different heuristics are applied to Wikipedia's data in order to extract facts and candidate entities, and the connection between Wikipedia and WordNet is established. The second stage is the application of quality control techniques. The results were evaluated by human judges, who found that 74 heuristics had precision greater than 95%.

Ponzetto and Strube, in [13], describe an experiment for automatic creation of a taxonomy from Wikipedia's category system. Further articles by Ponzetto and Strube [15]; Zirn, Nastase and Strube [18]; and Nastase and Strube [19] describe methods to automatically distinguish between classes and instances from the taxonomy generated in [13].

Wikipedia's category structure seems to be an excellent source of data for ontologies extraction, since it contains relationships between concepts. In the work of Strube and Ponzetto [12, 13] categories are used to describe concepts. [18, 19] report more

[3] Infobox is a Wikipedia standard model with basic information about the entity described in the article.

detailed analysis of category titles, describing the extraction of classes and instances. [13] describes very clearly the steps for identifying is-a and not-is-a relations in the category structure, however, it is not clear how the lexical-syntactic patterns were used.

Data extraction from infoboxes and WordNet seems to be promising in English language. However, currently, this feature does not appear to be useful in Portuguese for two reasons: infoboxes are little used in Wikipedia's Portuguese version and the available versions of WordNet in Portuguese language does not have a significant amount of content.

The reviewed papers reported experiments based on the full content of the encyclopedia and not on a specific category. Furthermore, the reviewed studies performed the extraction of content in English only. The performance of a work that extracts an ontology from the Portuguese version of Wikipedia should take in consideration the characteristics of this release, such as size, less use of models and the different spellings of the same term, given the inherent differences between Brazilian, African, Asian and European Portuguese.

3 Extraction Method

This section presents our semi-automatic method for extracting domain ontologies from Portuguese Wikipedia's category system.

The input to the extraction is Wikipedia's database, particularly the tables with information about categories. After applying the method, we obtain a domain ontology described in OWL, containing classes, instances and relations. In the following subsections we describe the stages of the semi-automatic extraction method illustrated in Figure 1.

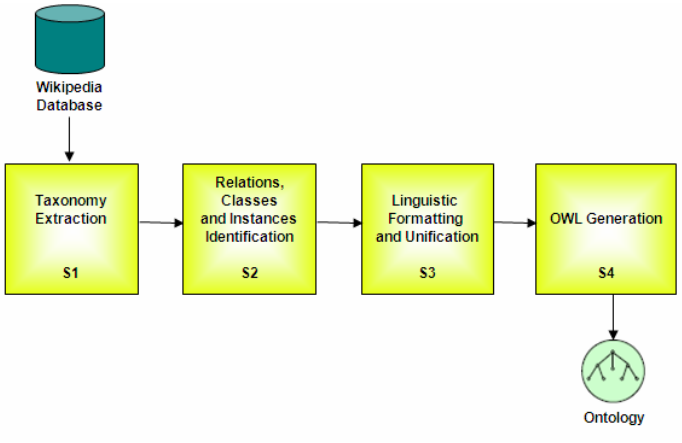


Fig. 1. The method input is Wikipedia's database. It generates a domain ontology containing classes, instances and relations.

3.1 Stage 1 - Taxonomy Extraction

The goal is to obtain a taxonomy where concepts are category titles and the hierarchical relation between concepts is established by the category structure organization in the database. The input here is Wikipedia's database (particularly the tables containing data related to the category structure), the category used as source for extraction and the depth of the query. The output is a taxonomy with the category structure.

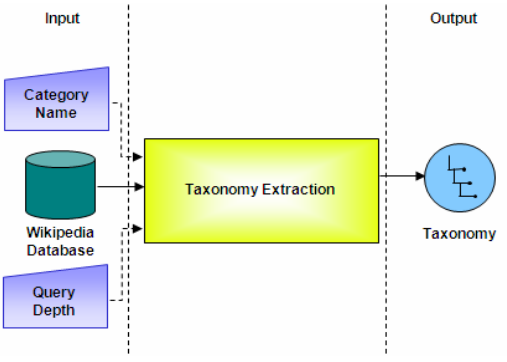


Fig. 2. First Stage (S1) generates a taxonomy where the hierarchical relation is established by the category structure in Wikipedia

Wikipedia covers different domains of knowledge and the organization of its categories enables the connection of concepts that belong to more than one domain. Since our goal is to obtain a domain ontology, we must limit the selection of subcategories to a limited depth, in order to obtain the largest number of concepts but also ensuring that the selected categories belong to the original domain.

The result is the selection of the subcategories of the chosen category. We emphasize that the number of levels to be searched in the category tree depends on the characteristics of the category to be used as the basis for extraction. Therefore, this number must be previously defined by the ontology engineer.

3.2 Stage 2 - Relations, Classes and Instances Identification

This stage (S2) performs the extraction of relations, new classes and instances, through the analysis of the concepts in the taxonomy generated in the previous step (see Figure 3).

An ontology composed by classes, instances and relations results from this stage. To reach this structure it is necessary previously: to define the relations to be extracted; to define the heuristics to be used to extract relations and to distinguish between classes and instances.

In this stage, each concept is analyzed, checking if there is, embedded in it, a semantic relation other than hyponymy. In this case, the is-a relation is replaced with a new one; and we check if the class name remains the same, or if it should be replaced by a new class and instance.

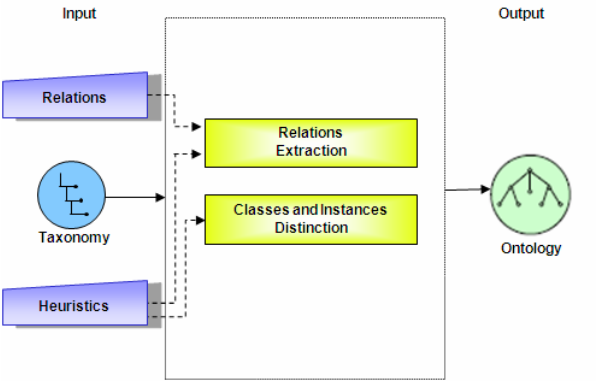


Fig. 3. Second Stage (S2) extracts relations, new classes and instances from the taxonomy generated in S1

The activities in this stage were based on the Category Heuristics reported in [16] and on methods described in [15, 18, 19] to automatically distinguish between classes and instances in a taxonomy.

3.2.1 Relations Extraction

According to Gruber [20], relations are a set of tuples that represent a relationship between objects in a universe of discourse.

Strube and Ponzetto claim that Wikipedia's category structure does not constitute a taxonomy with a well-formed subsumption hierarchy, but a thematically organized thesaurus [13]. The exclusive use of hyponymy relations (is-a) does not reflect the semantic relationship between concepts in the categories taxonomy extracted from Wikipedia. The use of other relations in conjunction with is-a is essential to accurately describe the semantic connections between concepts. For example, the category of “Capital in Asia” is registered under Capital (is-a), but the category Philosophy is under the category “Abstraction and Belief” (deals-with) and also under “Humanities” (is-a) [13].

The choice for which relations should be extracted is directly related to the domain represented in the ontology. It is first necessary to examine the taxonomy obtained in the previous step to then define, from its concepts, which relations represent the semantic relationships between classes. For example, in the Tourism taxonomy, some categories have location relations embedded in their title, such as "Zoos in Germany", which can be represented by the relationship "Zoos located-in Germany".

3.2.2 Distinction between Classes and Instances

Instances represent the objects of the domain on which is our interest, while classes are interpreted as sets of instances [21]. To characterize an instance, we suggest the use of these characteristics [22]: instances are proper nouns, which means that they should be capitalized and instances are unique, which implies that they should not have hyponyms; it is meaningless to have an instance of an instance.

At this point, the concepts selected in the previous stage, as having a different relationship than hyponymy, are analyzed in order to check if it is necessary to create new

classes and instances. For example, we have identified in th category "Zoos in Germany" the relationship "Zoos located-in Germany". From this point, we characterize "Zoos" as a new class and "Germany" as an instance.

The heuristics used to analyze the category title should be manually defined in advance, because they depend on: domain, relations to be extracted and the category structure used as source.

3.3 Stage 3 - Formatting and Linguistic Unification

In this stage (S3) the ontology names are standardized. As a first task we format titles, so they can be represented in OWL. This task is performed in three steps:

1. Remove special characters;
2. Replace spaces with underscore;
3. Convert all characters to lowercase.

The second task unifies different spellings, considering that Wikipedia's Portuguese version, according to its manual of style[4], does not use a specific version of the common language, regardless their country of origin. Thus, the same term may be registered more than once with different spellings, with the differences inherent to all variants of Portuguese.

3.4 Stage 4 – OWL Generation

The last stage (S4) generates the OWL description of the ontology obtained in previous stages. The OWL description allows the visualization and refinement of the extracted ontology in ontology editors such as Protégé[5], as well as easy access by other applications.

4 Case Study

In this section we describe a case study developed to validate the method presented in the previous section. We have conducted an experiment extracting a Tourism ontology containing classes, instances and relations (is-a, located-in) from Portuguese Wikipedia's Tourism category[6]. The generated ontology was confronted against a Golden Standard and the metrics Precision and Recall were calculated.

4.1 Prototype

The prototype developed to conduct the case study was implemented in PHP, accessing Portuguese Wikipedia's database in Mysql[7] and generating an OWL file. The prototype was designed in four steps, according to the method being validated. Next we describe the prototype steps.

---

[4] http://pt.wikipedia.org/wiki/Wikipedia:Livro_de_estilo
[5] http://protege.stanford.edu/
[6] http://pt.wikipedia.org/wiki/Categoria:Turismo
[7] Database dump obtained in http://download.wikimedia.org/backup-index.html from Wikipedia's Portuguese version of January 05, 2009.

Step 1 – Categories Selection. In order to select the depth of the Tourism category query, we have manually performed an analysis of its subcategories graph and from this observation, we decided to set the search in three levels of subcategories, trying to get as many concepts as possible without exceeding the Tourism domain.

Upon completion of this step, we get a taxonomy where the concepts are the categories titles and the hierarchical relationship is established by how the category structure was organized in Wikipedia's database.

Step 2 – Relations, Classes and Instances Identification. To accomplish this task we have previously defined the relations and instances to be extracted and the heuristics to be used. We have noticed that many of these semantic relations would be better represented by the located-in relation. For example, the two categories with the largest number of subcategories are "Transportation by country"[8] and "Tourism by country"[9], which are categories whose semantic content is related to location. In addition, some categories present located-in relationship embedded in their title, for example, "Spas in Brazil"[10] which in our view includes the relationship "Spa" located-in "Brazil".

From this definition, we found that located-in relations did not occur only among classes, but between class and instance of a place. In the relation "Spa located-in Brazil", we classify "Spa" as class and "Brazil" as instance. To accomplish the task of identifying location relationships and to make the distinction between classes and instances we have proposed four heuristics:

Heuristic 1: infers the existence of located-in relationships in the subcategories of categories whose title contains the words "country", "city", "province" or "state". For example, "Touristic attractions in Curitiba"[11] is subclass of "Tourist attractions by city"[12]. The application of the rule generates the instance "Curitiba" and the relation "Touristic attractions by city" located-in "Curitiba" and, also erases the class "Touristic attractions in Curitiba".

Heuristic 2: infers located-in relationships in categories containing certain prepositions[13] in its title. For example, from "Airports in Argentina"[14] the application of this rule generates the class "Airports", the instance "Argentina", the relation "Airports" located-in "Argentina" and erases the class "Airports in Argentina".

Heuristic 3: for classes containing only one lexical item in their title, search in Wikipedia's database for connections between the correspondent category and other categories related with locations. If the connection is found, the class is transformed in instance and a located-in relation is established with its super class. I.e, "Krakow"[15] is subclass of "UNESCO World Heritage"[16] and "Cities of Poland"[17]: this creates the relation "UNESCO World Heritage" located-in "Krakow".

[8] "Transporte por País" in Portuguese.
[9] "Turismo por País" in Portuguese.
[10] "Termas do Brasil" in Portuguese.
[11] "Atrações turísticas de Curitiba" in Portuguese.
[12] "Atrações turísticas por cidade" in Portuguese.
[13] In Portuguese: "de/do/da" and "em/no/na".
[14] "Aeroportos da Argentina" in Portuguese.
[15] "Cracóvia" in Portuguese.
[16] "Patrimônio Mundial da UNESCO" in Portuguese.
[17] "Cidades da Polônia" in Portuguese.

Heuristic 4: performs a quality control by excluding wrong mappings. If an instance was also mapped as a class, the mapping as instance is eliminated.

Step 3 – Spelling Unification. At this stage, we standardize classes and instances names, allowing the OWL creation in the next step.

To do so, we perform a function that replaces string "cç" (Portugal's Portuguese) spelling for "ç"; we remove words accents, ie. replacing "ã" with "a"; we convert uppercase to lowercase; we replace blanks by underscore.

Step 4 – OWL Generation. Finally, the last stage generates an OWL file containing the extracted ontology description.

The ontology created with the prototype execution consists of 165 classes and 156 instances.

4.2 Evaluation

In order to evaluate the results obtained with the case study reported in the previous session, we compute Precision and Recall. From these, we investigate some of the causes of successes and mistakes of the prototype, examining the similarities and differences between the Golden Standard and the ontology being evaluated.

The Golden Standard used to calculate the metrics was manually constructed from the Tourism Category Network, revised and refined by a linguist. The Golden Standard was developed following three steps:

1. Tourism Category Structure export in three levels into a taxonomy in OWL.
2. Manual construction of the ontology, from the taxonomy generated in the previous step.
3. Review and refinement of the ontology by a linguist.

Confronting both ontologies as a whole, we have obtained 79.51% Precision and 91.95% Recall. These results are satisfactory and comparable to those found in the literature for other languages.

The main differences between the ontology generated by the prototype and the Golden Standard are in the is-a relation mapping. We have achieved for this analysis 73.03% Precision and 91.98% Recall. The main cause of differences was produced by the Heuristic 2.

For example, in the Golden Standard the class "hotels"[18] is a subclass of "lodging_facilities"[19], while in the extracted ontology the class "hotels" is also a subclass of "tourism_in_south_america"[20]. This happened because applying the Heuristic 2 in the class "hotels_from_brazil"[21]: generates the class "hotels"; generates the instance "Brazil"; generates the relation "hotels" located-in "Brazil"; places the class "hotels" as subclass of "tourism_in_south_america" (original position of the class "hotels_from_brazil").

The best results were achieved by the instance mapping, obtaining 99.35% Precision and 94.44% Recall. Only one instance was wrongly mapped by prototype:

[18] "hotéis" in Portuguese.
[19] "meios_de_hospedagem" in Portuguese.
[20] "turismo_na_america_do_sul" in Portuguese.
[21] "hoteis_do_brasil" in Portuguese.

"superhighway"[22]. Heuristic 3 caused this failure, since "superhighway" contains only one word in its title and is connected to the "Rio de Janeiro City Transports"[23].

5 Conclusion

We have presented a method for semi-automatic extraction of domain ontologies from Portuguese Wikipedia. Toward this end we exploited Wikipedia's category structure and the categories names as source for ontology components extraction. The method input is Wikipedia's database, particularly the tables containing information about categories. After applying its four stages, it generates a domain ontology described in OWL.

The method was validated through a case study that produced a Tourism ontology. In order to evaluate this ontology, a Golden Standard was manually constructed from the Tourism Category structure, revised and refined by a linguist. The results were satisfactory: 79.51% Precision and 91.95% Recall.

This evaluation showed that the main differences between the ontology generated by the prototype and the reference ontology are in the mapping of is-a relations. One of the reasons that led to this inequality was the exclusion of repeated classes in the Golden Standard by the human revisor. The prototype had its best performance in the instance mapping.

From this experience, we point that the appropriate definition of heuristics is the key point for implementing the method successfully.

We believe that the use of a Golden Standard is suitable and widely used in this type of problem. However, we are working to find other alternatives for evaluation. For example, it would be interesting to perform extrinsic evaluation using information retrieval systems.

Future work includes the method improvement, seeking its automation. To accomplish this, we point to the automation of the search depth obtention in the method first stage and the use of other components of the encyclopedia as text and links between articles as data source.

References

1. Zareen, S., et al.: Wikipedia as an Ontology for Describing Documents. In: Proceedings of the Second International Conference on Weblogs and Social Media (2008)
2. Wu, F., Weld, D.S.: Autonomously semantifying wikipedia. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM 2007), pp. 41–50. ACM, New York (2007)
3. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5(2), 199–220 (1993)
4. Guarino, N.: Formal Ontology. In: Proceedings of the 1st International Conference on Information Systems. IOS Press, Trento (1998)
5. Smith, B., Welty, C.: FOIS introduction: Ontology- towards a new synthesis. In: Proceedings of the International Conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, October 17-19. ACM, New York (2001)

[22] "supervia" in Portuguese.
[23] "Transportes da cidade do Rio de Janeiro" in Portuguese.

6. Krötzsch, M., Cić, D., Völkel, M.: Wikipedia and semantic web - the missing links (2005)
7. Maedche, A.D.: Ontology Learning for the Semantic Web. Kluwer Academic Publishers, Dordrecht (2002)
8. Hepp, M., Bachlechner, D., Siorpaer, K.: Harvesting Wiki Consensus - Using Wikipedia Entries as Ontology Elements. IEEE Internet Computing 11(5), 54–65 (2007)
9. Lima, V., Nunes, M., Vieira, R.: Desafios do Processamento de Línguas Naturais. In: SEMISH - XXXIV Seminário Integrado de Software e Hardware, Anais do XXVII Congresso da SBC, Rio de Janeiro (2007)
10. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic Wikipédia. In: Proceedings of the 15th International Conference on World Wide Web, pp. 585–594 (2006)
11. Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: Proc. of the 17th Int. Conf. on WWW, pp. 635–644. ACM, New York (2008)
12. Ponzetto, S., Strube, M.: Knowledge Derived from Wikipedia for Computing Semantic Relatedness. Journal of Artificial Intelligence Research 30, 181–212 (2007)
13. Ponzetto, S.P., Strube, M.: Deriving a large scale taxonomy from Wikipedia. In: Proceedings of the 22nd National Conference on Artificial Intelligence, vol. 2, pp. 1440–1445. AAAI Press, Menlo Park (2007)
14. Nastase, V., Strube, M.: Decoding Wikipedia Categories for Knowledge Acquisition. In: Twenty-Third AAAI Conference on Artificial Intelligence, pp.1219–1224 (2008)
15. Ponzetto, S.P., Strube, M.: WikiTaxonomy: A Large Scale Knowledge Resource. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N. (eds.) Proceeding of the 2008 Conference on ECAI 2008. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 751–752. IOS Press, Amsterdam (2008)
16. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Large Ontology from Wikipedia and WordNet. Web Semant. 6(3), 203–217 (2008)
17. Syed, Z., Finin, T., Joshi, A.: Wikipedia as an Ontology for Describing Documents. In: Proceedings of the International Conference on Weblogs and Social Media (2008)
18. Zirn, C., Nastase, V., Strube, M.: Distinguishing between Instances and Classes in the Wikipedia Taxonomy. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 376–387. Springer, Heidelberg (2008)
19. Nastase, V., Strube, M.: Decoding Wikipedia Categories for Knowledge Acquisition. In: AAAI 2008, pp. 1219–1224 (2008)
20. Gruber, T.R.: Ontolingua: A Mechanism to Support Portable Ontologies. Technical Report (1992)
21. Horridge, C., Knublauch, H., Rector, A., Stevens, R., Wroe, C.: A practical guide to building OWL Ontologies using the Protégé OWL Plug-in and CO-ODE Tools (2004), http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf
22. Miller, G.A., Hristea, F.: WordNet nouns: Classes and instances. Computational Linguistics 32(1), 1–3 (2006)

Ontology Reasoning in Agent-Oriented Programming

Claudio Fuzitaki[1,⋆], Álvaro Moreira[1], and Renata Vieira[2]

[1] Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
{alvaro.moreira,fuzitaki}@inf.ufrgs.br
[2] Faculty of Informatics, Pontifical Catholic University of Rio Grande do Sul, Brazil
renata.vieira@pucrs.br

Abstract. DL-Lite is being regarded as an effective logic for ontology reasoning due both to its expressive power and its computational properties. Considering that ontologies are important constructs for multi-agent system, in this paper we propose the integration of ontology reasoning and agent-oriented programming. More specifically, we consider an agent-oriented programming language based on DL-Lite with belief bases consisting of an immutable TBox, with the characterization of concepts and roles, and of an ABox with factual knowledge, which can change as the result of perception of the environment, internal actions, and inter-agent communication. We discuss the benefits of ontological reasoning and we give algorithms for belief base querying, plan selection, and for a principled approach for belief base update. The language we propose, AgentSpeak-DL, is a subset of AgentSpeak, a well known BDI multi-agent oriented programming language.

1 Introduction

Although the semantic web research community acknowledges the key role of autonomous agents in the development of semantic web applications, few current agent-oriented programming languages are incorporating support for ontology-based reasoning in their languages.

Description Logics [1] are at the core of widely known ontology languages, such as the Ontology Web Language (OWL) [2]. The DL-Lite family is a family of Description Logics that is capable of capturing the main notions of conceptual modeling and ontology description, while keeping reasoning tasks tractable [3].

In this paper we propose the core of a logic agent-oriented programming language based on DL-Lite. In this language, the belief base of an agent consists of a TBox, with the characterization of concepts and roles, and of an ABox, with factual knowledge. We argue that this DL-Lite based language compares favorable with its predicate logic-based counterpart, not only because it provides the means for a more structured knowledge representation in the belief base, but also because it offers more powerful reasoning tasks both for belief base querying and for plan selection.

Beliefs can change as the result of perception of the environment, internal actions, and inter-agent communication. Hence, another concern of the agent-oriented programming language community is to equip its languages with principled belief revision

⋆ Bolsista do CNPq - Brasil.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 21–30, 2010.

mechanisms [4]. In this context, we define for our language a mechanism for belief revision based on recent work proposing DL-Lite ontology update and removal algorithms [5].

The programming language we are presenting, called AgentSpeak-DL, is based on predicate-logic AgentSpeak, a BDI based language with a formal semantics, support for inter-agent communication [6], and with a fully functional programming environment [7]

It is important to notice, however, that plan selection and goal testing are present in a similar way, in other agent oriented languages. Hence, we believe that the ideas brought here have a general appeal for agent-oriented programming as a paradigm.

The paper is organized as follows: Section 2 discusses related work. In Section 3 we present DL-Lite [3], an expressive Description Logic with polynomial reasoning tasks. In Section 4 we present the AgentSpeak-DL language giving the advantages of DL-Lite ontology reasoning for programming, and also discussing the ideas behind a principled belief revision mechanism. In Section 5 we give algorithms for belief querying, plans selection and for belief update and removal.

2 Related Work

In [8] a proposal for a BDI like programming language based on Description Logic was presented. That work sketched the main ideas on having ontological reasoning in an agent programming language. It didn't deal properly with ontology update. Besides, no algorithms were given.

Building on the ideas of [8], the work of [9] described the implementation of ontological reasoning tasks for Jason [7], a framework for developing multiagent programs in AgentSpeak. It uses a bridge approach: the knowledge base continues to be based on predicate logic, but some annotated rules and facts can be sent to an external DL reasoner.

A series of negative results regarding the complexity of the update operation for various description logics is given in [10]. Working with a simple and efficient description logic, such as DL-Lite is thus justified since the DL-Lite family [3] keeps reasoning tasks tractable and it is also expressive enough to capture the main notions of ontology description.

The association of DL-Lite with agent programming has also been proposed in [11], which discusses practical issues concerning the use of a simple ontology framework inside a reactive, rational agent. That work also outlines the implementation of a concrete approach for handling dynamic beliefs about individuals. The authors use a justification based truth maintenance system to efficiently maintain ontological consistency of the belief store when it is modified.

In our paper a stronger connection with a BDI like programming language is presented. We address ontological reasoning, providing algorithms for belief base querying, plan selection, and belief update and removal. The language we propose is an extension of predicate-logic AgentSpeak, but being it closely related to current available agent-oriented programming languages, we consider that our discussions can be considered useful for the agent-oriented programming language community.

3 The Description Logic DL-Lite

Description Logics constitute a family of logic languages for knowledge representation and reasoning differing in their expressive power and in the computational properties of their reasoning tasks. The representation of a knowledge base in a Description Logic consists of two parts: a TBox, which contains the intensional description of the domain of interest, and an ABox with extensional information.

A Description Logic characterization of a domain of interest (TBox) starts with a choice of *primitive atomic concepts*, designated by unary predicate symbols, and a choice of *primitive atomic roles*, designated by binary predicate symbols.

The specific Description Logic that we consider is this paper is a member of the DL-Lite family [3]. More precisely it corresponds to the DL-Lite$_{\mathcal{F}}$ logic introduced in [5]. Contrary to the majority of expressive description logics, DL-Lite reasoning tasks are tractable.

The syntax of DL-Lite$_{\mathcal{F}}$ concepts is given as follows, where A is a metavariable for primitive atomic concepts, R represents a primitive atomic role, and B and C are basic and general concepts, respectively:

$$B ::= A \mid \exists R \mid \exists R^{-} \qquad C ::= B \mid \neg B \mid C_1 \sqcap C_2$$

From the grammar above observe that the only way to construct a new role is to apply the inverse constructor to primitive atomic roles. Observe also that negation can only be applied to basic concepts.

DL-Lite TBox assertions have the following form:

$$B_1 \sqsubseteq B_2 \qquad B_1 \sqsubseteq \neg B_2 \qquad B \sqsubseteq C_1 \sqcap C_2 \qquad funct\ R$$

Assertion $B_1 \sqsubseteq B_2$ expresses that instances of the concept B_1 are also instances of concept B_2; assertion $B_1 \sqsubseteq \neg B_2$ says that the sets denoted by B_1 and B_2 are disjoint, and $B \sqsubseteq C_1 \sqcap C_2$ expresses that instances of the concept B belong to the set formed by the intersection of the denotations of concepts C_1 and C_2. Assertions of the form $funct\ R$ define that the binary relation R is a function. Assertions of the form $B \sqsubseteq C_1 \sqcap C_2$, although convenient, do not add to the expressive power of the logic and they can be easily replaced by the assertions $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$.

The assertions allowed in a DL-Lite ABox have the form: $B(a)$ or $R(a,b)$ where a and b are constants denoting objects of the domain of interest. An assertion $B(a)$ states that the object a is an instance of the concept B, and $R(a,b)$ states that the pair of objects (a,b) is an instance of the role R. Observe that B can be a concept $\exists R$. An assertion $\exists R(a)$ means that a belongs to the set of objects that come first in the binary relation R.

4 The AgentSpeak-DL Language

A AgentSpeak-DL program consists of a set of initial beliefs followed by a set of plans. A plan is formed by a *triggering event*, followed by a conjunction of belief literals representing a *context*, and a *body*:

$$triggering\_event : context \leftarrow body$$

The plan *body* is a sequence of calls of subplans (predicates prefixed with "!"), queries to the belief base (predicates prefixed with "?"), and belief addition and removals (predicates prefixed with "+" and "-", respectively).

Agents have to respond to the occurrence of *events*, and plans are sequences of actions that agents have to execute when dealing with a related event. Events are represented by the addition of corresponding predicates in a set of events. The runtime environment of an agent-oriented language checks this set of events, looking for a new event to be treated, and tries to find *relevant* plans for it.

The main difference from a predicate logic-based agent oriented language and the description logic-based language we propose here is that the belief base of an agent, instead of consisting of ground literals, consists of a TBox and of an Abox expressed in the DL-Lite given in the previous section. The main advantage in relation to predicate logic AgentSpeak is that the reasoning capabilities are more powerful and do not rely only on unification.

As an example of a simple AgentSpeak-DL program we consider a conference scheduler agent in the domain of conference speakers. In Fig. 1 we have the TBox and ABox which constitute the belief base of the scheduler agent. The intensional description is characterized by TBox assertions.

[1] $TBox$: $\exists \mathsf{willPresent} \sqsubseteq \mathsf{availablePresenter}$
[2] $\mathsf{late} \sqsubseteq \neg \mathsf{availablePresenter}$
[3] $\mathsf{availablePresenter} \sqsubseteq \mathsf{presenter}$
[4] $\mathsf{paperPresenter} \sqsubseteq \mathsf{presenter}$
[5] $\mathsf{invitedSpeaker} \sqsubseteq \mathsf{presenter}$
[6] $ABox$: $\mathsf{willPresent}(\mathsf{john}, \mathsf{paper})$

Fig. 1. Example of AgentSpeak-DL belief base of scheduler agent

According to these assertions we have that: someone who is scheduled to make a presentation is an available presenter (line 1); someone who is late is not an available presenter (line 2); and that available presenters, paper presenters, and invited speakers are all presenters (lines 3, 4 and 5). The ABox simply says that John will present a paper (line 6).

From the ABox and from the TBox given in Fig. 1 we can infer, for instance, that $\mathsf{availablePresenter}(\mathsf{john})$ and hence that $\mathsf{presenter}(\mathsf{john})$, and also that $\neg\mathsf{late}(\mathsf{john})$.

In Fig. 2, we give examples of plans to handle a "next presenter" event. The first plan says that, if a presenter of a paper is late, s/he is rescheduled to the end of the session (and the session continues with the next scheduled speaker). If an invited speaker is late, apologies are given to the audience and the speaker is announced (this event only happens when the invited speaker actually arrives, assuming the paper sessions must not begin before the invited talk).The last two plans announce paper presenters and invited speakers in case they are not late.

```
+paperPresenter(X)  : late(X)   ← !reschedule(X).
+invitedSpeaker(X)  : late(X)   ← !apologise; !announce(X).
+paperPresenter(X)  : ¬late(X)  ← !announce(X).
+invitedSpeaker(X)  : ¬late(X)  ← !announce(X).
```

Fig. 2. Example of AgentSpeak-DL plans for scheduler agent

The reasoning capabilities of DL allows agents to infer knowledge that is implicit in the belief base. As an example, we can remove the last two plans given in the example of Fig. 2, replacing them by a single plan with presenter as triggering event:

```
+presenter(X): ¬late(X)  ← !announce(X).
```

This new plan is more general than the two plans removed since it can deal both with invited speakers and paper presenters.

5 Ontological Reasoning in AgentSpeak-DL

In the previous section we saw the benefits of ontological reasoning for belief base testing and for plan selection. In this section, we present how these activities can be performed when interpreting AgentSpeak-DL programs, more specifically, we give algorithms for the following tasks:

- *Querying the belief base* - given the set bs of beliefs of an AgentSpeak-DL agent, and a test goal $?at$, the function $\mathsf{Test}(bs,at)$ returns a set with all substitutions θ such that θat can be derived from the ontology in the agent's beliefs, where θat is the predicate that results from the application of substitution θ to at.
- *Collecting relevant plans* - given the plans and the beliefs of an agent ag, and a triggering event te, the function $\mathsf{RelPlans}(ag,te)$, returns all plans $te' : ct \leftarrow h$ such that $te' \sqsubseteq te$ can be derived from the ontology in the belief base of the agent.
- *Belief addition* - given the belief base bs, and a ground predicate at, the function $\mathsf{Update}(bs,at)$ returns both the set of beliefs to be removed from, and the set of beliefs to be added to the belief base in order to accommodate at in consistent way.
- *Belief removal* - the inputs for the function $\mathsf{Remove}(bs,at)$ are a belief base bs, and a predicate at possible containing variables. The function also returns a set of beliefs to be added, and a set to be removed from the belief base bs. These sets of beliefs is determined based on the result of querying the belief base with new at.

We start with a normalization procedure [3], that adds to the belief base implicit knowledge that can be easily made explicit, and with a procedure to check whether a normalized belief base is consistent.

Normalization. The normalization process modifies a belief base $\mathscr{K} = (\mathscr{T}, \mathscr{A})$ as follows: (1) for each $R(a,b) \in \mathscr{A}$, the ABox $\mathscr{A}$ is expanded by adding to it assertions $\exists R(a)$ and $\exists R^{-}(b)$ (this process adds to the belief base, knowledge that is implicit in

$R(a,b)$); (2) the TBox is closed with respect to the following rule: if $B_1 \sqsubseteq B_2$ is in $\mathscr{T}$, and either $B_2 \sqsubseteq \neg B_3$ or $B_3 \sqsubseteq \neg B_2$ is in $\mathscr{T}$ then add $B_1 \sqsubseteq \neg B_3$ to $\mathscr{T}$. After this closure, to decide whether B_1 and B_2 are disjoint concepts, it suffices to look for $B_1 \sqsubseteq \neg B_2$ or for $B_2 \sqsubseteq \neg B_1$ in $\mathscr{T}$.

Consistency. Given a normalized belief base $\mathscr{K} = (\mathscr{T}, \mathscr{A})$ a procedure for verifying consistency checks the following conditions: (1) there is an assertion $B_1 \sqsubseteq \neg B_2$ in $\mathscr{T}$, and a constant a such that both $B_1(a)$ and $B_2(a)$ are in $\mathscr{A}$; (2) there is assertion $funct R$ in $\mathscr{T}$, and constants a, b, and c, such that both $R(a,b)$ and $R(a,c)$ are in $\mathscr{A}$. If at least one of the conditions above is true, the belief base is inconsistent. If none of then holds, $\mathscr{K}$ is consistent. We can see that the only sources of inconsistency in DL-Lite are violations of disjointness and functionality properties.

From now on we assume that the input for our algorithms are taken from normalized and consistent belief bases.

5.1 Querying the Belief Base

In AgentSpeak-DL queries for the belief base are performed by evaluating test goals of the form $?B(t)$ and $?R(t_1,t_2)$, where B is a basic concept, R is a binary role, and t, t_1 and t_2 are constants or variables.

Querying a belief base with $?B(x)$, when the constants $a_1, \ldots a_n$ denote the objects that belong to the concept B, returns a set of substitutions $\{[x \mapsto a_1], \ldots [x \mapsto a_n]\}$.

Similarly, the result of the test goal $?R(x,b)$, where x is a variable and b is a constant, is the set of substitutions $\{[x \mapsto a_1], \ldots [x \mapsto a_n]\}$, where $a_1, \ldots a_n$ are such that $(a_1,b), \ldots (a_n,b)$ belong to the relation R. If $(a_1,b_1), \ldots (a_n,b_n)$ all belong to the relation R, a query like $R(x,y)$, results in $\{[x \mapsto a_1; y \mapsto b_1], \ldots [x \mapsto a_n; y \mapsto b_n]\}$

A substitution is selected from the set of substitution returned by the function Test and it is propagated to the rest of the plan. If a test goal fails and returns an empty set, an event is added to the set of events. Observe that, if a test goal with no variables succeeds, the result will be a singleton set with the empty substitution as its only member.

The issues involved in building this set of substitutions in the presence of ontological reasoning can be illustrated by the following example: suppose that we have an ABox with assertions $B_1(a)$, $B_1(b)$, $B_2(c)$, $B_2(d)$ and we want to evaluate the test goal $?B_2(x)$. In the presence of a positive inclusion assertion $B_1 \sqsubseteq B_2$ in the TBox, asserting that every instance of B_1 is also an instance of B_2, the result of this test goal has to include not only substitutions $[x \mapsto c]$ and $[x \mapsto d]$, but also substitutions $[x \mapsto a]$ and $[x \mapsto b]$, which are the substitutions produced by the evaluation of test goal $?B_1(x)$. Hence, for evaluating a given test goal in the presence of positive inclusion assertions, the first thing to do is to formulate an evaluate additional test goals.

Figure 3 shows the algorithm for the function $ref(\mathscr{T}, at)$, responsible for reformulating a test goal at according to a TBox $\mathscr{T}$. Observe that, since DL-Lite admits only unary concepts in positive inclusion assertions $B_1 \sqsubseteq B_2$, test goals like $?R(t_1,t_2)$ are not reformulated (lines 1 and 2). The function calls, in line 8, the auxiliary function $clos(\mathscr{T})$ that computes the transitive closure of the $\sqsubseteq$ relation in $\mathscr{T}$.

After query reformulation, the resulting queries are evaluated by the function *ans* given in Fig. 5. This function takes as input an ABox $\mathscr{A}$ and a predicate at, and returns

a set of substitutions θ such that θat is in $\mathscr{A}$. It is important to distinguish between returning a empty set (failure) and returning a singleton set with the empty substitution (meaning that $at \in \mathscr{A}$).

Algorithm $ref(\mathscr{T}, at)$
Input : TBox $\mathscr{T}$ and predicate at
Output : set S with queries refomulated from at
[01] **if** $at = R(t_1,t_2)$ **then**
[02] $S := \{R(t_1,t_2)\}$
[03] **else** (at is $B(t)$)
[04] $S := \{B(t)\}$
[05] **repeat**
[06] $S' = S$
[07] **for each** $B(t) \in S'$
[08] **if** $B_1 \sqsubseteq B_2 \in clos(\mathscr{T})$
[09] **then** $S := S \cup \{B_1(t)\}$
[10] **until** $S = S'$
[11] **return** S

Fig. 3. Algorithm for query reformulation

Algorithm $\mathsf{Test}(\mathscr{T}, \mathscr{A}, ?at)$
Input : belief base $(\mathscr{T}, \mathscr{A})$, test goal $?at$
Output : set of substitutions
[1] **return** $\bigcup_{q \in ref(\mathscr{T},at)} ans(\mathscr{A}, q)$

Fig. 4. Algorithm for Test

Algorithm $ans(at, \mathscr{A})$
Input : ABox $\mathscr{A}$ and predicate at
Output : set S of substitutions
[01] $S := \{\}$
[02] **if** $at = B(x)$ **then**
[03] **for each** $q \in \mathscr{A}$
[04] **if** $q = B(c)$
[05] for a constant c
[06] **then** $S := S \cup \{[x \mapsto c]\}$
[07] **else if** $at = B(a)$ **and** $B(a) \in \mathscr{A}$
[08] **then** $S := \{[\,]\}$
[09] **else if** $at = R(x,b)$
[10] ($at = R(a,x)$) **then**
[11] **for each** $q \in \mathscr{A}$
[12] **if** $q = R(a,b)$
[13] **then** $S := S \cup \{[x \mapsto a]\}$
[14] ($\{[x \mapsto b]\}$)
[15] **else if** $at = R(x,y)$ **then**
[16] **for each** $q \in \mathscr{A}$
[17] **if** $q = R(a,b)$ **then**
[18] $S := S \cup \{[x \mapsto a; y \mapsto b]\}$
[19] **else if** $at = R(a,b)$ **and** $R(a,b) \in \mathscr{A}$
[20] **then** $S := \{[\,]\}$
[21] **return** S

Fig. 5. Algorithm for instance cheking

Finally, the algorithm for Test is given in Fig. 4. This algorithm calls *ref* which returns the set of all test goals resulting from the reformulation of the initial test goal *at*, calls *ans* to obtain sets of substitutions and acumulates all of them.

5.2 Plan Selection

In predicate logic-based AgentSpeaK, looking for plans that are relevant for an event $+B(t)$ amounts to go through the plan library collecting plans with triggering event $+B(t')$ such that t and t' *unify*. A plan like $+B(x) : ct \leftarrow h$, for instance is relevant for an event $!B(a)$ and we can say that a substitution $[x \mapsto a]$ is a witness for this fact.

In the presence of positive inclusion assertions in the agent's TBox, plan selection can potentially produce a larger set of plans. Consider for instance, the plans $+B_1(x) : ct \leftarrow h$ and $+B_3(y) : ct \leftarrow h$; and consider also that AgentSpeak-DL is looking for a plan relevant for event $+B_1(a)$ in the presence of a TBox like $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq B_3$.

According to this TBox, concept B_2 is more general than concept B_1, hence, a plan for events B_2 would be *safe* for dealing with events B_1 (and plan for event B_3 too). In other words, the search for plans that are relevant for event $B_1(t)$, in the context of a positive inclusion $B_1 \sqsubseteq B_2$, is reformulated to include the search for plans for event $B_2(t)$ (which are relevant for event B_1), and so on. In Fig. 6 we present an algorithm building new events.

Only unary predicates can appear in positive inclusion assertions, hence, just unary predicates are subject to reformulation, which is performed by the function **Events**.

Algorithm **Events**$(\mathscr{T}, at)$
Input : TBox $\mathscr{T}$, and events at
Output : set S with events

[01] **if** $at = R(t_1,t_2)$ **then**
[02] $S := \{R(t_1,t_2)\}$
[03] **else** (at is $B(t)$)
[04] $S := \{B(t)\}$
[05] **repeat**
[06] $S' = S$
[07] **for each** $B(t) \in S'$
[08] **if** $B_1 \sqsubseteq B_2 \in clos(\mathscr{T})$
[09] **s.t.** $B_1 = B$ **then**
[10] $S := S \cup \{B_2(t)\}$
[11] **until** $S = S'$
[12] **return** S

Fig. 6. Algorithm for event reformulation

Algorithm $Rel(ps, e)$
Input : plans ps and event e
Output : set S of plans paired with substitutions
[1] $S := \{\}$
[2] **for each** $te : ct \leftarrow h \in ps$
[3] **if** $\mathsf{Unify}(te, e) = R$ **then**
[4] $S := S \cup (te : ct \leftarrow h, R)$
[5] **return** S

Algorithm $\mathsf{RelPlans}((\mathscr{T}, \mathscr{A}, ps), e)$
Input : agent $(\mathscr{T}, \mathscr{A}, ps)$ and event e
Output : set of plans paired with substitution
[1] **return** $\bigcup_{te \in \mathbf{Events}(\mathscr{T}, e)} Rel(ps, te)$

Fig. 7. Collecting plans for an event

After event reformulation, plans paired with witness substitutions are produced by the first algorithm in Fig. 7. But note that now the reasoning is based on unification.

And finally the algorithm for $\mathsf{RelPlans}(ag, te)$ is given in Fig. 7.

5.3 Belief Addition and Removal

We define the function $\mathsf{Update}(ag_{bs}, b)$ (Fig. 8) which, given the belief base ag_{bs} of an agent ag, and a belief b, returns the par $(\mathsf{in}, \mathsf{out})$, where in and out are sets with beliefs that are added and removed, respectively, from the original belief base. The ABox $\mathscr{A}$ is a component of the belief base ag_{bs}.

Our algorithm calls the function $ComputeUpdate(ag_{bs}, b)$, defined in [5]. The output of this function is a new ABox $\mathscr{A}'$ with the belief b included.

The removal of a (ground) belief at from a belief base $(\mathscr{T}, \mathscr{A})$ is performed by calling the same function *ComputeUpdate* used for computing belief addition. The difference is that, besides the belief base, $\neg at$ has to be passed as argument. This is intuitive, since, for not being able to conclude at anymore, we have to make an effort to include in the belief base contradictions to at.

Algorithm Update(ag_{bs}, b)
Input: belief base ag_{bs}, belief b
Output: pair $(\mathsf{in}, \mathsf{out})$ of belief sets

[1] $\mathscr{A}' := ComputeUpdate(ag_{bs}, b)$
[2] $\mathsf{in} := \mathscr{A}' \setminus \mathscr{A}$
[3] $\mathsf{out} := \mathscr{A} \setminus \mathscr{A}'$
[4] **return** $(\mathsf{in}, \mathsf{out})$

Fig. 8. Algorithm for Update

6 Conclusions and Future Work

The development of multiagent applications that make effective use of knowledge sources is an important research issue for the development of the Semantic Web. Ontologies are key components in this project, as they can be expressed logically, allowing for sound reasoning in a specific domain.

Description logics are at the core of ontology languages. They represent a family of languages in a range of varied expressiveness power and computational properties. In this work we referred specifically to DL-Lite, which is being regarded as an effective logic for ontology reasoning due both to its expressive power and its computational properties.

On the basis of this logic, we discussed the integration of agent-oriented programming with ontological reasoning, showing that its descriptive and reasoning power can have a significant impact in agent-oriented paradigm. We share motivations and inspiration with other previous work in the area such as [8,9,11,12], but we differ from those mainly by presenting a BDI language extension while considering efficient ontology reasoning algorithms.

A short term future work is to redefine the formal operational semantics of the language using the presented algorithms with a prototype implementation.

As long term future work, we plan to integrate ontologies and speech-act-based agent communication. The area of ontology development, use and reuse is full of controversies. Some argue that, in its basic sense, an ontology must be the same and only one for a community of knowledge. The other view is that ontologies are to be freely created for particular usages. According to this second view, ontologies must be mapped, merged and agreed upon on the fly, whenever consensus is in order.

Regarding multi-agent systems, it is likely that the area will have to face these two views. There will be cases in which agents work together on the basis of a common shared ontology. In this case, agents simply share common concepts and rules (or the same TBox). Hence, the approach we introduced earlier in the paper is easily adapted for communication, since communication will mainly be related to changes in the ABox. The other case, on the other hand, is very complex. Indeed, a whole body of research on ontology mapping can be found in the literature. Such issues are in the scope of our future work.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

2. Motik, B., Hayes, P., Horricks, I.: OWL web ontology language semantic and abstract syntax, W3C recommendation (February 10, 2004), http://www.w3.org/TR/2004/REC-owl-semantics-20040210/ (Last visited in January 2009)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Veloso, M.M., Kambhampati, S. (eds.) AAAI, pp. 602–607. AAAI Press, The MIT Press (2005)
4. Alechina, N., Bordini, R.H., Hübner, J.F., Jago, M., Logan, B.: Belief revision for AgentSpeak agents. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) AAMAS, pp. 1288–1290. ACM, New York (2006)
5. Giacomo, G.D., Lenzerini, M., Poggi, A., Rosati, R.: On the approximation of instance level update and erasure in description logics. In: AAAI, pp. 403–408. AAAI Press, Menlo Park (2007)
6. Vieira, R., Moreira, Á.F., Wooldridge, M., Bordini, R.H.: On the formal semantics of speech-act based communication in an agent-oriented programming language. Journal of Artificial Intelligence Research (JAIR) 29, 221–267 (2007)
7. Bordini, R., Hubner, J., Wooldridge, M.: Programming Multi-agent Systems in AgentSpeak Using Jason. John Wiley and Sons, Chichester (2007)
8. Moreira, Á.F., Vieira, R., Bordini, R.H., Hübner, J.F.: Agent-oriented programming with underlying ontological reasoning. In: [14], pp. 155–170
9. Klapiscak, T., Bordini, R.H.: JASDL: A practical programming approach combining agent and semantic web technologies. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 91–110. Springer, Heidelberg (2009)
10. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic aboxes. In: [13], pp. 46–56
11. Clark, K.L., McCabe, F.G.: Ontology schema for an agent belief store. International Journal of Man-Machine Studies 65(7), 640–658 (2007)
12. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Actions and programs over description logic ontologies. In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 250, CEUR-WS.org (2007)
13. Doherty, P., Mylopoulos, J., Welty, C.A. (eds.): Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5. AAAI Press, Menlo Park (2006)
14. Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.): DALT 2005. LNCS (LNAI), vol. 3904. Springer, Heidelberg (2006)

System Design Modification with Actions

Maria Viviane de Menezes[1], Silvio do L. Pereira[2], and Leliane N. de Barros[1]

[1] Department of Computer Science
IME-USP, Brazil
{mariavm,leliane}@ime.usp.br
[2] Department of Information Technology
FATEC-SP\CEETEPS, Brazil
slago@pesquisador.cnpq.br

Abstract. System designers are expected to use error-detecting and correcting techniques. Although, model checking approaches have been used for verification of errors in large complex systems, they can only detect the error. the task of correcting the system design (called *model update*) is completely left to the system designer. Recent works on model update can suggest changes in the system model which do not consider domain contingencies and constraints. In this paper, we present a model update approach that can be used to automatically suggest modifications in a system based on the actions that are behind state transitions and a set of domain constraints. We claim that with this approach we can develop more realistic system error-correcting tools.

Keywords: Model Checking, Model Update, Temporal Logic Reasoning about action, Automated design.

1 Introduction

By using formal methods to specify a system behaviour, we can apply *model checking* techniques [6] to automatically detect not met requirements. In order to understand how model checking works, let us consider the well-known *microwave oven* scenario presented by [2], which represents two main microwave usage processes: food heating and cooking.

The *properties* that describe the current state of *microwave oven* system are: *started* (indicating the oven is operating), *closed* (indicating the microwave oven door is closed), *heated* (indicating the food inside oven is heated), *cooked* (describing that the food is cooked) and *error* (indicating errors detected during the system operation). Those properties are propositional atoms used to describe what is true in the system state and, their negation describe what is false. We must also consider the *actions* responsible for the changes in the system properties, i.e., how the state of the system changes with the actions. In this example, the actions are: *start*, *finish*, *open-door*, *close-door*, *warm up*, *cook* and *reset*. Figure 1 shows the formal model of a microwave system.

In the initial state s_1, two actions can happen: *start* and *close-door*. The *start* action takes the system to state s_2, which indicates the existence of an

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 31–40, 2010.

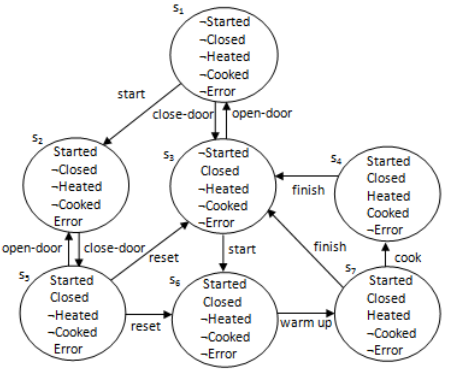


Fig. 1. Formal model of a microwave oven, adapted from [2]

error. Notice that this error will persist even if the action *close-door* is executed (transition from s_2 to s_5). In this situation, the system has to be reseted (action *reset*) which can take the system to one of the two states: s_3 and s_6, both without error (note that *reset* is a non-deterministic action: an action with uncertain effect). However, if the *close-door* action is executed in the initial state s_1, the error does not occur and the food inside microwave oven can be heated and\or cooked without having reset the oven.

Suppose that a designer wants to verify if the specification in Figure 1 satisfies the temporal formula φ defined as: "*once the microwave oven is started, the food inside will be heated in some future state*". That means, the property φ is satisfied in a system model with no state where both *started* and $\neg$*heated* properties are true. The paths $[s_1, s_2, s_5, s_3, s_1, \cdots]$ and $[s_1, s_2, s_5, s_2, \cdots]$ are examples where φ is not satisfied. A *model checker* will report one of these paths, however, the task of modifying the system model is completely left to the designer.

Model updating is a technique that extends model checking functions in order to help a system designer to repair a faulty system. To automatically suggest changes in a given model, model updater uses *primitive update operations* such as: *add a relation element*, *remove a relation element*, *change labelling function on one state*, *add a state* and *remove a state* [10]. For example, a possible correction in the model of Figure 1 is *remove the transition between* s_1 *and* s_2. This means that "*It is not allowed to start the microwave oven with the opened door*".

Motivation. In Figure 1, the state transitions are labelled with actions. However, actions are not part of a *Kripke structure*, which is the formalism used in most of the model checking and updating known approaches [1,3,10]. Moreover, those works do not consider the contingencies and constraints of the application domain. In this paper, we intend to show that representing actions in the formal model of a system, and considering the domain constraints, can allow for more realistic changes in the system model in order to effectively correct errors introduced in the specification design phase.

An example where representing actions may allow for a more rational model checking and updating is shown in Figure 2. Suppose that φ is a desired property:

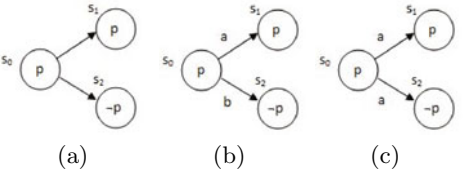


Fig. 2. (a) Kripke structure. (b) and (c) Labelled transition model.

"*from the initial state* s_0*, all transitions take to state in which* p *is true*". A traditional model checker would represent the system model as in Figure 2(a), i.e., by a *Kripke* model. Since the state s_2 does not satisfy φ, the CTL model checker would detect this error and a model update would indicate "*remove the relation* (s_0, s_2)". However, if we represent the transition actions (Figure 2(b)) a model checker would not detect an error, since there is an action a that takes the system to state s_2 that satisfies φ. Figure 2(c) shows another example: by knowing that the two transitions correspond to non-deterministic effects of action a, removing one transition implies removing the other.

Since actions are not part of *Kripke* structure, we can only represent them using a logic whose semantics is based on actions. Pereira and Barros (2008) proposed an extension of CTL, called α-CTL, whose semantics considers actions behind transitions and a model checker based on this logic, named α-MODEL CHECKER. In this paper, we present a model updating approach based on α-CTL that can be used to automatically suggest minimal changes in a state transition system labelled with actions.

Related works. The idea of integrating model checking and automatic modification has been investigated in recent years by exploring a variety of approaches. Buccafurri et al (1999) joint abductive theory revision with model checking, developing an algorithm that can identify errors and provide modifications for concurrent systems. However, this approach is quite restricted since only relation elements can be changed in a Kripke model. There are important works on belief update area that has contributed to evolve the theory of model update, such as: Winslett (1988) proposes a pioneering idea towards a model-base minimal change approach for knowledge-base update; Katsuno and Mendelzon (1991) generalizes Winslett's work and elaborates a set of postulates for finite and propositional knowledge-base update; and Harris and Ryan (2003) discusses about theoretical properties of system update. Finally, Zhang and Ding (2008) combined belief update and CTL model checking techniques to develop their framework to automatically repair a fault system, which was used as the basis of our work.

The remainder of this article is organized as follows: in Section 2, we show the basic concepts of CTL model checking and model update; in Section 3, we define a labelled transition system; in Section 4, we present a model checker based on α-CTL; in Section 5, we show how to perform model update, based on α-CTL, that can suggest modifications in a labelled transition system according with minimal change criterion; and finally, in Section 6, we present our conclusions.

2 CTL Model Checking and Updating

Model checking consists of solving the problem $\mathcal{K} \models \varphi$, where $\mathcal{K}$ is a formal model of a system and φ is a formal specification of a property to be verified in this system. Essentially, a model checker (Figure 3) is an algorithm that receives a pair $(\mathcal{K}, \varphi)$ as input and systematically visits the states of the model $\mathcal{K}$, in order to verify if the property φ holds. When all states in $\mathcal{K}$ satisfy property φ, the model checker returns success; otherwise, it returns a counter-example (e.g., a state in the model $\mathcal{K}$ where the property φ is violated).

Fig. 3. Model checker

The property φ to be verified is specified on *temporal logic*. A large variety of temporal logics has been proposed and the abundance of such formalisms may be organised by classifying them according to their particular view of "time". *Linear time* logics, e.g. LTL [8], think of time as a set of paths, where a path is a sequence of time instances. *Branching-time* logics represent time as a tree, rooted at the present moment and branching out into the future. In this paper, we will consider a logic where time is branching, namely CTL [2].

2.1 CTL: Computation Tree Logic

CTL is a branching time temporal logic that allows us to reason about alternative time lines (i.e., alternative futures). This logic is the formalism used in most of the model checking approaches to specify the properties to be verified.

The CTL formulas are composed by atomic propositions, propositional operators and temporal operators. The symbols $\circ$ (next), $\Box$ (invariantly), $\diamond$ (finally) and $\sqcup$ (until), combined with the quantifiers $\exists$ and $\forall$, are used to compose the temporal operators of this logic. Let $\mathbb{P} \neq \emptyset$ be a finite set of atomic propositions, denoting states properties, the syntax of CTL is inductively defined as:
$\varphi \doteq p \in \mathbb{P} \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists \circ \varphi_1 \mid \forall \circ \varphi_1 \mid \exists\Box\varphi_1 \mid \forall\Box\varphi_1 \mid \exists(\varphi_1 \sqcup \varphi_2) \mid \forall(\varphi_1 \sqcup \varphi_2)$.

The semantics of CTL is defined over a *Kripke structure* [5] $\mathcal{K} = \langle S, L, T \rangle$, where S is a set of states, $L : S \mapsto 2^{\mathbb{P}}$ is a state labelling function and $T \subseteq S \times S$ is a transition relation. A path in $\mathcal{K}$ is a sequence of states $[s_0, s_1, \cdots]$ such that $s_i \in S$ and $(s_i, s_{i+1}) \in T$, for all $i \geqslant 0$.

2.2 Model Update Framework

Let $\mathcal{K}$ be a formal model of a system and φ be a formal specification of a property that is not satisfied in this system, i.e., $\mathcal{K} \not\models \varphi$. *Model update* [10] consists of generating a new model $\mathcal{K}'$ that satisfies the input formula ($\mathcal{K}' \models \varphi$) and has a

Fig. 4. Model updater receives $\mathcal{K}$ and φ and returns an updated model $\mathcal{K}$'

minimal change w.r.t. the original model $\mathcal{K}$. Then, a *model updater* (Figure 4) is an algorithm that receives a pair $(\mathcal{K}, \varphi)$, where $\mathcal{K} \not\models \varphi$, and returns a new model $\mathcal{K}'$, where $\mathcal{K}' \models \varphi$. The updated model $\mathcal{K}'$ can be viewed as a possible correction on the original system specification.

Zhang and Ding (2008) proposed a formal framework for CTL model update, defining primitive operations and specifying a minimal change principle for CTL model updating, as listed next.

PU1: Adding one relation element. Let be $\mathcal{K} = \langle S, L, T\rangle$, its updated model $\mathcal{K}' = \langle S', L', T'\rangle$ is obtained from $\mathcal{K}$ by adding only one new relation element. That is: $S' = S; L' = L$ and $T' = T \cup (s_i, s_j)$, where $s_i, s_j \in S$ and $(s_i, s_j) \notin T$.

PU2: Removing one relation element. Let be $\mathcal{K} = \langle S, L, T\rangle$, its updated model $\mathcal{K}' = \langle S', L', T'\rangle$ is obtained from $\mathcal{K}$ by removing only one existing relation element. That is: $S' = S$; $L' = L$ and $T' = T - (s_i, s_j)$, where $(s_i, s_j) \in T$ for two states $s_i, s_j \in S$.

PU3: Adding one state. Let be $\mathcal{K} = \langle S, L, T\rangle$, its updated model $\mathcal{K}' = \langle S', L', T'\rangle$ is obtained from $\mathcal{K}$ by adding only one new state. That is: $S' = S \cup s^*, s^* \notin S$; $T' = T$ and $\forall s \in S, L'(s) = L(s)$ and $L'(s^*)$ is a set of true variables assigned in s^*.

PU4: Removing one isolated state. Let be $\mathcal{K} = \langle S, L, T\rangle$, its updated model $\mathcal{K}' = \langle S', L', T'\rangle$ is obtained from $\mathcal{K}$ by removing only one isolated state. That is: $S' = S - s^*$, where $s^* \in S$ and $\forall s \in S$ such that $s \neq s^*$, neither (s, s^*) nor (s^*, s) is not in T; $T' = T$ and $\forall s \in S', L'(s) = L(s)$.
The primitive operations *PU1-PU4* were used to define minimal change criterion for CTL model update. However, *PU1-PU4* can add and remove states and relations without considering state constraints and actions. In this work, we show that by considering the actions in the system specification, we can extend the set of primitive operations *PU1-PU4* and refine the minimal change criterion in order to suggest a more rational update to a system designer.

3 Labelled Transition System

Let $\mathbb{P} \neq \emptyset$ be a finite set of atomic propositions, denoting properties of a system, and $\mathbb{A} \neq \emptyset$ be a finite set of actions, representing the events of a system.

Definition 1. *A Labelled Transition System with signature* $(\mathbb{P}, \mathbb{A})$ *is defined by* $\mathcal{M} = \langle S, L, T\rangle$*, where:*

- $S \neq \emptyset$ *is a finite set of states;*

– $L : \mathcal{S} \mapsto 2^{\mathbb{P}}$ *is a state labelling function and*
– $T : S \times \mathbb{A} \times S$ *is a state transition relation.*

A labelled transition system with signature $(\mathbb{P}, \mathbb{A})$ can be represented as a *transition graph*, where states are labelled with subsets of $\mathbb{P}$ and transitions are labelled with elements of $\mathbb{A}$. Set $\mathcal{S}$ has all possible states of a model, $L(s)$ assigns for each state $s \in S$ a proposition set and labelling function T assigns for each transition an action $a \in \mathbb{A}$. Given two states $s_i, s_j \in S$ and an action $a \in \mathbb{A}$, a transition between s_i and s_j is represented by (s_i, a, s_j).

4 α-CTL Model Checking

α-CTL [7] is a branching time temporal logic that allows us to discern among various actions that produce state transitions. On one hand, a CTL formula $\forall \circ \varphi$ holds on a state s if and only if it holds on all successors of s, independently of the actions labelling the transitions from s to its successors; on the other hand, in α-CTL , to enforce that actions play an important role in its semantics, we use a different set of "dotted" symbols to represent temporal operators: $\odot$ (next), $\boxdot$ (invariantly), $\diamondsuit$ (finally) and $\sqcup\!\!\cdot$ (until).

Definition 2. *Let $p \in \mathbb{P}$ be an atomic proposition. The syntax of* α-CTL *is inductively defined as:* $\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists \odot \varphi \mid \forall \odot \varphi \mid \exists \boxdot \varphi \mid \forall \boxdot \varphi \mid \exists(\varphi_1 \sqcup\!\!\cdot\, \varphi_2) \mid \forall(\varphi_1 \sqcup\!\!\cdot\, \varphi_2)$

According to α-CTL's syntax, well-formed formulas are in *negative normal form*, where the scope of negation is restricted to the atomic propositions. Since α-CTL is based on a fix-point semantics this restriction allows to easily define the semantics for the formulas. Furthermore, all temporal operators are prefixed by a path quantifier ($\exists$ or $\forall$). The temporal operators derived from $\diamondsuit$ are defined as: $\exists \diamondsuit \varphi_2 \doteq \exists(\top \sqcup\!\!\cdot\, \varphi_2)$ and $\forall \diamondsuit \varphi_2 \doteq \forall(\top \sqcup\!\!\cdot\, \varphi_2)$.

The semantics of α-CTL is defined over a *labelled transition system* $\mathcal{M} = \langle S, L, T \rangle$. Intuitively, a state s in $\mathcal{M}$ satisfies a formula $\forall \odot \varphi$ (or $\exists \odot \varphi$) if there exists an action α that, when executed in s, *necessarily* (or *possibly*) reaches an immediate successor of s which satisfies the formula φ. In other words, the modality $\odot$ represents the set of α-successors of s, for some particular action $\alpha \in \mathbb{A}$; the quantifier $\forall$ requires that all these α-successors satisfy φ; and quantifier $\exists$ requires that some of these α-successors satisfy φ. More details about the semantics of α-CTL can be found in [7]. A model checker for α-CTL can be directly implemented from its semantics [7]. Given a model $\mathcal{M} = \langle S, L, T \rangle$ and an α-CTL formula φ, the model checker computes the set C of states that do not satisfy the formula φ in $\mathcal{M}$; then, if C is the empty set, it returns success; otherwise, it returns C as counter-example.

$\alpha -$ ModelChecker$(\varphi, \mathcal{M})$
1 $C \leftarrow S \setminus$ Intention$(\varphi, \mathcal{M})$
2 **if** $C = \emptyset$ **then return success**
3 **else return** C

The basic operation on this model checker is implemented by the INTENTION function, that inductively computes the set of states in $\mathcal{M}$ that satisfy φ. The complexity of this model checker is given by the INTENTION function which is linear w.r.t. the size of state space ($O(|S|)$). Details about α-MODELCHECKER can be found in [7].

5 α-CTL Model Update

In order to design a real world system, we should start by deciding which are the system properties that matter to guarantee a desired system behaviour. This is done by defining the set of propositions $\mathbb{P}$. Then, the system designer should take into account domain constraints (sometimes called *state constraints*) which is simply a constraint that establishes what combinations of $p \in \mathbb{P}$ may hold in a state.

Definition 3. *(Complete Model) Given a set of propositions $\mathbb{P}$, a set of domain constraints Δ and a set of actions $\mathbb{A}$, a complete model is a labelled transition system $\mathcal{M}^* = \langle S^*, L^*, T^* \rangle$, as in Definition 1, where $S^* = \{s | s \in 2^{\mathbb{P}} \wedge s \models \Delta\}$ and $T^\star$ is induced by the semantics of actions in $\mathbb{A}$.*

That means, $\mathcal{M}^\star$ contains all states $s \in 2^{\mathbb{P}}$ that satisfy the domain constraints Δ and all transitions $T^\star$ induced by the set of actions $\mathbb{A}$.

Definition 4. *(Partial Model) The system model that the designer wants to correct is a partial model $\mathcal{M} = \langle S, L, T \rangle$, such that $\mathcal{M} \subseteq \mathcal{M}^\star$ or, more precisely:*

- *$S \subseteq S^*$ is a finite set of states;*
- *$L : S \mapsto 2^{\mathbb{P}}$ such that, for all $s \in S, L(s) = L^*(s)$;*
- *$T \subseteq T^*$ such that, if $(s_i, a, s_j) \in T$ and $(s_i, a, s_k) \in T^*$, then $(s_i, a, s_k) \in T$.*

In Figure 5, we have a labelled transition system, representing a complete model $\mathcal{M}^*$, where the highlighted substructure is the partial model $\mathcal{M} \subseteq \mathcal{M}^*$ given by system designer. The rationality is: considering the complete model as a compilation of the domain constraints and the semantics of actions, any realistic change in the partial model should be based on the complete model.

Definition 5. *(α-CTL Model Update) Given a partial model $\mathcal{M} \subseteq \mathcal{M}^\star$, an initial state $s_0 \in S$ and an α-CTL formula φ such that $(\mathcal{M}, s_0) \not\models \varphi$, α-CTL model updating problem consists of generating a new model $\mathcal{M}'$ such as (i) $(\mathcal{M}', s_0) \models \varphi$ and (ii) $\mathcal{M}'$ has a minimal change with respect to the original model $\mathcal{M}$ and those minimal changes are elements of the complete model $\mathcal{M}^*$, i.e., $\mathcal{M}' \subseteq \mathcal{M}^*$.*

5.1 Primitive Operations for α-CTL Model Updating

To update a partial model $\mathcal{M}$, w.r.t. a complete model $\mathcal{M}^\star$ and with the goal of satisfying an α-CTL formula φ, we define the follow primitive updates operations. Note that, different from the primitive update operations *PU1-PU4*, we now define operations in a (action) labelled transition system.

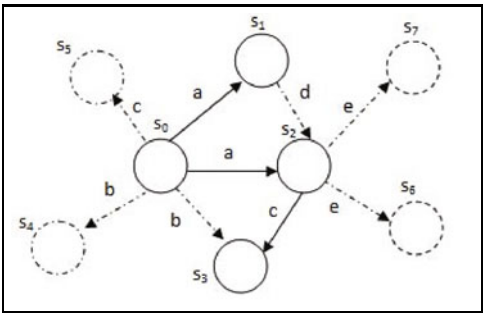


Fig. 5. A labelled transition system. Solid lines represent a partial model $\mathcal{M} \subseteq \mathcal{M}^\star$.

PUA1: Adding transitions induced by an action. Given a partial model $\mathcal{M} = \langle S, L, T \rangle$, a corresponding updated model $\mathcal{M}' = \langle S', L', T' \rangle$, with respect to $\mathcal{M}^\star = \langle S^\star, L^\star, T^\star \rangle$, can be obtained from $\mathcal{M}$ by adding a transition between states $s_i, s_j \in S$. In other words: $T' = T \cup \{(s_i, a, s) \in T^\star : \exists a \in \mathbb{A}, (s_i, a, s_j) \in T^\star\}$; $S' = S \cup \{s : \exists a \in \mathbb{A}, (s_i, a, s) \in T'\}$ and $L'(s) = L^\star(s), s \in S'$.

Adding a transition between two states s_i and s_j in the partial model is possible only if there is some transition between these states in the complete model. For example, in Figure 5 we cannot add a transition between states s_2 and s_4 in the partial model (solid lines), because there is no transition between s_2 and s_4 in the complete model (dashed lines). But, it is possible to add a transition between s_0 and s_3, since a transition (s_0, b, s_3) exists in the complete model. However, all the effects of action b must also be added. In Figure 5 in order to add a transition between states s_0 and s_3, using action b, we must also add state s_4 and the transition between s_0 and s_4.

PUA2: Removing transitions induced by an action. Given a partial model $\mathcal{M} = \langle S, L, T \rangle$, a corresponding updated model $\mathcal{M}' = \langle S', L', T' \rangle$, with respect to $\mathcal{M}^\star = \langle S^\star, L^\star, T^\star \rangle$, can be obtained from $\mathcal{M}$ by removing an existing transition between states $s_i, s_j \in S$, which is labelled by some action $a \in \mathbb{A}$. More formally: $T' = T - \{(s_i, a, s) \in T^\star : (s_i, a, s_j) \in T\}$; $S' = S$ and $L'(s) = L(s), s \in S'$.

It is important to observe that to remove an existing transition between states s_i and s_j labelled by an action a, we must also remove all transitions from s_i using action a, i.e., all non-deterministic effects of the action a from s_i. E.g., to remove (s_0, a, s_1) in the partial model of Figure 5, it is also necessary to remove the transition (s_0, a, s_2).

PUA3: Adding a new state. Given a partial model $\mathcal{M} = \langle S, L, T \rangle$, a corresponding updated model $\mathcal{M}' = \langle S', L', T' \rangle$, with respect to $\mathcal{M}^\star = \langle S^\star, L^\star, T^\star \rangle$, can be obtained from $\mathcal{M}$ by adding only one new state. That is: $S' = S \cup \{s\}$, for some $s \in S^\star$, such that $s \notin S$; $L'(s) = L^\star(s), s \in S'$ and $T' = T$.

It is possible to add one state s in the partial model if and only if this state there exists in the complete model.

PUA4: Removing an isolated state. Given a partial model $\mathcal{M} = \langle S, L, T\rangle$, a corresponding updated model $\mathcal{M}' = \langle S', L', T'\rangle$, with respect to $\mathcal{M}^\star = \langle S^\star, L^\star, T^\star\rangle$, can be obtained from $\mathcal{M}$ by removing only one isolated state. That is: $S' = S - \{s\}$, for some $s \in S$, such that for all $s_i \in S$, $s_i \neq s$, and $a \in \mathbb{A}$, we have $(s_i, a, s) \notin T$ and $(s, a, s_i) \notin T$; $L'(s) = L(s), s \in S'$ and $T' = T$.

5.2 Minimal Change Criterion for α-CTL Model Updating

Based on the work of Zhang and Ding (2008), we can establish a *minimal change criterion* for a labelled transition system using the new primitive update operations (PUA1 – PUA4), defined in previous section.

Given a partial model $\mathcal{M}'$ by a system designer and an α-CTL formula φ such that $\mathcal{M}' \not\models \varphi$, the partial model can be corrected in different ways by applying different combinations of the primitive operations (PUA1 – PUA4). E.g., $\mathcal{M}'$ is updated model w.r.t. $\mathcal{M}$ after applying the following combination of operations $\{\text{PUA2}, \text{PUA4}, \text{PUA2}, \text{PUA4}, \text{PUA1}\}$. Thus, we need a criterion which allows us to measure the changes in the different possible updated models of $\mathcal{M}$ and choose the one which is more close to the original model $\mathcal{M}$.

Given a labelled transition system $\mathcal{M} = \langle S, L, T\rangle$ and a corresponding updated model $\mathcal{M}' = \langle S', L', T'\rangle$, for each operation PUA$i$, $\mathit{Diff}_{\text{PUAi}}$ $(\mathcal{M}, \mathcal{M}')$ denoting the difference between these two models caused by PUAi is:

$$\mathit{Diff}_{\text{PUAi}}(\mathcal{M}, \mathcal{M}') = |T' - T| + |S' - S|.$$

We also define $\mathit{Diff}(\mathcal{M}, \mathcal{M}') = \langle \mathit{Diff}_{\text{PUA1}}(\mathcal{M}, \mathcal{M}'\rangle,\ \mathit{Diff}_{\text{PUA2}}(\mathcal{M}, \mathcal{M}')$, $\mathit{Diff}_{\text{PUA3}}(\mathcal{M}, \mathcal{M}')$, $\mathit{Diff}_{\text{PUA4}}(\mathcal{M}, \mathcal{M}')\rangle$ denoting the overall difference between $\mathcal{M}$ and $\mathcal{M}'$ after applying a sequence of primitive operations.

Definition 6. *(Ordering the updated models w.r.t. $\mathcal{M}$) Given $\mathcal{M}$ a partial model and $\mathcal{M}'_1$, $\mathcal{M}'_2$ two corresponding updated models, with respect to a complete model $\mathcal{M}^\star$. We say that $\mathcal{M}'_1$ is at least as close to $\mathcal{M}$ as $\mathcal{M}'_2$ (denoted by $\mathcal{M}'_1 \leq_{\mathcal{M}} \mathcal{M}'_2$) if only if for the set of operations that transformed $\mathcal{M}$ into $\mathcal{M}'_2$, and the a set of operations that transformed $\mathcal{M}$ into $\mathcal{M}'_1$, we have:*

$$\mathit{Diff}_{PUAi}(\mathcal{M}, \mathcal{M}'_1) \leq \mathit{Diff}_{PUAi}(\mathcal{M}, \mathcal{M}'_2), \text{ for } i = 1..4$$

We denote $\mathcal{M}'_1 <_{\mathcal{M}} \mathcal{M}'_2$ if $\mathcal{M}'_1 \leq_{\mathcal{M}} \mathcal{M}'_2$ and $\mathcal{M}'_2 \not\leq_{\mathcal{M}} \mathcal{M}'_1$. For example, if $\mathit{Diff}(\mathcal{M}, \mathcal{M}'_1) = \langle 5, 2, 4, 6\rangle$ and $\mathit{Diff}(\mathcal{M}, \mathcal{M}'_2) = \langle 3, 1, 1, 5\rangle$. We say that $\mathcal{M}'_2 <_{\mathcal{M}} \mathcal{M}'_1$, i.e., $\mathcal{M}'_2$ is more close to $\mathcal{M}$ than $\mathcal{M}'_1$ is.

Definition 6 presents a measure on the difference between two labelled transition system with respect to a partial model given by designer. Intuitively, we say that model M'_1 is closer to M relative to model M'_2 if M'_1 is obtained from M by applying all primitive update operations that cause fewer changes than those applied to obtain model M_2. Having the ordering specified in Definition 6, we can define a α-CTL model updating formally.

Definition 7. *(Admissible Update) Let be a partial model $\mathcal{M} = \langle S, L, T\rangle$, w.r.t complete model $\mathcal{M}^\star$, $s_0 \in S$ and an α-CTL formula φ $\mathcal{M}' = Update((\mathcal{M}, s_0), \varphi)$ is an admissible update if only if:*

- $(\mathcal{M}', s_0) \models \varphi$
- *There is not a model* $\mathcal{M}'' = Update((\mathcal{M}, s_0), \varphi)$ *such that* $(\mathcal{M}'', s_0) \models \varphi$ *and* $\mathcal{M}'' <_{\mathcal{M}} \mathcal{M}'$.

6 Conclusion

In this work we have presented a model updating approach that considers the application domain contingencies and constraints. Our approach is based on the work of [7] that considers actions behind the transitions in a state transition system to perform model checking using a new branching time temporal logic named α-CTL. We have defined a new set of primitive update operators *PUA1-PUA4* based on the work of [10], that can suggest a more realistic set of model updates obeying domain constraints and the actions semantics (compiled in a complete model $\mathcal{M}^*$). We also formalized the minimal change principle for α-CTL model update. The α-CTL model updater takes (i) a complete model $\mathcal{M}^*$ (induced by the domain actions and constraints), (ii) a partial model $\mathcal{M}$ such that $\mathcal{M} \subseteq \mathcal{M}^*$, (iii) an α-CTL formula and returns an updated model $\mathcal{M}' \subseteq \mathcal{M}^*$ where $\mathcal{M}' \models \varphi$ and $\mathcal{M}'$ has minimal change w.r.t. $\mathcal{M}$.

Acknowledgements. This work has been supported by CNPq Project 304322/2009-1, FAPESP project 2009/07039-4 and FAPESP project 2008/03995-5 (LogProb).

References

1. Buccafurri, F., Eiter, T., Gottlob, G., Leone, N.: Enhancing model checking in verification by AI techniques. J. Artif. Intell. 112, 57–104 (1999)
2. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
3. Harris, H., Ryan, M.: Theoretical foundations of updating systems. In: 18th IEEE International Conference on Automated Software Engineering, pp. 291–294 (2003)
4. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: 2nd International Conference on the Principles of Knowledge Representation and Reasoning, pp. 387–394 (1991)
5. Kripke, S.: Semantical Considerations on Modal Logic. J. Acta Philosophica Fennica 16, 83–94 (1963)
6. Müller-Olm, M., Schmidt, D., Steffen, B.: Model-Checking: A Tutorial Introduction. In: Cortesi, A., Filé, G. (eds.) SAS 1999. LNCS, vol. 1694, pp. 330–354. Springer, Heidelberg (1999)
7. Pereira, S.L., Barros, L.N.: A logic-based agent that plans for extended reachability goals. Journal Autonomous Agents and Multi-Agent Systems 16, 327–344 (2008)
8. Pnueli, A.: The temporal logic of programs. In: 18th Symposium Foundations of Computer Science, pp. 46–57 (1977)
9. Winslett, M.: Reasoning about action using a possible models approach. In: 7th American Association for Artificial Intelligence, pp. 89–93 (1988)
10. Zhang, Y., Ding, Y.: CTL Model Update for System Modifications. J. Artif. Int. Res. 31, 113–155 (2008)

Learning Terminologies in Probabilistic Description Logics

Kate Revoredo[1], José Eduardo Ochoa-Luna[2], and Fabio Gagliardi Cozman[2]

[1] Departamento de Informática Aplicada, Unirio
Av. Pasteur, 458, Rio de Janeiro, RJ, Brazil
[2] Escola Politécnica, Universidade de São Paulo,
Av. Prof. Mello Morais 2231, São Paulo - SP, Brazil
katerevoredo@uniriotec.br, eduardo.ol@gmail.com, fgcozman@usp.br

Abstract. This paper investigates learning methods where the target language is the recently proposed probabilistic description logic CR$\mathcal{ALC}$. We start with an inductive logic programming algorithm that learns logical constructs; we then develop an algorithm that learns probabilistic constructs by searching for conditioning concepts, using examples given as interpretations. Issues on learning from entailments are also examined, and practical examples are discussed.

1 Introduction

A description logic offers a formal language where one can describe concepts such as "A Professor is a Person who teaches in a University". To do so, a description logic typically uses a decidable fragment of first-order logic, trying to reach a practical balance between expressivity and complexity. The last decade has seen a significant increase in interest in description logics as a vehicle for large-scale knowledge representation, for instance in the semantic web. Indeed, the language OWL [1], proposed by the W3 consortium as the data layer of their architecture for the semantic web, is an XML encoding for an expressive description logics.

Description logics are not geared towards the representation of uncertainty about objects and concepts: one cannot express that "with low probability a Person is a Professor". The literature contains a number of proposals that add probabilistic uncertainty to description logics, as this is central to the management of semantic data in large repositories. Cozman and Polastro have proposed [3] a probabilistic extension of the popular logic $\mathcal{ALC}$ [2], called Credal $\mathcal{ALC}$ (CR$\mathcal{ALC}$), where sentences such as $P(\mathsf{Professor}|\mathsf{Researcher}) = 0.4$, refereing to the probability that an object is a Professor given that it is a Researcher, are allowed. These sentences are called *probabilistic inclusions*. Inference algorithms based on Relational Bayesian networks [12] have been proposed; because exact inference does not seem to be scalable when quantified concepts are employed, approximate inference algorithms have been developed [3,4].

An important question is how to automatically learn concepts expressed with description logics. Previous efforts with concept learning in CR$\mathcal{ALC}$ [14] have

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 41–50, 2010.

focused only on logical concept learning, using the ideas of refinement operators from DL-Learner [13] and DL-FOIL [9], and not dealing with probabilistic inclusions. In this paper, we assume that (logical) concept definitions can be learned using any algorithm in the literature and we attack the problem of learning probabilistic inclusions. For that, we propose an algorithm that learns the structure of the probabilistic inclusion by discovering how a concept is probabilistically conditioned on other concepts and then by estimating the probability of the probabilistic inclusion. The proposed algorithm learns from examples given as interpretations. We also discuss the generalization of these ideas to other settings; namely, learning from entailments and learning from proofs [7].

Section 2 provides background knowledge on description logics (Section 2.1), on probabilistic description logics (Section 2.2) and on relevant learning algorithms based on Inductive Logic Programming (Section 2.3). Section 3 proposes a learning algorithm for CR$\mathcal{ALC}$ probabilistic inclusions and Section 4 reports on some preliminaries experiments. Section 5 concludes the paper.

2 Background

Assume we are given a repository of HTML pages where researchers and students have stored data about publications, courses, languages. We might wish to extract the definition of some concepts, such as Researcher and Person, and relationships among them. Suppose however that we are unable to state deterministic relations among concepts, but instead we can only give probabilistic inclusions such as $P(\mathsf{Professor}|\mathsf{Researcher}) = 0.4$. Probabilistic inclusions are allowed in probabilistic description logics, as reviewed in this section.

2.1 Description Logics

Description logics (DLs) form a family of representation languages that are typically decidable fragments of first order logic (FOL) [2]. Knowledge is expressed in terms of *individuals*, *concepts*, and *roles*. The semantics of a description is given by a *domain* Δ (a set) and an *interpretation* $\cdot^{\mathcal{I}}$ (a functor). Individuals represent objects through names from a set $N_{\mathsf{I}} = \{a, b, \ldots\}$. Each *concept* in the set $N_{\mathsf{C}} = \{C, D, \ldots\}$ is interpreted as a subset of a domain $\mathcal{D}$. Each *role* in the set $N_{\mathsf{R}} = \{r, s, \ldots\}$ is interpreted as a binary relation on the domain.

Concepts and roles are combined to form new concepts using a set of *constructors*. Constructors in the $\mathcal{ALC}$ logic are *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$), and *value* restriction ($\forall r.C$). *Concept inclusions/definitions* are denoted respectively by $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. Concepts ($C \sqcup \neg C$) and ($C \sqcap \neg C$) are denoted by $\top$ and $\bot$ respectivelly. Information is stored in a *knowledge base* ($\mathcal{K}$) divided in two parts: the TBox (terminology) and the ABox (assertions). The TBox lists concepts and roles and their relationships. A TBox is acyclic if it is a set of concept inclusions/definitions such that no concept in the terminology uses itself. The ABox contains assertions about objects.

Given a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$, the reasoning services typically include (i) consistency problem (to check whether the $\mathcal{A}$ is consistent with respect to the $\mathcal{T}$); (ii) entailment problem (to check whether an assertion is entailed by $\mathcal{K}$; note that this generates class-membership assertions $\mathcal{K} \models C(a)$, where a is an individual and C is a concept); (iii) concept satisfiability problem (to check whether a concept is subsumed by another concept with respect to the $\mathcal{T}$). The latter two reasoning services can be reduced to the consistency problem [2].

2.2 Probabilistic Description Logics and CR$\mathcal{ALC}$

Several probabilistic descriptions logics (PDLs) have appeared in the literature. Heinsohn [10], Jaeger [11] and Sebastiani [16] consider probabilistic inclusion axioms such as $P_{\mathcal{D}}(\mathsf{Professor}) = \alpha$, meaning that a randomly selected object is a Professor with probability α. This characterizes a *domain-based* semantics: probabilities are assigned to subsets of the domain $\mathcal{D}$. Sebastiani also allows inclusions such as $P(\mathsf{Professor}(\mathsf{John})) = \alpha$ as well, specifying probabilities over the interpretations themselves. For example, one interprets $P(\mathsf{Professor}(\mathsf{John})) = 0.001$ as assigning 0.001 to be the probability of all interpretations where John is a Professor. This characterizes an *interpretation-based* semantics.

The PDL CR$\mathcal{ALC}$ is a probabilistic extension of the DL $\mathcal{ALC}$ that adopts an interpretation-based semantics. It keeps all constructors of $\mathcal{ALC}$, but only allows concept names in the left hand side of inclusions/definitions. Additionally, in CR$\mathcal{ALC}$ one can have probabilistic inclusions such as $P(C|D) = \alpha$ or $P(r) = \beta$ for concepts C and D, and for role r. For any element of the domain, the probability that this element is in C, given that it is in D is α. If the interpretation of D is the whole domain, then we simply write $P(C) = \alpha$. The semantics of these inclusions is roughly (a formal definition can be found in [4]) given by:

$$\forall x \in \mathcal{D} \ : \ P(C(x)|D(x)) = \alpha,$$

$$\forall x \in \mathcal{D}, y \in \mathcal{D} \ : \ P(r(x,y)) = \beta.$$

We assume that every terminology is acyclic; no concept uses itself. This assumption allows one to represent any terminology $\mathcal{T}$ through a directed acyclic graph. Such a graph, denoted by $\mathcal{G}(\mathcal{T})$, has each concept name and role name as a node, and if a concept C directly uses concept D, that is if C and D appear respectively in the left and right hand sides of an inclusion/definition, then D is a *parent* of C in $\mathcal{G}(\mathcal{T})$. Each existential restriction $\exists r.C$ and value restriction $\forall r.C$ is added to the graph $\mathcal{G}(\mathcal{T})$ as nodes, with an edge from r to each restriction directly using it. Each restriction node is a *deterministic* node in that its value is completely determined by its parents.

The semantics of CR$\mathcal{ALC}$ is based on probability measures over the space of interpretations, for a fixed domain. Inferences, such as $P(\mathsf{A}_o(\mathsf{a}_0)|\mathcal{A})$ for an ABox $\mathcal{A}$, can be computed by propositionalization and probabilistic inference (for exact calculations) or by a first order loopy propagation algorithm (for approximate calculations) [4].

2.3 Learning in Description Logics

The techniques developed in this paper are inspired by two systems based on inductive logic programming (ILP) [5], namely, DL-Learner [13] and DL-FOIL [9]. The goal in ILP is to find an hypothesis H (a definite clause program) that covers all examples in a set E_p of positive examples and none examples in a set E_n of negative examples. In the context of DLs, the hypothesis is a knowledge base and the examples are assertions. Consider a concept name Target, a knowledge base $\mathcal{K}$ not containing Target, and sets of positive and negative examples with elements of the form $\mathsf{Target}(a)$, where a is an object of the domain $\mathcal{D}$. Acyclicity is assumed; that is, recursive inclusions/definitions are not learned. The learning problem is to find a concept C such that Target does not occur in C and for $\mathcal{K}' = \mathcal{K} \cup \{\mathsf{Target} \equiv C\}$ we have $\mathcal{K}' \models E_p$ and $\mathcal{K}' \not\models E_n$. This problem can be solved through a search in the space of concepts. Lehmann and Hitzler [13] impose an ordering on this search space and then use *refinement operators* to traverse it. Intuitively, downward (upward) refinement operators construct specializations (generalizations) of hypotheses. De Raedt [7] defines three settings for learning a hypothesis in ILP:

i) learning from entailments: the examples are definite clauses and a hypothesis H covers an example e with background knowledge B if and only if $B \cup H \models e$. In many well-known systems, such as FOIL [15], one requires that the examples are ground facts, a special case of definite clauses.
ii) learning from interpretations: the examples are *Herbrand interpretations* (set of true ground facts that completely describe a possible situation) and an hypothesis H covers an example e with background knowledge B if and only if e is a model of $B \cup H$.
iii) learning from proofs: the examples are ground proof-trees and an example e is covered by a hypothesis H with background knowledge B if and only if e is a proof-tree for $H \cup B$.

The key difference between learning from interpretations and learning from entailment is that interpretations carry much more information. Indeed, when learning from entailments, an example may consist of a single fact, whereas when learning from interpretations, complete descriptions must be available. Therefore, learning from interpretations requires less computational effort than learning from entailments [6]. Another point is that learning from interpretations is well suited for learning from positive examples only.

DL learning algorithms such as DL-Learner and DL-FOIL adopt learning from entailments. In that case, examples are assertions about the new concept Target and an example e is covered by a knowledge base $\mathcal{K}$ if and only if $\mathcal{K} \models e$. A DL learning algorithm can be proposed based on learning from interpretations setting, as we propose in the next section for the PDL CR$\mathcal{ALC}$.

3 Learning in CR$\mathcal{ALC}$

In this section, we propose our algorithm for probabilistic inclusion learning in CR$\mathcal{ALC}$. We assume learning from interpretations, so as to handle gracefully the

interpretation-based semantics of CR$\mathcal{ALC}$. The idea is as follows. Suppose that a concept C is learned (logically only, without probabilities!) as $C \equiv D \sqcap F$. However, there is noise in the data and a probabilistic inclusion relating these concepts would be interesting. Therefore, probabilistic inclusions of the form $P(C|X) = \alpha$ are examined, where X are concepts based on D and F, and α is a constant learned by maximum likelihood estimation. Here we search for the best conditioning concept, by considering parts of the concept definition; in our example, X can be the whole domain, D, F or D $\sqcap F$, therefore, probabilities such as $P(C)$, $P(C|D)$, $P(C|F)$ and $P(C|D \sqcap F)$ are evaluated. We then examine, based on an evaluation function, whether it is best to discard the logical definition (that is, $C \equiv D \sqcap F$). In short, we focus on learning probabilistic inclusions given a base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a set of examples E. We assume that a description logic learning algorithm, for instance DL-Learner or DL-FOIL, is first run to learn definitions for concepts in $\mathcal{T}$.

3.1 Direct Inference and EM

As CR$\mathcal{ALC}$ adopts an interpretation-based semantics, we would ideally need data on each individual in order to learn descriptions. For instance, if we wanted to learn the probability of $P(\mathsf{Professor}(a))$, we would need a dataset with information about the object a having or not the property Professor. Different objects can have different probabilities; most probabilistic description logics that adopt interpretation-based semantics combine these different probabilities into a single value to be adopted by all objects (that is, $\forall x : P(C(x)) = \alpha$).

Most available datasets contain information about all (or at least many) objects of the domain, instead of information about one specific object. Learning with such a dataset means learning probabilities with domain-based semantics. In this paper we consider that the probabilities learned with domain-based semantics are a good approximation for probabilities with interpretation-based semantics; that is, we conduct *direct inference* by transfering learned probabilities from domain to interpretations. Our algorithm learns probabilities in probabilistic inclusions by approximating the probabilities estimated with domain-based semantics.

For estimating domain-based probabilities, we resort to the *Expectation Maximization (EM)* algorithm [8]. The likelihood function (the *score*) to be maximized is

$$L = \prod_{e \in E} P(e|\mathcal{K}).$$

3.2 Learning the Structure of Concepts

Our algorithm for structure learning starts by taking concept names as probabilistically independent, and by learning probabilistic inclusions such as $P(C)$. Then the algorithm verifies whether a concept name provides better results (better inference) if probabilistically conditioned on another concept name or on a

concept description. For that the algorithm considers possibilities from the logical definition of the concept (as learned by ILP), since it seems nature that the concept C can only be conditioned on concepts that the learning process learned that it is related with. The algorithm uses subsumption, and keeps only the conditioning concept that provides the highest score. When a candidate probabilistic inclusion uses only part of a original logically learned concept description, further analysis is needed. A decision must be taken as to whether the probabilistic inclusion should be added and the original concept should be discarded. (Note that if both the original concept description and the probabilistic inclusion were kept together, then we might not be able to construct a unique probability distribution out of the learned probabilistic terminology). The procedure is detailed in Algorithm 1.

Algorithm 1. Algorithm for learning probabilistic inclusions

Require: a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a training set E.
Ensure: a new knowledge base $\mathcal{K} =< \mathcal{T}_f, \mathcal{A} >$ with probabilistic inclusions
1: $\mathcal{T}_f = \mathcal{T}$
2: calculate score $score_f$ using E;
3: **for all** concept $C \in \mathcal{T}$ **do**
4: search in $\mathcal{T}$ for the definition of concept C and include its parts in a set H;
5: **if** concept C is subsumed by some other concept D **then**
6: include D in H;
7: **end if**
8: $score_c = score_f$;
9: initialize A_c;
10: **for all** $h \in H$ **do**
11: set $\mathcal{T}_h = \mathcal{T} \cup P(C|h)$;
12: learn probabilities using E;
13: calculate score $score_h$ using E;
14: **if** $score_h > score_c$ **then**
15: $score_c = score_h$
16: $A_c = P(C|h)$
17: **end if**
18: **end for**
19: $\mathcal{T}_f = \mathcal{T}_f \cup A_c$ (insertion restrictions are examined).
20: **end for**

Algorithm 1 receives a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a training set E, composed by interpretations. For each concept in the terminology $\mathcal{T}$, the algorithm searches for its definitions. All subsumed concepts of this definition are considered separately and included in the set H (line 4); the empty and the proper definition are also inserted in H. If an inserted concept is subsumed by another concept, this latter concept is also included in H (line 6). The variable $score_c$ is initialized with the score of $\mathcal{K}$ (line 8), and a variable A_c is initialized as empty (line 9). Each concept included in H is considered separately as

conditioning the concept of interest (line 11). The probability of this new probabilistic inclusion is estimated using the EM algorithm and then the new score is calculated (lines 12 e 13). If this new score improves the current one, the probabilistic inclusion is taken as the current best one and is kept in A_c (line 16). After learning a probability inclusion for each concept the algorithm ends, returning a knowledge base with a new terminology. If the initial score is not improved by neither probability inclusion proposed for a concept, this concept is not associated with any probabilistic inclusion.

After the learning algorithm is run, a pruning routine can verify whether the terminology can be simplified. For example, the inclusion $\mathsf{Researcher} \sqsubseteq \mathsf{Professor}$ can be automatically deleted from the terminology in the presence of probabilistic inclusions $P(\mathsf{Researcher}|\mathsf{Professor}) = 0.4$ and $P(\mathsf{Researcher}|\neg\mathsf{Professor}) = 0.0$.

4 Preliminary Results

Experiments have been run on data extracted from the Lattes Curriculum Platform, a public repository containing data about Brazilian researchers, such as name, address, education, professional experience, areas of expertise. Bibliographic output and participation on examination boards are our main interest. For these experiments, concept descriptions were learned using DL-Learner, available at http://dl-learner.org/Projects/DLLearner. Our algorithm was implemented in the Java language, running in an Ubuntu Linux system with 4GB RAM, 2.4GHZ INTEL CORE 2 DUO. We selected 202 researchers randomly and extracted data such as:

$\mathsf{Researcher}(\mathsf{r1}), \mathsf{Researcher}(\mathsf{r2}), \mathsf{Researcher}(\mathsf{r4}), \ldots$
$\mathsf{wasAdvised}(\mathsf{r8}, \mathsf{r179}), \mathsf{wasAdvised}(\mathsf{r30}, \mathsf{r83}), \mathsf{wasAdvised}(\mathsf{r33}, \mathsf{r1}), \ldots$
$\mathsf{sharePublication}(\mathsf{r1}, \mathsf{r32}), \mathsf{sharePublication}(\mathsf{r4}, \mathsf{r12}), \mathsf{sharePublication}(\mathsf{r5}, \mathsf{r115}), \ldots$
$\mathsf{sameExaminationBoard}(\mathsf{r1}, \mathsf{r32}), \mathsf{sameExaminationBoard}(\mathsf{r4}, \mathsf{r12}), \ldots$
$\mathsf{hasSameInstitution}(\mathsf{r1}, \mathsf{r27}), \mathsf{hasSameInstitution}(\mathsf{r1}, \mathsf{r28}), \ldots$
$\mathsf{advises}(\mathsf{r1}, \mathsf{r33}), \mathsf{advises}(\mathsf{r1}, \mathsf{r171}), \mathsf{advises}(\mathsf{r1}, \mathsf{r81}), \ldots$

We aimed at learning a probabilistic terminology that best describes a set of collaboration patterns among researchers. Learning was performed in two steps. The first step was concerned with learning definitions for concepts (such as NearCollaborator, NullMobilityResearcher, StrongRelatedResearcher) given positive and negative examples. For instance, the pair of individuals r_1 and r_4 could be positive examples of the NearCollaborator concept. Intuitively, a near collaborator is a researcher who shares a publication with a colleague of the same institution. On the other hand, an institution can be interested in asking about mobility of their former students; that is, whether they remain as staff members after finishing their PhDs or not. The concept NullMobilityResearcher is intended to define a researcher with null mobility. The concepts StrongRelatedResearcher and InheritedResearcher define categories of relationships among researchers.

As noted previously, DL-Learner was employed to generate initial concepts based on background knowledge and examples. A preliminary terminology is:

$$
\begin{aligned}
\mathsf{NearCollaborator} &\equiv \mathsf{Researcher} \\
&\quad \sqcap (\exists \mathsf{hasSameInstitution}.\exists \mathsf{sharePublication}. \\
&\quad \exists \mathsf{sharePublication}.\exists \mathsf{advises}.\mathsf{Researcher}) \sqcap \\
&\quad (\exists \mathsf{sharePublication}.\exists \mathsf{hasSameInstitution}. \\
&\quad \exists \mathsf{sharePublication}.\mathsf{Researcher}) \\
\mathsf{FacultyNearCollaborator} &\equiv \mathsf{NearCollaborator} \\
&\quad \sqcap \exists \mathsf{sameExaminationBoard}.\mathsf{Researcher} \\
\mathsf{NullMobilityResearcher} &\equiv \mathsf{Researcher} \\
&\quad \sqcap (\exists \mathsf{hasSameInstitution}.\exists \mathsf{sameExaminationBoard}. \\
&\quad \exists \mathsf{wasAdvised}.\forall \mathsf{advises}.\exists \mathsf{hasSameInstitution}.\mathsf{Researcher}) \sqcap \\
&\quad (\exists \mathsf{wasAdvised}.\exists \mathsf{hasSameInstitution}.\mathsf{Researcher}) \\
\mathsf{StrongRelatedResearcher} &\equiv \mathsf{Researcher} \\
&\quad \sqcap (\exists \mathsf{sharePublication}.\mathsf{Researcher} \sqcap \\
&\quad \exists \mathsf{wasAdvised}.\mathsf{Researcher}) \\
\mathsf{InheritedResearcher} &\equiv \mathsf{Researcher} \\
&\quad \sqcap (\exists \mathsf{sameExaminationBoard}.\mathsf{Researcher} \sqcap \\
&\quad \exists \mathsf{wasAdvised}.\mathsf{Researcher}).
\end{aligned}
$$

The next step focused on learning probabilistic inclusions; Following concept definitions, possible probabilistic inclusions for concept NearCollaborator are:

$$
\begin{aligned}
&P(\mathsf{NearCollaborator} | \mathsf{Researcher} \sqcap (\exists \mathsf{hasSameInstitution}.\exists \mathsf{sharePublication}. \\
&\quad \exists \mathsf{sharePublication}.\exists \mathsf{advises}.\mathsf{Researcher}) \sqcap (\exists \mathsf{sharePublication}. \\
&\quad \exists \mathsf{hasSameInstitution}.\exists \mathsf{sharePublication}.\mathsf{Researcher})) = \alpha, \\
&P(\mathsf{NearCollaborator} | \mathsf{Researcher}) = \beta, \\
&P(\mathsf{NearCollaborator} | \mathsf{Researcher} \sqcap \exists \mathsf{hasSameInstitution}. \\
&\quad \exists \mathsf{sharePublication}.\exists \mathsf{sharePublication}.\exists \mathsf{advises}.\mathsf{Researcher}) = \gamma \\
&P(\mathsf{NearCollaborator} | \mathsf{Researcher} \sqcap \exists \mathsf{sharePublication}. \\
&\quad \exists \mathsf{hasSameInstitution}.\exists \mathsf{sharePublication}.\mathsf{Researcher}) = \theta.
\end{aligned}
$$

Based on a probabilistic score (maximum likelihood on the set of examples), the last candidate was chosen. This choice implies that the original concept description must be removed. The complete probabilistic terminology is:

$$
\begin{aligned}
&P(\mathsf{Researcher}) = 1.0 && P(\mathsf{wasAdvised}) = 0.29 \\
&P(\mathsf{hasSameInstitution}) = 0.83 && P(\mathsf{sharePublication}) = 0.73 \\
&P(\mathsf{sameExaminationBoard}) = 0.41 && \\
&P(\mathsf{NearCollaborator} && | \ \mathsf{Researcher} \sqcap \exists \mathsf{sharePublication}.\exists \mathsf{hasSameInstitution}. \\
& && \exists \mathsf{sharePublication}.\mathsf{Researcher}) = 0.95 \\
&\mathsf{FacultyNearCollaborator} \equiv && \mathsf{NearCollaborator} \\
& && \sqcap \exists \mathsf{sameExaminationBoard}.\mathsf{Researcher} \\
&P(\mathsf{NullMobilityResearcher} && | \ \mathsf{Researcher} \sqcap \exists \mathsf{wasAdvised}. \\
& && \exists \mathsf{hasSameInstitution}.\mathsf{Researcher}) = 0.98 \\
&\mathsf{StrongRelatedResearcher} \equiv && \mathsf{Researcher} \\
& && \sqcap (\exists \mathsf{sharePublication}.\mathsf{Researcher} \sqcap \\
& && \exists \mathsf{wasAdvised}.\mathsf{Researcher}) \\
&\mathsf{InheritedResearcher} \equiv && \mathsf{Researcher} \\
& && \sqcap (\exists \mathsf{sameExaminationBoard}.\mathsf{Researcher} \sqcap \\
& && \exists \mathsf{wasAdvised}.\mathsf{Researcher})
\end{aligned}
$$

We can observe that both StrongRelatedResearch and InheritedResearcher have no probabilistic inclusions. Conversely, NearCollaborator and NulleMobilityResearcher concepts have been replaced by suitable shorter probabilistic inclusions.

To investigate the application of learned concepts, some inferences in this resulting CR$\mathcal{ALC}$ terminology were calculated using an exact (propositionalization) algorithm [3]. When Researcher(0) and Researcher(1)[1] are given as evidence, we obtain $P(\mathsf{NullMobilityResearcher}(0)) = 0.46$. Further evidence changes this value to:

$$P(\mathsf{NullMobilityResearcher}(0)|\exists \mathsf{wasAdvised}(0,1)) = 0.83,$$

and to:

$$P(\mathsf{NullMobilityResearcher}(0)|\exists \mathsf{wasAdvised}(0,1) \sqcap \exists \mathsf{hasSameInstitution}(0,1)) = 0.9.$$

This last probability value indicates that individual 0 can be placed within NullMobilityResearcher. (Note that given the same evidence, the originally learned deterministic ontology would misclassify the individual 0.)

5 Conclusion

We have presented techniques for learning concepts in a probabilistic description logic from relational data. Learning occurs in two steps. In the first step, concept descriptions are learned by existing description logic learning algorithms. The second step searches for probabilistic inclusions that can improve the logical descriptions. The algorithm actually compares logical and probabilistic inclusions, keeping the most accurate ones and discarding the others (where performance is measured using likelihood). The idea is that by letting probabilities leave descriptions that are a little more flexible, we can obtain better models for dealing with real data. Experiments, focused on learning a probabilistic terminology from a real-world domain (the Lattes scientific repository), suggest that probabilistic inclusions do lead to improved likelihoods.

Probabilistic description logics offer expressive languages in which to conduct learning, while charging a relatively low cost for inference. The present contribution offers novel ideas for this sort of learning task; we note that the current literature on this topic is rather scarce. Our future work is to investigate the scalability of our learning methods, with an interest in applications for the semantic web.

Acknowledgements

The second author is supported by CAPES and the third is partially supported by CNPq. The work reported here has received substantial support through FAPESP grant 2008/03995-5.

[1] In these calculations, indexes $0, 1 \ldots n$ represent individuals.

References

1. Antoniou, G., van Harmelen, F.: Semantic Web Primer. MIT Press, Cambridge (2008)
2. Baader, F., Nutt, W.: Basic description logics. In: Description Logic Handbook, pp. 47–100. Cambridge University Press, Cambridge (2002)
3. Cozman, F.G., Polastro, R.B.: Loopy propagation in a probabilistic description logic. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 120–133. Springer, Heidelberg (2008)
4. Cozman, F.G., Polastro, R.B.: Complexity analysis and variational inference for interpretation-based probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence (2009)
5. De Raedt, L. (ed.): Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996)
6. De Raedt, L.: Logical settings for concept-learning. Artificial Intelligence 95(1), 187–201 (1997)
7. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.H. (eds.): Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911. Springer, Heidelberg (2008)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal Royal Statistical Society B 44, 1–38 (1977)
9. Fanizzi, N., D'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
10. Heinsohn, J.: Probabilistic description logics. In: International Conf. on Uncertainty in Artificial Intelligence, pp. 311–318 (1994)
11. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Principals of Knowledge Representation (KR), pp. 461–472 (1994)
12. Jaeger, M.: Relational bayesian networks: a survey. Linkoping Electronic Articles in Computer and Information Science 6 (2002)
13. Lehmann, J.: Hybrid learning of ontology classes. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 883–898. Springer, Heidelberg (2007)
14. Ochoa-Luna, J.E., Cozman, F.G.: An algorithm for learning with probabilistic description logics. In: 5th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 8th International Semantic Web Conference (ISWC), Chantilly, USA, pp. 63–74 (2009)
15. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
16. Sebastiani, F.: A probabilistic terminological logic for modelling information retrieval. In: ACM Conf. on Research and Development in Information Retrieval (SIGIR), pp. 122–130 (1994)

Knowledge-Based System for the Maintenance Registration and Consistency among UML Diagrams

Cleverton Ferreira Borba[1] and Ana Estela Antunes da Silva[2]

[1] Adventist Universitary Center of São Paulo – UNASP
São Paulo – SP - Brazil
[2] Methodist University of Piracicaba - UNIMEP
Piracicaba – SP - Brazil
cleverton.borba@gmail.com, aeasilva@unimep.br

Abstract. This paper highlights the importance of software maintenance, specifically the UML (Unified Modeling Language) diagrams, created and changed, especially during the tasks of analysis and design of software. The main idea of this paper is to formalize the software maintenance phase in order to motivate the maintenance documentation of these diagrams taking into account a knowledge base which represents the consistency among UML diagrams. The consistency among the diagrams is done through a semantic network, and also formalized by the OCL (Object Constraint Language). Finally, the domain knowledge is represented by production rules which form the knowledge base. This knowledge base is the center of the knowledge-based system whose goal is guiding the developer in the maintenance of UML diagrams by recording and making the consistency of these diagrams. Thus the system has two contributions: storage of the maintenance of UML diagrams and diagnosis of consistencies among the diagrams participating in the maintenance phase.

Keywords: Software Maintenance, UML, Knowledge-Based Systems, OCL, Consistency.

1 Introduction

Despite recognizing the importance of the software maintenance phase, there are still difficulties to be overcome such as: cost, time and personnel available [16] [22] so that the life cycle of the software remains complete.

When a maintenance request is made, the code maintenance is mandatory for maintaining software functionality intact. The problem is that the correctness of the code becomes, sometimes, the only function of the maintenance phase. However, other activities of the life cycle should also be treated as part of the maintenance process. Diagrams developed in earlier phases have no proper consideration of the developer for many reasons causing a series of future problems.

Even when applied to the analysis and design phases, the maintenance of the diagrams is not always done in a complete manner. This maintenance should include all the diagrams involved, since the change in a diagram may lead to a change in several others.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 51–61, 2010.

This paper proposes the modeling of a knowledge base that represents the consistency between UML diagrams and formalization by means of production rules that govern the connection among UML diagrams. From these representations, a knowledge-based system is designed to store maintenance records.

This work has as main parts: 1 - Prioritization of maintenance documentation for UML diagrams, 2 - Application of the phases of analysis and design of the IEEE model for the maintenance of UML diagrams, 3 - Application of consistency among UML diagrams in order to keep track of their maintenance recording, 4 - Representation of the domain knowledge through semantic network, OCL and production rules, 5 - knowledge-based system for consistency and storage of maintenance of UML diagrams.

2 Maintenance in the Analysis and Design Phases of Software Development - A Challenge to Developers

According to [14] software products are not thrown into a state of freezing, they have errors and are in constant maintenance. Correcting these detected errors transforms and expands the original application.

More specifically one of the problems involved in software maintenance is the lack of existing documentation involving maintenance of UML diagrams [13]. In [2] the author states that the lack of maintenance of UML diagrams is due to the lack of tools that assist this task.

In this paper, the model of maintenance of the Institute of Electrical and Electronics Engineers (IEEE) [9] was adopted as a paradigm. The IEEE model is concerned with investigating issues related to who, what, when and how the process occurs when a change request is done. This work uses the IEEE model guidelines for recording the diagrams maintenance in the knowledge-based system. The input data for a maintenance request record will be given by:

- Type of maintained diagram: use case diagram, class diagram, sequence diagram, state diagram and activity diagram;
- Applicant: developer who requested the maintenance;
- System: system name to which the diagram is associated;
- Component diagram changed: describes the part of the diagram changed, for example, inclusion of actor in a use case diagram;
- Type of problem occurred: describes the nature of the problem, for example, a problem of the own diagram or due to some changes in the code which promoted a change in some diagram.

In the next section the domain knowledge about the consistency among UML diagrams will be presented.

3 Domain Knowledge

The UML has become one of the most used notations for developing and modeling software. According to [1], the main advantage of UML diagrams is their ability to express many views of design, ranging from structural behavioral, using a single

integrated formalism. Like any other language, the UML comes with a set of syntactic and semantic rules that derives from the understanding and interpretation of models written in this language.

Consistency among UML diagrams has been investigated in many fields and at many levels in the literature. To be able to apply the definition of consistency, there must be an understanding of what needs to be consistent in the context of UML.

One of the issues to be discussed in this paper is how important it is to check the consistency of UML diagrams. First, consistency problems reveal design flaws or misuse of the UML. When these problems are detected at an early phase in the design process, it is easier and more profitable to fix them than if they were discovered at a later stage.

Some authors [8], [10], [11] affirm that consistency among UML diagrams should occur in semantic and syntactic levels. These two levels are part of a meta-model which represents the consistency among UML diagrams. The consistency may occur in an intra-model level and in an inter-model level. In general, most of works concentrates on the need of assuring that consistencies are necessary for the own preservation of diagrams [7].

When consistencies among UML diagrams should be performed, three main categories of tasks may be taken into account [10], [11]:

- Development of a diagram allowing easy accessibility from an element to all other associated elements and objects;
- Valorization of the languages utilized to express constraints in the diagrams;
- Analysis of consistency by transforming diagrams into a more formal representation.

There are tools and systems that perform the consistency analysis on the diagrams expressed in these notations. Unfortunately, most of the UML modeling tools does not incorporate good management practices for consistencies among diagrams. According to [25], there is no work that exposes graphical interfaces, tools, or systems that perform consistency checking among diagrams.

The UML has interdependence among its diagrams. When developing a diagram, others can be generated automatically, depending on the tool, which facilitates and determines the project success. However, when changes occur and these UML diagrams are changed due to correction of errors or changing requirements, automatic changes in other elements and diagrams may be generated. If these updates are not corrected the quality of the diagrams maintenance is ruined.

3.1 Consistency among UML Diagrams

For this study, five of the UML diagrams and the consistency among them were analyzed. Table 1 was generated from observations of relationships among the UML diagrams [5] [6] [7] [8], [10], [11], [17], [20] [21], [24], [25]. Table 1 shows some of the main consistencies between these diagrams. This study considered: use case, class, activities, sequence and state diagrams. It is possible to notice that the main diagonal represents the relationship of the diagram to itself. All other relationships represent the effect of the maintenance of one diagram to some other diagram. For example,

Table 1. Consistency among UML diagrams

	Use Case Diagram	Class Diagram	Sequence Diagram	State Diagram	Activity Diagram
Use Case Diagram	•	All actors shown in the use cases are candidates for classes.	Any use case can be described in the sequence diagram. Any modification of a use case described changes the sequence diagram.	There is no direct link, but indirectly can be affected through the class diagram.	Any use case can be described by an activity. Any modification of a use case requires the maintenance of correspondent activity.
Class Diagram	When a class is modified, correspondent actors should be verified in the Use Case Diagram	•	Messages between the modified classes mean that the corresponding sequence diagram should be amended.	Changes in the internal dynamics of the class indicate that the state diagram must be changed.	Internal and external messages of the class are represented in the activity diagram.
Sequence Diagram	Every scenario described in the sequence diagram should be described in the Use Case Diagram. Any modification of a use case changes the sequence diagram.	Represents the messages exchanged between class instances	•	Sending and receiving messages may be actions of the state diagram. Time interval between two messages is a state in state diagram.	Each scenario of an operation can be shown by an activity diagram.
State Diagram	No direct link.	A state represents the internal dynamics of a class. It explains how the class reacts to external stimuli received.	The actions of the state diagram correspond to sending and receiving messages. States are intervals in the sequence diagram.	•	Actions may change activity flows in the Activity Diagram
Activity Diagram	If a use case is changed, this diagram should be changed.	A modification in the activity diagram may cause the need of a different method in the class diagram.	The change in an activity diagram can represent the change of the sending of an operation in the sequence diagram .	If an activity is changed a state may be created or excluded.	•

when a use case diagram is updated it is possible that a class should be created in the correspondent class diagram.

4 Knowledge Representation

Figure 1 shows the methodology used for the acquisition and representation of knowledge through a semantic network, production rules and OCL formalization. All concepts and relationships among them were obtained from literature as explained in section 3. Associations showed in Table 1 were discovered throughout literature analysis. As this is a domain with a lot of concepts and relationships among them, a semantic network was chosen to represent it. Next section explains the net.

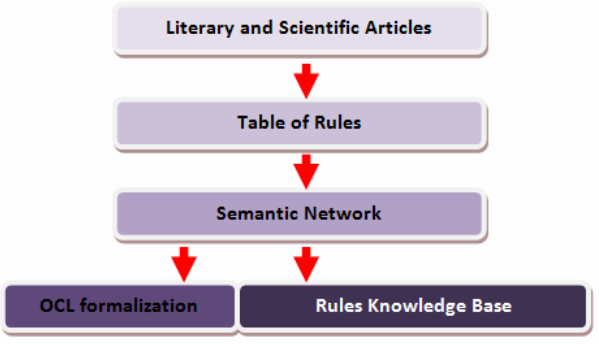


Fig. 1. Tasks performed and models used to represent the domain knowledge

4.1 Semantic Network

Table 1 represents knowledge acquired from the literature. Using Table 1 as a knowledge source a semantic network was developed. This abstraction of knowledge occurs as shown in Figure 2.

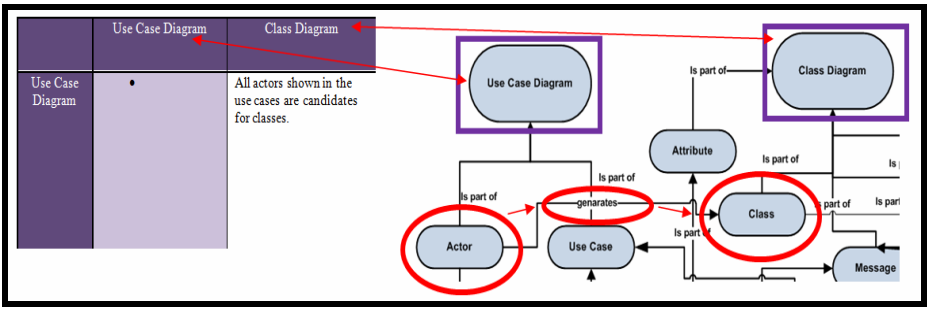


Fig. 2. Construction of the Semantic Network – An Example

For all found consistencies among the diagrams studied, a semantic network node is added. The relationships among nodes are created according to the explanation connecting diagrams in Table 1. Figure 3 shows the whole semantic network. The "part-of" relationship is an important one. It denotes the components of the UML diagrams. The class diagram, for example is composed by: associations, attributes, operations and classes. For each component there are relationships connecting them to either components of the same diagram or components of a different diagram. Another important semantic relationship is "generate". This relationship represents the connection between two concepts and the direction of the relationship implies the sequence in which the concepts will be updated.

4.2 Rules of Consistency among UML Diagrams

In [25], Zapata states that a diagram as a UML class diagram is not detailed enough to provide all the relevant aspects of a specification which causes the need to find a way

to describe these details. At first this drill was conducted in natural language, however the same author states that this method had generated constraints and ambiguity.

Such considerations raise the fact that UML diagrams need other theoretical elements in order to supply all relevant aspects of a specification. Therefore, a first step towards a more precise specification of the UML has been the creation of the textual language OCL (Object Constraint Language) [12].

The consistency found and presented in Table 1 was extracted from the literature and the knowledge acquired was represented through the semantic network, which in turn enabled the description of these consistency rules in a specific language for the UML diagrams. The formalism used was the Object Constraint Language (OCL) [5] [6] and [17].

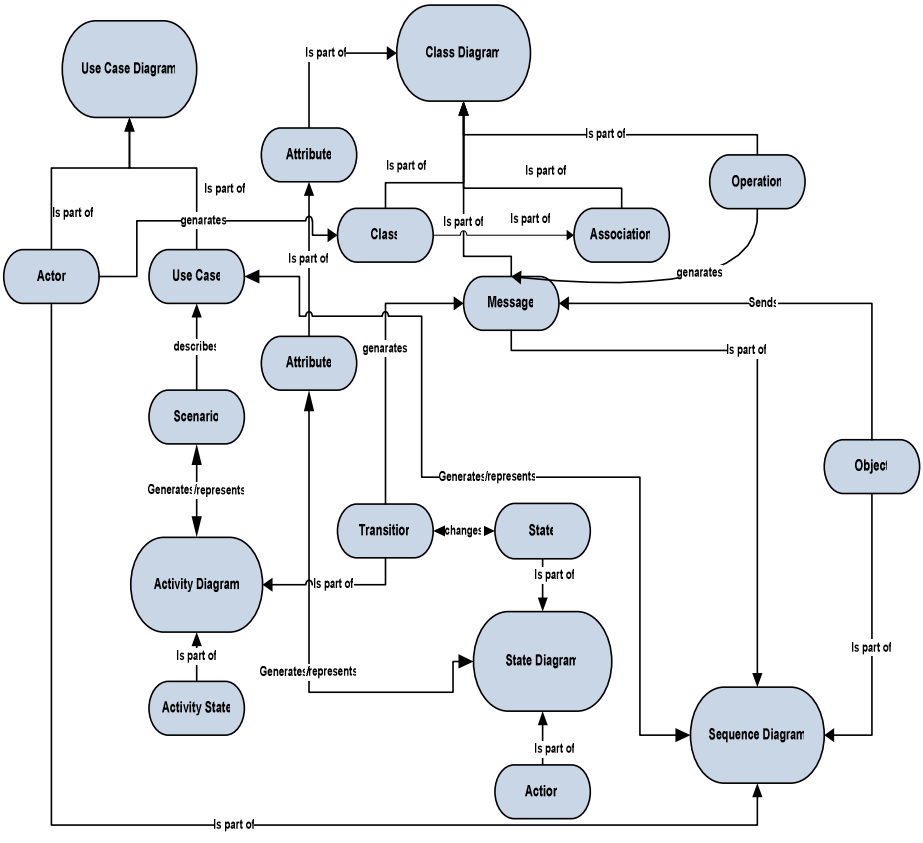


Fig. 3. Semantic Network of Consistencies among UML Diagrams

The example shown in Figure 4 represents the formalization of consistency that involves the "Class" of Class Diagram. The example was constructed taking into account the nodes and relationships defined in the semantic network of Figure 3. The responsibilities were taken from Table 1. The "OCLSpecification" part of the rule shows the objects of other diagrams that should be also updated when a class object of

Contract Maintenance Class Class Diagram
Name: ClassDiagram(Class)
Responsibilities: When performing maintenance class in Class Diagram, change the Class objects involved in other diagrams.
Type: System
Cross References: 1.1
Notes:
Exceptions:
Output:
OCL Specification:
context ClassDiagram (Class)
inv: Maintenance.ClassDiagram.Class = true implies
Maintenance.UseCaseDiagram.Actor = true and
Maintenance. SequenceDiagram.Ator = true

Fig. 4. Example of na OCL specification

a class diagram is created or updated. In this example when a class is updated it also necessary to update the correspondent actor of the use case diagram and the correspondent actor of the sequence diagram.

It is important to notice that the OCL representation allows the connection of each object of each diagram to others because of its syntax. For example the sentence: "maintenance.classdiagram.class=true" means that the object class in the class diagram will be maintained.

4.3 Representation of the Knowledge Base with Production Rules

The predicates and rules which form the knowledge base for the consistency of UML diagrams were developed taken into account the semantic network and the OCL rules shown in Figures 3 and 4, respectively. The predicates´ names are: Class**CD**, Operation**CD**, Attribute**CD**, Association**CD**, Actor**UCD**, UseCase**UCD**, Scenario**UCD**, State**SD**, Transition**SD**, Action**SeqD**, Attribute**SD**, Object**SEQD**, UseCase**SEQD**, Actor**SEQD**, Actor**AD**, State**AD**, Scenario**AD**, Transition**AD**. The suffix of each predicate indicates to which diagram the object belongs to. The prefix indicates the component of the diagram. For example, the ClassCD predicate represents a class in the class diagram. The suffixes are: CD (class diagram), UCD (use case diagram), SD (state diagram), SEQC (sequence diagram) and AD (activity diagram).

The following rules were created:

1. If Class**CD** = YES Then Actor**UCD** = YES and Actor**AD** = YES
2. If Operation**CD** = YES Then Transition**AD** = YES and Object**SEQD** = YES
3. If Attribute**CD** = YES Then Attribute**SD** = YES
4. If Association**CD** = YES Then Transition**AD** = YES and Object**SEQD**
5. If Actor**UCD** = YES Then Class**CD** = YES e Actor**AD** = YES and Actor**SEQD** = YES
6. If UseCase**UCD** = YES Then Scenario**AD** = YES and UseCase**SEQD** = YES
7. If State**SD** = YES Then Transition**SD** = YES
8. If Transition**SD** = YES Then Transition**AD** = YES Class**CD** = YES
9. If Attribute**SD** = YES Then Attribute**CD** = YES

10. If Object**SEQD** = YES Then Class**CD** = YES Object**SEQD** = YES
11. If UseCase**SEQD** = YES Then UseCase**UCD** = YES
12. If Actor**SEQD** = YES Then Actor**UCD** = YES
13. If Scenario**AD** = YES Then Scenario**UCD** = YES
14. If Actor**AD** = YES Then Actor**UCD** = YES
15. If Transition**AD** = YES Then Class**CD** = YES

The knowledge-based system will cover all rules using a forward chaining reasoning in order to reach a solution. The solution of the system consists of the presentation of all diagrams related to the diagram presented for maintenance. Whenever there is a change request for a UML diagram of a particular system the inference engine will be executed.

It is important to notice that each rule is either satisfied or not satisfied before the knowledge-based system check the next rule. The inference engine checks whether the facts of the next rule are true, thus enabling the continuation of the progressive linkage. The reasoning works as following. The left part of the rule is compared with the facts which are currently in the working memory. The rules that meet what is described in memory, have their predicates validated. This means the introduction of new facts into working memory. This method is applied until there are no more rules in the knowledge base to be tested.

Another work which presents the approach of UML diagrams and knowledge-based systems can be found in [26]. This work presents a formal method for UML specifications and consistency using a knowledge base system. The work proposes a tool to map UML specifications in order to verify whether the specifications are correct. These specifications are represented in UML as axioms. Although the cited work presents a model for performing consistencies among UML diagrams it does not intend to use the result as a way to record maintenance requests.

5 Implementation of the Knowledge-Based System

According to [18] and [19], knowledge-based systems are computer programs that make use of knowledge representation in order to solve problems that normally require some kind of specific knowledge to be solved. According to [18] and [19], knowledge bases are containers of knowledge and information, designed to store, share and disseminate expertise. The knowledge base and the inference engine are major components of knowledge-based systems.

Knowledge-based systems organize information so that there is a separation between the base of domain knowledge and knowledge of how to solve the problem.

The inference engine performs the role of the expert about how to solve the problem [23]. For this work the knowledge base used is presented in section 4 by the production rules which represent the domain knowledge of consistencies among UML diagrams. The inference engine reasoning used is forward chaining.

The main functionalities of the system are: register rules of consistency, receive change request, select type of maintenance, register maintenance, enable inference engine, check diagrams involved, register changes in diagrams, record maintenance of consistencies, end change request and query records.

For each maintenance it is necessary to check its connections to other diagrams. This verification is carried out by the set of rules previously registered in the

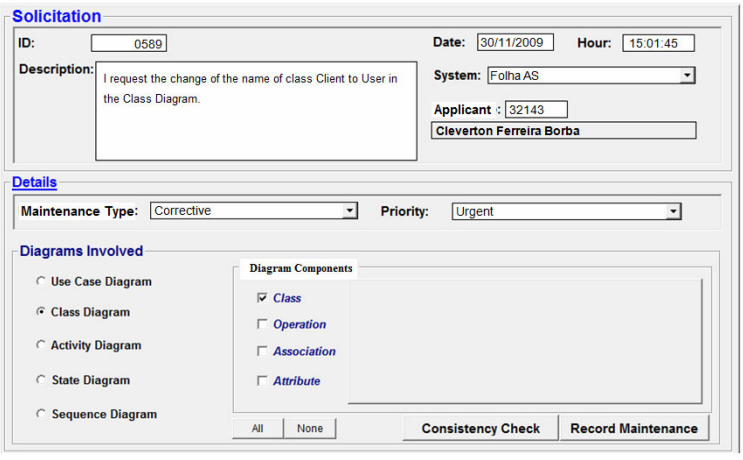


Fig. 5. Main Screen of the Knowledge-Based System

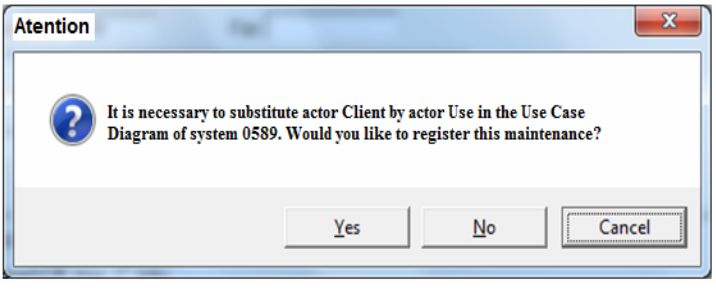


Fig. 6. Screen showing consistencies among maintenances of UML diagrams

knowledge base. Figure 6 shows the main screen of the knowledge-based system developed to record the maintenance of UML diagrams.

All maintenance records registered will form a knowledge base that will assist software designers to manage, correct and predict possible problems in the documentation of UML diagrams. A practical example of the use of the system is the need to maintain the name of a class in a Class Diagram. Following the record of this maintenance is presented step by step (the objects of the models, the systems they belong to and authorized users have already been registered):

1. There is a maintenance request which involves changing the Class Diagram. The system information request and the user applicant are recorded.
2. After registering the diagram which will be updated the user will click the button "Consistency Check". This event will start the execution of the knowledge base in order to find other diagrams and elements which will also need to be maintained.

3. And so the system will request information until all the rules of the knowledge base which are involved with objects of the Class Diagram and their connections have been checked.
4. At the end of the process, the user records all maintenance registers involved in the maintenance process.

The implementation of this knowledge-based system to record the maintenance of UML diagrams tries to inspire in its users the idea of the importance of practicing software maintenance for diagrams. Next section concludes this work.

6 Conclusion

Consistency among UML diagrams is cited by many authors. Some of them also use OCL in order to formalize such process. The need for formalization of consistencies among UML diagrams is also well defined in the literature. This work differs from these authors for proposing, besides the formalization of consistencies, the registration of the maintenance of such consistencies.

A knowledge-based system is developed in order to keep and guide maintenance requests. The instances of this knowledge base can have several purposes. The first one concerns a historical data set of maintenance which can help developers to: analyze which diagrams are being most modified and analyze which system has been most modified. The second purpose is a proposal of future work. The knowledge base could be used to mining records of maintenance in order to find patterns in maintenance records. This result could help developers in several ways. One of them would be to classify possible errors made when developing UML diagrams. This would help preventing corrective maintenance of diagrams. Another important point of the knowledge based system is to establish a habit for developers of maintaining and keeping documentation of maintenance of UML diagrams.

At last, this work presents the combination of two important areas of Computer Science. Artificial Intelligence is presented as a means to support software developers in one specific phase of Software Engineering: the maintenance and consistencies among UML diagrams.

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: Uml – Guia Do Usuário, 10th edn. Editora Campus, Rio Dejaneiro (2000)
2. Dantas, C., Murta, L., Werner, C.: Odyssey-Wi: Uma Ferramenta Para Mineração De Rastros De Modificação Em Modelos Uml Versionados. Disponível Em (2009), http://Reuse.Cos.Ufrj.Br/Prometeus/Publicacoes/Ts%20odyssey-Wi%20final.Pdf
3. Durkin, J.: Expert Systems: Design And Development. Prentice-Hall, New Jersey (1994)
4. Fernandes, A.D.R.S., Monteiro, G.C.S., Guerra, R.S., Castro, S.: OCL: Object Constraint Language. Disponível Em, Porto (2009), http://Paginas.Fe.Up.Pt/~Aaguiar/Es/Artigos%20finais/Es_Final_23.Pdf
5. Ha, I., Kang, B.: Meta-Validation of Uml Structural Diagrams And Behavioral Diagrams With Consistency Rules. Dep't of Computer Engineering, Yeungnam University, Korea, IEEE (2003)

6. Ha, I., Kang, B.: Cross Checking Rules To Improve Consistency Between Uml Static Diagram And Dynamic Diagram. In: Fyfe, C., Kim, D., Lee, S.-Y., Yin, H. (eds.) IDEAL 2008. LNCS, vol. 5326, pp. 436–443. Springer, Heidelberg (2008)
7. Hausmann, J., Sauer, S.: Extended Model Relations With Graphical Consistency Conditions. In: Disponível Em, Kuzniarz, pp. 61–74 (2002)
8. Hnatkowska, B., Huzar, Z., Kuzniarz, L., Tuzinkiewicz, L.: A Systematic Approach To Consistency Within Uml Based Software Development Process. In: Disponível Em, Kuzniarz, pp. 16–29 (2002)
9. IEEE: Standard For Software Maintenance. Software Engineering Standards Committee of The IEEE Computer Society (June 1998)
10. Kuzniarz, L., Reggio, G., Sourrouille, J., Huzar, K.: Workshop on Consistency Problems. In: Uml-Based Software Development, Uml 2002. Blekinge Institute of Technology. Research Report (2002)
11. Kuzniarz, L., Reggio, G., Sourrouille, J., Huzar, K.: Workshop on Consistency Problems. In: Uml-Based Software Development Ii. Uml 2003. Blekinge Institute of Technology. Research Report (2003)
12. OCL, Uml 2.0 Ocl Specification. Disponível Em (2009), http://Www.Omg.Org/Docs/Ptc/03-10-14.Pdf
13. Paduelli, M.M.: Manutenção De Software: Problemas Típicos E Diretrizes Para Uma Disciplina Específica, Usp – São Carlos (2007)
14. Peters, J.F.: Engenharia De Software. Tradução De Ana Patrícia Garcia. Campus, Rio De Janeiro (2001)
15. Pigoski, T.M.: Practical Software Maintenance. John Wiley & Sons, Chichester (1997)
16. Pressman, R.S.: Engenharia De Software. Tradução José Carlos Barbosa Dos Santos, 6th edn. Person Education Do Brasil, São Paulo (2006)
17. Sapna, P.G., Mohanty, H.: Ensuring Consistency In Relational Repository of Uml Models. In: 10th International Conference on Information Technology. IEEE, Los Alamitos (2007)
18. Rich, E., Knight, K.: Inteligência Artificial. Makron Books, São Paulo (1995)
19. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, New Jersey (2003)
20. Schreiber, G.: Knowledge Engineering And Management. MIT Press, Cambridge (1999)
21. Sengupta, S., Bhattacharya, S.: Formalization of Uml Diagrams And Their Consistency Verification - A Z Notation Based Approach. In: ISEC 2008, Hyderabad, India, pp. 151–152 (2008)
22. Sommerville, I.: Engenharia De Software. In: Tradução André Maurício De Andrade Ribeiro, Addison Wesley, São Paulo (2003)
23. Waterman, D.A.: A Guide To Expert Systems. Addison Wesley Publishing Company, New York (1986)
24. Zapata, C.M., González, G., Gelbukh, A.: A Rule-Based System For Assessing Consistency Between Uml Models. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 215–224. Springer, Heidelberg (2007)
25. Zapata, C.M., González, G.: Especificación Formal En Ocl De Reglas De Consistencia Entre Los Diagramas De Clases Y Casos De Uso De Uml Y El Modelo De Interfaces. In: Revista Ingenieriás, Universidad De Medellín, Julio-Dicienbre, Año, vol. 6(12), pp. 169–191 (2008)
26. Zizman, A., Kozlenkov, A.: Knowledge Base Approach to Consistency Management of UML Specifications. In: City University, Department of Computing, London, UK. IEEE, Los Alamitos (2001)

Semantic Mapping with a Probabilistic Description Logic

Rodrigo Polastro, Fabiano Corrêa, Fabio Cozman, and Jun Okamoto Jr.

Escola Politécnica da Universidade de São Paulo

Abstract. Semantic mapping employs explicit labels to deal with sensor data in robotic mapping processes. In this paper we present a method for boosting performance of spatial mapping, through the use of a probabilistic ontology, expressed with a probabilistic description logic. Reasoning with this ontology allows segmentation and tagging of sensor data acquired by a robot during navigation; hence a robot can construct metric maps topologically. We report experiments with a real robot to validate our approach, thus moving closer to the goal of integrating mapping and semantic labeling processes.

1 Introduction

Until a few years ago, robotic mapping tasks were restricted to 2D reconstruction of environments [1]. With recent improvements in sensors and algorithms, researches are now aiming at 3D maps of large environments with ever increased detail, using both laser sensors and images of vision systems [2].

There is current interest in annotating measured structures with semantically meaningful labels such as "building" or "tree" [3]. The classification of data in such categories allows dimensionality reduction in mapping related problems [4], as well as the construction of maps that are useful beyond navigation (for instance, useful for task planning [5]). Work on this topic is grouped under the name of semantic mapping. As stated by Hertzberg and Saffiotti [6], two points must be present in applications to fully use semantic knowledge in robotics:

1. an explicit representation of knowledge available to the robot; that is, an ontology for the domain of interest;
2. the need for grounding the symbols used in this representation in physical objects that can be detected by robot sensors.

Even though several proposals claim to be using semantics in robotics when automatically classifying sensor data in categories, if those categories are not related to each other, there is no semantic reasoning involved.

Galindo et al [5] have presented one of the few proposals that really explore semantics in depth. They provide an ontology for indoor environments and use it to reason during task planning. Two ways of exploiting semantics are investigated: deduction of new information, and improvement in task planning efficiency. They

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 62–71, 2010.

combine a spatial hierarchy with a conceptual hierarchy expressed with a description logic. One hierarchy is related to the another through anchoring [7].

Limketkai et al [8] propose a more integrated mapping process, although they only explore semantics to construct a map from laser data. They use a first-order probabilistic model based on Relational Markov Networks [9] to classify lines processed from laser sensor data, so as to identify doors and corridors in the environment. Spatial and relational information between the objects in scene are used to facilitate the discrimination. Wang and Domingos [10] apply Markov Logic networks (with extensions to handle continuous variables) to that same domain.

Regarding only sensor data classification, some authors propose different methods to cluster sensor data with the objective of creating a topological map of the environment. Posner et al [11] consider as a scene whatever data a robot acquires with its sensors, and shows how to classify outdoor scenes (they use odometry information to provide continuity in the classification). Zivkovic et al [12] use omnidirectional images of indoor environments and propose a method that identifies images that composes different rooms, constructing thus a topological map.

Vasudevan et al [13] use identification of certain objects and their spatial position in the environment to create a map of objects. They detect doors with a laser sensor and use SIFT [14] to detect other objects from images.

In this paper, we propose a combination of traditional methods in robotics to find known objects in images and to register 3D points, while using a probabilistic description logic to relate datasets with different areas of the environment. This way we can split sensor data in small clusters, map them separately, and them assemble them together on a topologic semantic map. Thus moving towards the goal of adding high-level abilities to robotic navigation.

2 Representing Environments in CR$\mathcal{ALC}$

This paper introduces a method for boosting spatial mapping through reasoning over objects contained in the environment. To achieve that, relationships between the environment and the objects in it should be first modeled. We resort to a *description logic* for such modeling [15]. Description logics are largely used to build ontologies, as they usually contain a fragment of first-order logic and can organize concepts into hierarchies. Such a logic seems to be the right tool in the present context, as we need a language to describe high-level labels; for instance, an "office" must be described as a set of "chairs", "tables" and "computers" in some structured manner.

The challenge is that there is usually uncertainty attached to descriptions of contexts in robotics; in particular, there is always uncertainty associated with sensor data. Besides, in practice two instances of the same label are often distinct; that is, two different offices contain chairs and tables, but the quantity of this objects is unique to each office; moreover one of them may have vending machines for its employees while the other may not. As another example, two parks may

have completely different vegetation but both still need to be labeled by a single concept "park". Standard description logics cannot model alone these matters. We thus resort to a *probabilistic description logic*; that is, a description logic that allows probabilities to be attached to its formulae.

2.1 Credal CR$\mathcal{ALC}$

Cozman and Polastro have recently proposed a probabilistic description logic, called CR$\mathcal{ALC}$ [16,17], that adopts an interpretation-based semantics and resorts to graph-theoretical tools so as to allow judgements of stochastic independence to be expressed. This logic was chosen over other probabilistic description logics [18,19,20,21,22,23,24] as it is based in the popular $\mathcal{ALC}$ logic and attends to our needs.

The vocabulary of CR$\mathcal{ALC}$ contains *individuals*, *concepts*, and *roles* [15]. Concepts and roles are combined to form new concepts using a set of *constructors* from $\mathcal{ALC}$ [25]. These constructors are *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$) and *value* restriction ($\forall r.C$). A *concept inclusion* is denoted by $C \sqsubseteq D$ and a *concept definition* is denoted by $C \equiv D$, where C and D are concepts.

CR$\mathcal{ALC}$ also supports probabilistic inclusions. A probability inclusion reads $P(C|D) = \alpha$, where D is a concept and C is a concept name. The semantics of such a probabilistic inclusion is:

$$\forall x : P(C(x)|D(x)) = \alpha, \tag{1}$$

where it is understood that probabilities are over the set of all interpretation mappings $\mathcal{I}$ for a domain $\mathcal{D}$. If D is the "true" concept $\top$, then we simply write $P(C) = \alpha$. Probabilistic inclusions are required to only have concept names in their conditioned concept (that is, inclusions such as $P(\forall r.C|D)$ are not allowed).

We also allow assessments such as $P(r) = \beta$ to be made for roles, with semantics

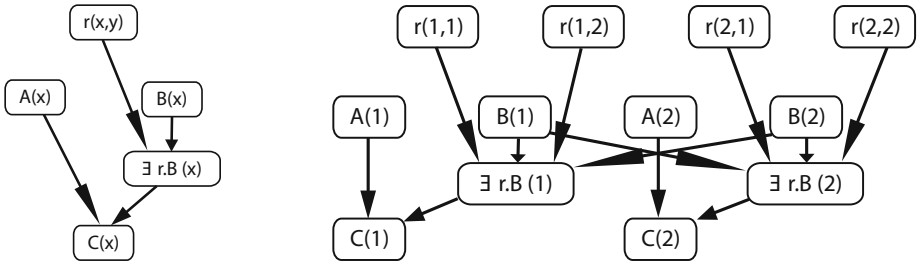
$$\forall x, y : P(r(x, y)) = \beta, \tag{2}$$

where again the probabilities are over the set of all interpretation mappings.

We assume that every terminology is acyclic; that is, a concept does not use itself. Under some additional restrictions (unique-name assumption, precision of probability assessments, known and finite domain), any terminology in CR$\mathcal{ALC}$can be grounded into a Bayesian network [16,17]. For better understanding of the grounding, consider the concepts A, B, C and the role r, and suppose $P(A) = 0.7$, $B \sqsubseteq A$, $P(B|A) = 0.4$, $P(r) = 0.5$ and $C \equiv A \sqcap \exists r.B$. In Figure 1 we have: a) ontology graph; b) the grounded network for 2 individuals.

2.2 An Ontology for the Domain of Spatial Mapping

To characterize the status of a robot's location with the information of its sensors, we contribute proposing a probabilistic ontology as follows. We start with two primitive concepts $\mathsf{Object}(x)$ and $\mathsf{Environment}(x)$. As CR$\mathcal{ALC}$ requires that a



(a) Ontology graph. (b) Grounding.

Fig. 1. Graph for a simple ontology and its grounding for a domain with 2 individuals

priori probabilities must be specified for primitive concepts; as there is no prior information on objects, we assign relatively neutral probabilities as follows:

$$P(\mathsf{Object}) = 0.5,$$
$$P(\mathsf{Environment}) = 0.5.$$

We introduce two roles, one to express that an environment contains an object, and the other to express that two objects are near. We leave the probabilities for these roles rather neutral, using the same probabilities:

$$P(\mathsf{contains}) = 0.5,$$
$$P(\mathsf{near}) = 0.5.$$

We propose the following object hierarchy, using resources in CR$\mathcal{ALC}$:

$$\begin{aligned}
\mathsf{InteriorObject} &\sqsubseteq \mathsf{Object},\\
\mathsf{ExteriorObject} &\sqsubseteq \mathsf{Object},\\
\mathsf{OfficeObject} &\sqsubseteq \mathsf{Object},\\
\\
\mathsf{Table} &\sqsubseteq \mathsf{InteriorObject},\\
\mathsf{Chair} &\sqsubseteq \mathsf{InteriorObject},\\
\\
\mathsf{Cabinet} &\sqsubseteq \mathsf{InteriorObject} \sqcap \mathsf{OfficeObject},\\
\mathsf{Monitor} &\sqsubseteq \mathsf{InteriorObject} \sqcap \mathsf{OfficeObject},\\
\mathsf{Printer} &\sqsubseteq \mathsf{InteriorObject} \sqcap \mathsf{OfficeObject},\\
\\
\mathsf{Sign} &\sqsubseteq \mathsf{ExteriorObject},\\
\mathsf{Extinguisher} &\sqsubseteq \mathsf{ExteriorObject},\\
\mathsf{Switchbox} &\sqsubseteq \mathsf{ExteriorObject},
\end{aligned}$$

$$\begin{aligned} \mathsf{Door} &\sqsubseteq \mathsf{Object}, \\ \mathsf{Board} &\sqsubseteq \mathsf{Object}. \end{aligned}$$

Note that $B \sqsubseteq A$ implies $P(B|\neg A) = 0$ but it implies nothing about $P(B|A)$ remaining $P(B|A) \in [0, 1]$; whenever this happens we adopt a uniform distribution over possible objects.

Composite objects can now be described:

$$\begin{aligned} \mathsf{Desk} &\equiv \mathsf{Table} \sqcap \exists \mathsf{near}.\mathsf{Chair}, \\ \mathsf{Entrance} &\equiv \mathsf{Door} \sqcap \exists \mathsf{near}.\mathsf{Sign}. \end{aligned}$$

Finally, the whole environment can be described as:

$$\begin{aligned} \mathsf{Room} \equiv\ & \mathsf{Environment} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Door} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Table} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Chair} \\ & \sqcap\ \neg\exists \mathsf{contains}.\mathsf{ExteriorObjects} \\ \mathsf{Office} \equiv\ & \mathsf{Room} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Desk} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Cabinet} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Monitor} \\ \mathsf{Classroom} \equiv\ & \mathsf{Room} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Board} \\ & \sqcap\ \neg\exists \mathsf{contains}.\mathsf{OfficeObjects} \\ \mathsf{Hallway} \equiv\ & \mathsf{Environment} \\ & \exists \mathsf{contains}.\mathsf{Entrance} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Extinguisher} \\ & \sqcap\ \exists \mathsf{contains}.\mathsf{Switchbox} \\ & \sqcap\ \neg\exists \mathsf{contains}.\mathsf{InteriorObjects} \end{aligned}$$

The graph representing this ontology is presented in Figure 2.

The network generated by this ontology is relatively dense due to the presence of quantifiers, and the computational effort needed to calculate probabilities is not trivial. Although exact inference is theoretically possible, there are reliable algorithms for approximate inference that seem more adequate in practice. In particular, Loopy Propagation "breaks" the network connections, replacing local probability distribution with more convenient functions.

3 Semantic Mapping

Mapping larger environments requires dealing with huge amounts of data. Another difficulty in mapping, particularly in 3D mapping, is data association; that

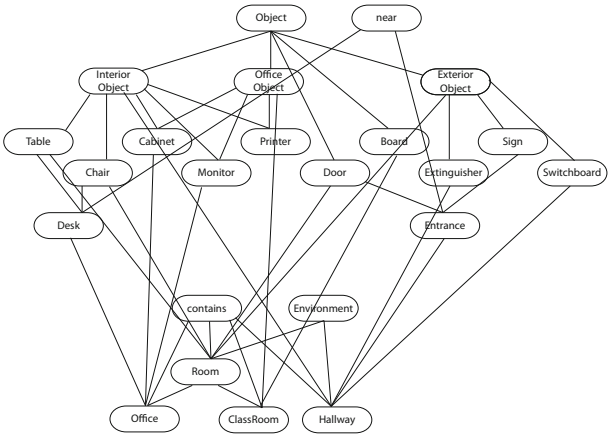


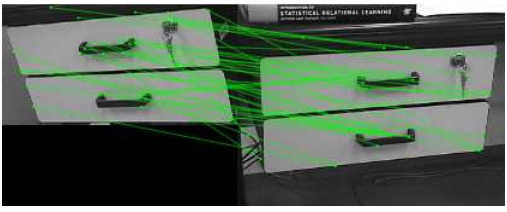
Fig. 2. Graph for the proposed ontology

is, sensor readings from two different positions must be registered into a common coordinate system. It is desirable to separate all the data in various groups, labeled as ceiling, wall, or perhaps kitchen or office.

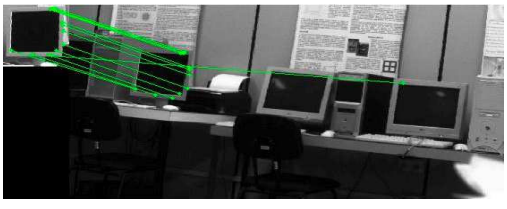
Our proposed scheme for semantic mapping is an offline process. The robot navigates through the environment, collecting images, laser and odometry data in several positions, and then the data are processed at once. The idea is to identify objects in the images collected by a robot, and to classify the locations in which those images were taken. Then, we use the laser and odometry data related to those images to map smaller areas of the environment. For instance, data of an office let us reconstruct spatially only the office; using odometry information it becomes possible to determine a topology of the environment, and then to unify all metric maps in a single one through the detected doors.

The classification of objects in images is done using a previous set of features, robustly extracted using the SIFT algorithm. The objects present in the ontology have a SIFT descriptor previously computed. Figure 3 shows two templates of objects in the matching with scenes acquired by the robot, and the robot Pioneer 3-AT mounted with a SICK laser to obtain 3d data and a camera to recognize objects in images, used in the experiments.

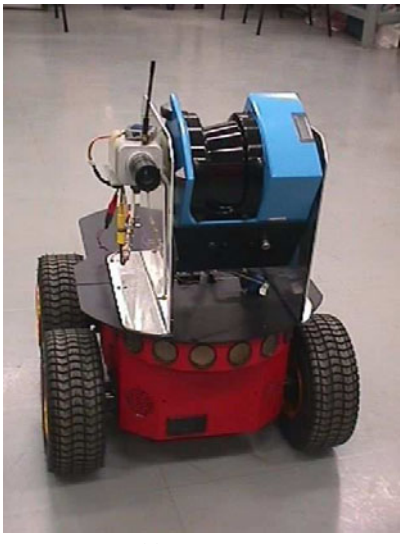
As the objects in the images are identified, the probabilistic description logic model is used to label sensor data. Inferences are conducted in the probabilistic graph instantiated by the CR$\mathcal{ALC}$ model, thus producing probabilities for the types of environments. Not all sensor are labeled as some images may contain objects not matched against SIFT descriptors. But using the principle of continuity, sensor data not labelled between two or more with identical labels received the same label.



(a) Desk.



(b) Monitor.



(c) Pioneer AT-3.

Fig. 3. Correspondence between template objects and scenes acquired by a mobile robot

4 Experiments

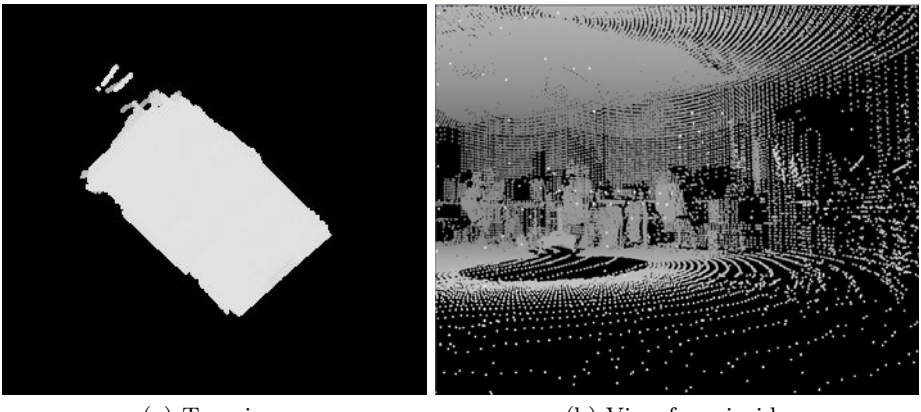
Our primary goal in doing these experiments is to evaluate the ontology and the reasoning procedure with CR$\mathcal{ALC}$. Both must be appropriate to deal with real robotic problems and sensor data.

We provide some results of an experiment consisting in the robot navigating through three different areas of an indoor environment: two labs and a hallway connecting both. In this experiment, we gathered some images, 3D points from a laser sensor, and the estimated pose in each data gathering location given by odometry and a gyroscope. We determine a priori a set of objects that could be easily identified through SIFT features and used them to characterize different areas.

Following the path of the robot, that goes from a laboratory to the other, passing by the hallway, we picked sequentially 6 points to gather data from the laser and camera, two in each area. Each point was then classified as Office, Classroom or Hallway, accordingly to the identified objects in the images using SIFT; the result of the reasoning can be seen in Table 1. Whenever two sequential points have different labels, the data are split into a new area. In this case, we found 3 different areas. Table 1 shows the inferred values for each point. Note that in the ontology the possible environments were not set as mutually exclusive; hence the probabilities are for individual objects and are not required to add to one across objects.

Table 1. Identified ambients

Datapoints						
Location	1	2	3	4	5	6
Observations						
Objects	3 chairs 1 table 1 monitor 1 cabinet 1 door	2 chairs 2 tables 1 monitor 1 cabinet 1 door	3 doors 2 signs 1 extinguisher 1 switchbox	2 doors 2 signs 1 extinguisher 1 switchbox 1 board	2 chairs 1 table 2 monitors 1 cabinet 1 door	2 chairs 1 table 1 monitor 2 cabinets 1 door
Inferred Probability for Each Label						
P(Office)	**0.4278**	**0.4258**	0.0000	0.0000	**0.4280**	**0.4280**
P(Classroom)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
P(Hallway)	0.0000	0.0000	**0.1578**	**0.1619**	0.0000	0.0000
Area	1		2		3	



(a) Top-view. (b) View from inside

Fig. 4. An example of a 3d metric map of a given area

Mapping each identified area alone, we could map the 3 environments. Figure 4 shows the 3d map from one of the areas, a laboratory. Figure 4(a) is a top-view from the laboratory and Figure 4(b) is a view from inside the laboratory.

5 Conclusion and Future Work

Semantic knowledge can have a significant impact in robotics. In this paper, we show how to model a robotic mapping task with a probabilistic description logic, and we show how to process sensor data so as to reason about the identity of specific areas in an indoor environment. The main objective of this process is to automatically segment the data acquired by a robot, so that the map could

be constructed topologically, thus providing a way of scale the mapping process to larger environments. This objective is said to be achieved, once we managed to automatically split the data in convenient smaller and tagged sets, each one being mapped alone.

Instead of using a SIFT algorithm to provide labels, we might have used an SVM classifier, or perhaps a Conditional Random Field, with better accuracy and generalization. We plan to integrate the image labeling process with the probabilistic description logic in the near future.

Acknowledgments

The first author is supported by FAPESP process 2008/53292-0. The second author is supported by FAPESP process 2009/14396-8. The third author is partially supported by CNPq. The project has been partially supported by FAPESP process 2008/03995-5.

References

1. Thrun, S.: Exploring Artificial Intelligence in the New Millennium. In: Robotic mapping: a survey, pp. 1–36. Morgan Kaufmann Publishers, San Francisco (2003)
2. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building rome in a day. In: IEEE 12th International Conference on Computer Vision, pp. 72–79 (2009)
3. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.: Discriminative learning of Markov random fields for segmentation of 3d scan data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 169–176 (2005)
4. Nuechter, A.: 3D Robotic Mapping. Springer Tracts in Advanced Robotics (2009)
5. Galindo, C., Fernndez-Madrigal, J.-A., Gonzlez, J., Saffiotti, A.: Robot task planning using semantic maps. Robotics and Autonomous Systems 11, 955–966 (2008)
6. Hertzberg, J., Saffiotti, A.: Using semantic knowledge in robotics. Robotics and Autonomous Systems 56, 875–877 (2008)
7. Coradeschi, S., Saffiotti, A.: An introduction to the anchoring problem. Robotics and Autonomous Systems 43, 85–96 (2003)
8. Limketkai, B., Liao, L., Fox, D.: Relational object maps for mobile robots. In: Proceedings of the International Joint Conference on Artificial Intelligence, vol. 1, pp. 1471–1476 (2005)
9. Taskar, B., Abbeel, P., Wong, M.-F., Koller, D.: Relation Markov Networks. In: Introduction to Statistical Relational Learning, pp. 175–199. MIT Press, Cambridge (2007)
10. Wang, J., Domingos, P.: Hybrid Markov logic networks. In: Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 2, pp. 1106–1111 (2008)
11. Posner, I., Schroeter, D., Newman, P.: Using scene similarity for place labeling. In: Proceedings of the 10th International Symposium on Experimental Robotics (2006)
12. Zivkovic, Z., Booij, O., Krse, B.: From images to rooms. Robotics and Autonomous Systems 55, 411–418 (2007)

13. Vasudevan, S., Gchter, S., Nguyen, V., Siegwart, R.: Cognitive maps for mobile robots - an object based approach. Robotics and Autonomous Systems 55, 359–371 (2007)
14. Lowe, D.G.: Distinctive image features from scale-invariant key-points. International Journal of Computer Vision 60, 91–110 (2004)
15. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook. Cambridge University Press, Cambridge (2002)
16. Cozman, F.G., Polastro, R.B.: Loopy propagation in a probabilistic description logic. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 120–133. Springer, Heidelberg (2008)
17. Polastro, R.B., Cozman, F.G.: Inference in probabilistic ontologies with attributive concept descriptions and nominals. In: 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 7th International Semantic Web Conference, ISWC (2008)
18. Ding, Z., Peng, Y., Pan, R.: BayesOWL: Uncertainty modeling in semantic web ontologies. In: Soft Computing in Ontologies and Semantic Web. Studies in Fuzziness and Soft Computing, vol. 204, pp. 3–29. Springer, Heidelberg (2006)
19. Heinsohn, J.: Probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence, pp. 311–318 (1994)
20. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Principles of Knowledge Representation (KR), pp. 461–472 (1994)
21. Koller, D., Levy, A.Y., Pfeffer, A.: P-CLASSIC: A tractable probablistic description logic. In: AAAI, pp. 390–397 (1997)
22. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172, 852–883 (2008)
23. Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Conference on Formal Ontology in Information Systems (2006)
24. Sebastiani, F.: A probabilistic terminological logic for modelling information retrieval. In: Croft, W., Rijsbergen, C.V. (eds.) 17th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 122–130. Springer, Dublin (1994)
25. Schmidt-Schauss, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48, 1–26 (1991)

Markov Decision Processes from Colored Petri Nets

Monica Góes Eboli and Fabio Gagliardi Cozman

Escola Politécnica da Universidade de São Paulo
Av. Prof. Mello Moraes 2231
monica.eboli@gmail.com, fgcozman@usp.br

Abstract. Models that are suitable for planning are not always easy to specify. In this paper we investigate the conversion of Petri nets into factored Markov decision processes: the former are relatively easy to build while the latter are adequate for policy generation. To represent probabilities that are needed when planning under uncertainty, we introduce factored Petri nets; we then describe the conversion of factored Petri nets in Markov decision processes.

Keywords: Planning under uncertainty, Markov decision processes, Colored Petri nets.

1 Introduction

This paper describes a method that converts colored Petri nets into factored Markov decision processes. The goal is to associate the modeling power of Petri nets with the planning power of Markov decision processes. We are motivated by the observation that manufacturing plants need to plan their production mixes under uncertainty, and the easiest way to model these plants is through Petri nets.

Petri nets can represent various types of resources through colored marks, and can indicate the input of resources through source transitions. The stochastic behavior of those nets is induced by conflicts among resources. Alas, Petri nets do not lend themselves easily to policy generation. Our main contribution is a method that transforms a factored Petri net into a factored Markov decision process, as these processes can easily generate policies. We also contribute by introducing factored Petri nets (with a suitable Markov condition) and efficient encodings for probability tables.

We offer some background in Section 2; Sections 3 and 4 present respectively factored Petri Nets and our proposed conversion method, together with some experiments to evaluate the method. Section 5 concludes the paper.

2 Background

2.1 Petri Nets

Petri nets (PNs) offer tools to model and to analyze discrete event systems, by representing such systems graphically [1,2,3]. A Petri net has the following elements:

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 72–81, 2010.

- A set of places $P = \{p_1, p_2, \ldots, p_n\}$, represented graphically by circles;
- A set of transitions $T = \{t_1, t_2, \ldots, t_m\}$, represented graphically by rectangles;
- A set of directed arcs $A = \{a_1, a_2, \ldots, a_h\}$, represented by arrows;
- A set of marks (tokens), represented by dots inside circles.

Places represent passive elements of the system, usually denoting a specific type of resource in a particular stage. Transitions are the active elements and represent actions of the system. Directed arcs represent the relationships between places and transitions, defining either the resources that are needed for a transition, and outcomes of a transition. Tokens are located inside the places and represent the existence of a unit of the resource denoted by the place. The marking of a Petri net is the placement of tokens over places.

Petri nets evolve over time by the firing of the transitions. When a transition is fired, tokens are moved from input places to output places. Transitions in Petri nets can have two states, *enabled* and *disabled*. To be fired, a transition must be enabled, meaning its pre-conditions and post-conditions must be satisfied. Pre-conditions refer to the input places; the input places can have tokens indicating the resources for the desired operation are available. Post-conditions refer to output places, usually indicating physical space for the products of the operation.

An enabled transition may not fire when there is a conflict, leading to nondeterministic behavior in the PN. A conflict occurs when two enabled transitions compete for the same resource (token) in the pre-conditions or compete for space in the post-condition. Those two types of conflicts can be seen in Figure 1. To model these conflicts one must define a *random switch*; that is, a firing probability is associated with each possible transition. The specification of these probabilities is given in Section 3.

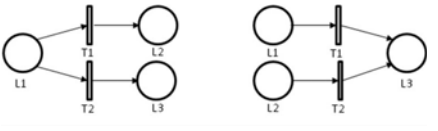


Fig. 1. Example of conflicts

2.2 Colored Petri Nets

Colored Petri nets let tokens be distinct [4,5,6,7], thus easing the modeling effort (as a smaller number of places is typically needed, when compared to standard Petri nets). In this subsection two types of colored Petri nets are shown, one in which the arcs contain *fixed inscriptions* and one in which the arcs contain *variable inscriptions*. Colored Petri nets with fixed inscriptions consist of [7]:

- Places, transitions and directed arcs as in a standard Petri net;
- Pre-/post-conditions as in a standard Petri net;
- Colored marks that flow though the net as tokens;
- An inscription in each directed arc, indicating a type of token.

The evolution of colored Petri nets is as follows.

- A transition t is enabled if every place P of its pre-conditions contains the items indicated by the inscription of the arcs from P to t. The post-conditions affect the net as before.
- When an enabled transition t fires:
  - For each place P of the pre-conditions of t, the indicated item in the arc from P to t is removed from P;
  - Every place P' of the post-conditions of t receives the item indicated in the arc from t to P'.

Figure 2 depicts a place-transition net, representing the heating of an oven and the baking of bread and cake. In Figure 4 the same system is represented by a colored Petri net with fixed inscriptions. Both examples represent the same system but the second model is simpler.

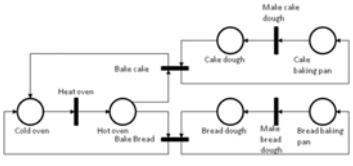


Fig. 2. Example of place transition net

Colored Petri nets with variable inscriptions consist of:

- Places, transitions, directed arcs and an initial configuration of individual items as in the nets with fixed inscriptions;
- Pre-/post-conditions defined as for nets with fixed inscriptions;
- Variables associated with inscriptions.

The evolution of these nets is as follows.

- All the arcs with the same variable should receive the same value;
- A transition t is enabled if all the values that should replace the value of the arc are present in its pre-conditions;
- The firing of an enabled transition t happens in such a way that:
  - The token is removed from all places that belong to the pre-conditions of t as determined by the variable substitution;

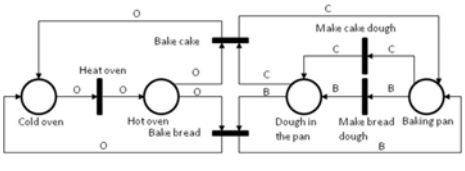


Fig. 3. Example of colored Petri nets with fixed inscriptions

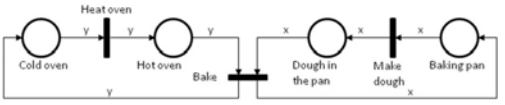


Fig. 4. Example of colored Petri nets with variable inscriptions

- Tokens are placed in all places that belong to the post-conditions of t as determined by the variable substitution.

In Figure 4, the same system modeled in Figure 2 and 3 is modeled with a smaller number of transitions and places.

2.3 Markov Decision Processes

The main elements of a Markov Decision Process (MDP) [9,8,10] are:

- a nonempty set of states S;
- a nonempty set of actions $A(s)$ representing the available actions in state s;
- a state transition function $F(s, a)$ mapping state s and action a into a nonempty set of states (where $|F(s, a)| \geq 1$);
- a positive reward $R(s, a)$ for taking a in s;
- a probabilistic distribution $P(\cdot|s, a)$ over $F(s, a)$ for all $s \in S$, $a \in A(s)$.

A complete description of the system (that is, a state) must have all the information needed by the decision agent to make a decision. We assume the state space, containing all states that can be reached, to be finite. Actions are interventions that the decision maker can perform in the system to change its evolution. There must be a function $F(s, a)$ that denotes the states that can be reached from the present state through a given action. For each state s and action a, a probability distribution over the next state is given. An MDP possesses the Markov property: the next state depends only on the present state (not on previous nor future states).

A *factored Markov decision process* is an MDP where states are not explicity listed. State variables $V = [V_1, V_2, \ldots, V_n]$ are given and any state is represented by a tuple $[v_1, v_2, \ldots, v_n]$, where v_i is a value of V_i. To specify probability distributions over the states, one must exploit independence relations among these variables; factorizations are important because they reduce the size of the model.

3 Factored Petri Nets

In this section we propose the novel concept of *factored Petri nets (FPNs)*, a variant of colored Petri nets that contain probabilities in factored form. Factored Petri nets are attactive because they can be easily turned into MDPs, as explained later. In an FPN: (i) all places have capacity $k = 1$; (ii) it is mandatory

to have at least one source transition (a transition without pre-conditions) that can provide a finite number of types of marks; and (iii) the arcs can only contain either a fixed value or a variable. A FPN is thus defined as follows:

- A nonempty set of places $P = \{p_1, p_2, \ldots, p_n\}$;
- A nonempty set of possible marks for each place $M = \{m_1, m_2, \ldots, m_z\}$;
- A nonempty set of transitions $T = \{t_1, t_2, \ldots, t_m\}$;
- A nonempty subset of T, $Tf = \{tf_1, tf_2, \ldots, tf_y\}$, that contains source transitions;
- A nonempty set of marks $M(tf_i) = \{mf_{i1}, .., mf_{ij}\}$, that can enter into the system through the sources;
- A nonempty set of directed arcs $A = \{a_1, a_2, \ldots, a_w\}$, each connecting a transition t_i to a place p_j or a place p_j to a transition t_i (and whose inscription may be a mark or a variable);
- A set of firing probabilities for each transition in a random switch;
- A nonempty set of rewards $R = \{r_1, r_2, \ldots, r_l\}$, associated with the Petri net states.

The evolution of states follows the same rules as for a colored Petri net, meaning that the future state of the net depends only on the present state. Only the present state is needed to obtain the evolution of the system: the firing probabilities depend only on the present state of the net.

We need a definition to proceed: The *conflict group* for a transition contains all the transitions that affect its firing probabilities, not only the ones that are directly concurrent.

We assume the firing probability of a transition to depend on:

- Places that are pre- and post-conditions of the transition.
- Places that are pre- and post-conditions of transitions in the conflict group.

We further assume that conflict groups are stochastically independent. To understand this assumption, consider the content of each place to be an independent variable and associate with each transition a binary variable that indicates the firing or not firing of the transition. The variables associated with the firing of a transition belonging to a conflict group are independent of the variables associated with transitions in another conflict group. It is important to note that this is only valid between different conflict groups; events in the same conflict group are not assumed independent.

Probabilities and rewards of a FPN are given in appropriate tables, as discussed in the following sections. One of the contributions of this paper is exactly to develop a compact way to represent such tables.

3.1 Probability Tables

The obvious way to represent the probabilities in an FPN is to list all possible states and to associate with each transition in each state a probability value. We now propose a more compact representation.

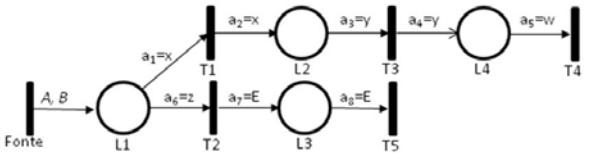


Fig. 5. Example of factored Petri nets

First, post-conditions only matter because they determine whether transitions are enabled or disabled. Hence, post-conditions do not need to be listed when specifying probabilities. If a transition is enabled we must specify its firing probability (even if it is zero); if the transition is disabled, we need not further specify any probability. If the firing transitions probabilities depend only on a few of the listed pre-conditions, the ones that do not matter need not be written in a table. The probability table of the conflict group in Figure 5 is in Table 1.

Table 1. Probability table for example in Figure 5

L1	T1	T2
A	0.6	0.4
B	0.7	0.3

Table 2. Reward table for example in Figure 5

L3	L4	Reward
Empty	A	1
Empty	B	2
A	Empty	1
A	A	2
A	B	3
B	Empty	2
B	A	3
B	B	4

3.2 Reward Tables

All the states that affect rewards must be listed in a reward table; as in probability tables, only the places that define those states need to be listed. That is, if the value of a reward is independent of the contents of a place, that place need not be present in the reward table. An example using the net in Figure 5 is depicted in Table 2. In this example, $L1$ and $L2$ do not affect the reward, hence they are not listed in the table.

4 From Factored Petri Nets to Factored Markov Decision Processes

Our proposed conversion from FPNs to MDPs has six steps:

1. Create state variables.
2. Associate those variables with their possible values.
3. Define actions.
4. Define dependences between variables.
5. Calculate transition probabilities for all variables.
6. Define rewards in the MDP.

First, for each place in the Petri net, a variable is created. That is, for each place L_i there is an equivalent variable named V_i. The next step is to specify values for variables, because not all types of marks are possible in all places. The third step is to define actions. Each source transition must be analyzed: an action consists of the combination of what is input into the system by each source. If there is more than one source, their actions must be combined to determine the action of the system.

The next step is to determine the dependences between variables. A variable in a stage depends only on variables in the previous stage that are pre and post-conditions of the transitions of the conflict groups to which the variable belongs to.

From the present state of the variables, we must determine the probability of reaching a particular value of a variable. Consider a variable V_1 that may have the values v_{11}, v_{12}, and empty. To calculate the probabilities of $\{V_1 = v_{11}\}$, the following rules must be used:

1. If $V_1 = v_{12}$, then $P(V_1 = v_{11}) = 0$.
2. If $V_1 = v_{11}$, then $P(V_1 = v_{11}) = 1 - \sum_j P(TE1_j)$, where $P(TE1_j)$ is the firing probability of the transitions that have V_1 as a pre-condition.
3. If $V1$ = empty, then $P(V_1 = v_{11}) = \sum_j P(TS1_j)$, where $P(TS1_j)$ is the firing probability of the transitions that have V_1 as a post-condition.

All probabilities just listed can be obtained from the probability tables; in case they are not given explicitly they can be inferred by the state of the net. Transitions not associated with conflicts either have a probability one of firing (are enabled) or have a probability zero of firing (are disabled).

The last step is specifying rewards in the MDP. By definition, the reward is given by a reward table; the FPN reward table must be converted to a reward table using MDP variables, simply by replacing places by their associated variables.

These operations are summarized in Algorithm 1. The algorithm's correctness is now proved:

Theorem 1. *Every FPN can be converted into a factored MDP.*

Algorithm 1. Converting an FPN into an MDP

Input: FPN
Output: MPD

1 Convert all the places in the FPN into state variables in the MDP;
2 Determine dependences among variables;
3 Convert the probabilities table by replacing the places and transitions by the MDP variables;
4 Complete the probability tables with the states that yield deterministic values (zero or one).
5 Associate the probability for each combination of V given the present state E defined by the variables on which V is dependent;
6 Convert the reward table from places in the FPN into variables of the MDP.

Proof: Given a state E in an FPN, a unique state V of MDP can be obtained:

1. For every place L_i a variable V_i is generated.
2. For every state of the FPN, a variable v_i has the same value as markings on its corresponding place; therefore an unique state $[v_1, v_2, \ldots, v_n]$ is obtained.

Given the set of source transitions T_f and the set of marks that can be generated by them $M(tf_i)$ of the FPN, a set of actions F of the MDP is generated.

1. For every combination of input marks from the source $[mf_{i1}, mf_{i2}, \ldots, mf_{ij}]$, where j is the number of source transitions, an action f_i belonging to the set of actions of the MDP is generated.
2. The set of all actions f_i composes the set of all actions F of the MDP.

Given the set of transitions firing probabilities dependent on the state of the Petri net $L(E_i) = \{l_1, l_2, \ldots, l_q\}$, the set of probabilities $P(V_i) = \{pv_1, pv_2, \ldots, pv_q\}$ of the MDP is generated.

1. For every variable that represents the firing probability p_j of the state E_i, a variable pv_j is obtained, representing the probability on the state V_i.
2. For all probabilities in the FPN, pv_j gets the value of p_j, hence there are unique sets of probabilities in a state $P(V_i)$: $[pv_1, pv_2, \ldots, pv_q]$.

Given a reward set W in FPN, a set of rewards R_v of MDP is built.

1. For every reward w of the state e_i, a reward r_v of the state v_i is built.
2. For every state in the FPN, the variable r_v receives the value of r, therefore there is an unique set of rewards $R_v = [rv_1, rv_2, ..., rv_n]$ equivalent to R.

QED

Because this work was motivated by the idea of planning manufacturing systems with stochastic behavior, four cases were created and tested, to measure the amount of time needed to generate policies using a recently developed method [11]. Plans for test cases com Figure 6 were generated within reasonable processing times. The test cases 1 and 2 took less than a second to be solved, and test cases 3 and 4 took three and thirteen seconds respectively.

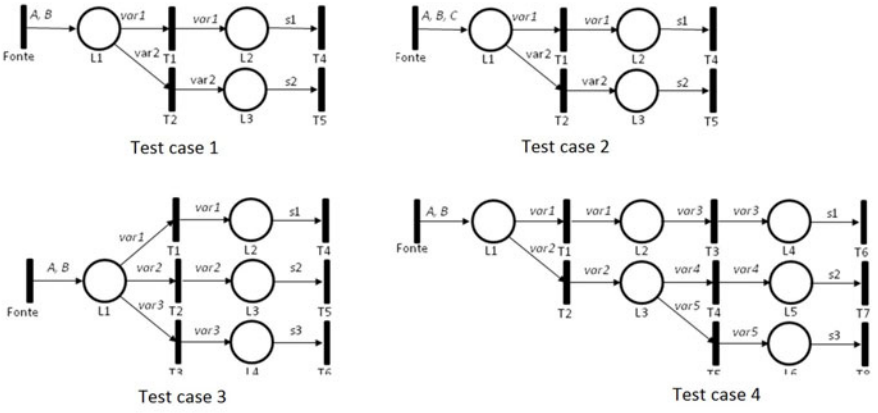


Fig. 6. Test cases

5 Conclusion

We have proposed a few techniques that simplify planning tasks by letting systems to be modeled by a popular and easy-to-understand language based on Petri nets, and by converting the resulting models to Markov decision processes so as to generate policies. We have introduced factored Petri nets; the contribution there is the identification of independence assumptions that can be assumed between conflict groups. The second contribution is the compact encoding of probability tables; we note that explicit encoding leads to exponentially large tables, so the gains are nontrivial. Finally, the main contribution of the paper is the conversion method from FPNs to MDPs. We are currently examining variants of FPNs that let probabilities be imprecisely specified; that is, that let probability intervals be given instead of point probabilities.

Acknowledgement

This project has been partially supported by FAPESP process 2008/03995-5 and the second author has been partially supported by CNPq.

References

1. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modeling with Generalized Stochastic Petri Nets. John Wiley and Sons, Chichester (1995)
2. Zhou, M., Venkatesh, K.: Modeling, Simulation and Control of Flexible Manufacturing Systems – A Petri Net Approach. World Scientific, Singapore (1999)
3. Hass, P.J.: Stochastic Petri nets for modeling and simulation. In: Winter Simulation Conference, Washington, pp. 101–112 (2004)

4. Jensen, K.: An introduction to the theoretical aspects of colored Petri Nets. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1993. LNCS, vol. 803, pp. 230–272. Springer, Heidelberg (1994)
5. Jensen, K.: An introduction to the practical use of colored Petri nets, pp. 237–292. Springer, Berlin (1998)
6. Jensen, K., Kristensen, L.M., Cristensen, S.: The Practioners Guide to Coloured Petri Nets, pp. 98–132. Springer, Berlin (1998)
7. Miyagi, P.E.: Controle Programavel. Editora Edgard Blcher (1996)
8. Russel, S., Norvig, P.: Inteligencia Artificial. Editora Campus (2004)
9. Puterman, M.L.: Markov Decision Processes. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York (1994)
10. Trevisan, F.W., Cozman, F.G., Barros, L.N.: Planning under risk and Knightian uncertainty. In: International Joint Conference on Artificial Intelligence, pp. 2023–2028 (2007)
11. Delgado, K.V., Barros, L.N., Cozman, F.G., Shirota, R.: Representing and solving factored Markov decision processes with imprecise probabilities. In: 6th International Symposium on Imprecise Probability: Theories and Applications, Durham, United Kingdom, pp. 169–178 (2009)

Incremental Learning of Multivariate Gaussian Mixture Models

Paulo Martins Engel and Milton Roberto Heinen

UFRGS – Informatics Institute
Porto Alegre, CEP 91501-970, RS, Brazil
{engel,mrheinen}@inf.ufrgs.br

Abstract. This paper presents a new algorithm for unsupervised incremental learning based on a Bayesian framework. The algorithm, called IGMM (for Incremental Gaussian Mixture Model), creates and continually adjusts a Gaussian Mixture Model consistent to all sequentially presented data. IGMM is particularly useful for on-line incremental clustering of data streams, as encountered in the domain of mobile robotics and animats. It creates an incremental knowledge model of the domain consisting of primitive concepts involving all observed variables. We present some preliminary results obtained using synthetic data and also consider practical issues as convergence properties discuss future developments.

Keywords: Incremental Learning, Unsupervised Learning, Bayesian Methods, Expectation-Maximization Algorithm, Finite Mixtures, Clustering.

1 Introduction

The Gaussian Mixture Model is a statistical modeling tool that has been successfully used in a number of important problems involving both pattern recognition tasks of supervised classification and unsupervised classification. In supervised classification, an observed pattern, viewed as a D-dimensional feature vector, is probabilistic assigned to a set of predefined classes. In this case, the main task is to discriminate the incoming patterns based on the predefined class model. In unsupervised classification, the classes are not predefined but are learned based on the similarity of patterns. In this case, the recognition problem is posed as a categorization task, or clustering, consisting in finding natural groupings (i.e. clusters) in multidimensional data, based on measured similarities among the patterns. Unsupervised classification is a very difficult problem because multidimensional data can form clusters with different shapes and sizes, demanding a flexible and powerful modeling tool.

In this paper, we focus in the so called unsupervised incremental learning [1,2], which considers building a model, seen as a set of concepts of the environment describing a data flow, where each data point is just instantaneously available to the learning system [3,4]. In this case, the learning system needs to take into account these instantaneous data to update its model of the environment. An important issue in unsupervised incremental learning is the stability-plasticity dilemma, i.e., whether a new presented data point must be assimilated in the current model or cause a structural change in the model to accommodate the new information that it bears, i.e., a new concept.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 82–91, 2010.

Although concept formation has a long tradition in machine learning literature, in the field of unsupervised learning, most methods assume the same restrictions pointed out above for probabilistic modeling [4]. The well known *k-means* algorithm [5,6], for instance, represents a concept as a mean of a subset or cluster of data. In this case, each data point must deterministically belong to one concept. The membership of a data point to a concept is decided by the minimum distance to the means of the concepts. To compute the means, we average all data points belonging to every concept, the number of concepts being fixed along all the learning process. For learning probabilistic models, the reference is the EM algorithm that follows the mixture distribution approach for probabilistic modeling explained above [7,6]. This algorithm proceeds in two steps: an *estimation* step (E) that computes the probabilistic membership (the *posterior probability*) of every data to each component of the mixture model based on a current hypothesis (a set of parameters), followed by a *maximization* step (M) that updates the parameters of the current hypothesis based on the maximization of the *likelihood* of the data. Here the number of concepts is fixed and must be known at the start of the learning process. The parameters of each distribution are computed through the usual statistical point estimators, a batch-mode approach which considers that the complete training set is previously known and fixed [6].

As the EM algorithm, IGMM also follows the mixture distribution modeling. However, its model can be effectively *expanded* with new components (i.e. concepts) as new relevant information is identified in the data flow. Moreover, IGMM adjusts the parameters of each distribution after the presentation of every single data point according to recursive equations that are approximate incremental counterparts of the batch-mode update equations used by the EM algorithm. In our attempt to solve the problem of unsupervised incremental learning, we face then two issues: how to tackle the stability-plasticity dilemma and how to update the values of distribution parameters as new data points are sequentially acquired. IGMM handles the stability-plasticity dilemma by means of a novelty criterion based on the likelihood of the mixture components.

The main contributions of this paper are the incremental recursive equations, derived from the Robbins-Monro stochastic approximation method [8], and the novelty criterion used to overcome the problem of the model complexity selection. Although in the past several attempts have been made to create an algorithm to learn Gaussian mixture models incrementally [9,10,11,12,13,14], most part of these attempts requires several data points to the correct estimation of the covariance matrices and/or does not handles the stability-plasticity dilemma. The IGMM algorithm, on the other hand, converges after the presentation of few training samples and does not require a predefined number of Gaussian distributions. The rest of this paper is organized as follows. Section 2 presents in details the proposed algorithm, called IGMM. Section 3 describes several experiments where the proposed model is evaluated and compared with the EM algorithm. Finally, Section 4 provides some final remarks and perspectives.

2 The Incremental Gaussian Mixture Model

This section describes the proposed model, called IGMM (standing for Incremental Gaussian Mixture Model), which was designed to learn Gaussian mixture models from

data flows in an incremental and unsupervised way. IGMM assumes that the probability density of the input data $p(\mathbf{x})$ can be modeled by a linear combination of component densities $p(\mathbf{x}|j)$ corresponding to independent probabilistic processes, in the form

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)p(j) \tag{1}$$

This representation is called a *mixture model* and the coefficients $p(j)$ are called the mixing parameters, related to the *prior* probability of $\mathbf{x}$ having been generated from component j of the mixture. The priors are adjusted to satisfy the constraints

$$\sum_{j=1}^{M} p(j) = 1 \tag{2}$$

$$0 \leq p(j) \leq 1 \tag{3}$$

Similarly, the component density functions $p(\mathbf{x}|j)$ are normalized so that

$$\int p(\mathbf{x}|j)d\mathbf{x} = 1 \tag{4}$$

The probability of observing vector $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_D)$ belonging to the jth mixture component, is computed by a multivariate normal Gaussian, with mean $\boldsymbol{\mu}_j$ and covariance matrix $\mathbf{C}_j$:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \mathbf{C}_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j)\right\} \tag{5}$$

IGMM adopts an incremental mixture distribution model, having special means to control the number of mixture components that effectively represent the so far presented data. We are interested in modeling environments whose overall dynamics can be described by a set of persistent concepts which will be incrementally learned and represented by a set of mixture components. So, we can now rely on a novelty criterion to overcome the problem of the model complexity selection, related to the decision whether a new component should be added to the current model. The mixture model starts with a single component with unity prior, centered at the first input data, with a baseline covariance matrix specified by default, i. e., $\boldsymbol{\mu}_1 = \mathbf{x}^1$, meaning the value of $\mathbf{x}$ for $t = 1$, and $(\mathbf{C}_1)^1 = \sigma_{ini}^2\mathbf{I}$, where σ_{ini} is user-specified configuration parameter.

New components are added by demand. IGMM uses a *minimum likelihood* criterion to recognize a vector $\mathbf{x}$ as belonging to a mixture component. For each incoming data point the algorithm verifies whether it minimally fits any mixture component. A data point $\mathbf{x}$ is not recognized as belonging to a mixture component j if its probability $p(\mathbf{x}|j)$ is lower than a previously specified *minimum likelihood-* (or *novelty-*) *threshold*. In this case, $p(\mathbf{x}|j)$ is interpreted as a *likelihood function* of the jth mixture component. If $\mathbf{x}$ is rejected by all density components, meaning that it bears new information, a new component is added to the model, appropriately adjusting its parameters. The novelty-threshold value affects the sensibility of the learning process to new concepts,

with higher threshold values generating more concepts. It is more intuitive for the user to specify a minimum value for the acceptable likelihood, τ_{nov}, as a *fraction* of the maximum value of the likelihood function, making the novelty criterion independent of the covariance matrix. Hence, a new mixture component is created when the instantaneous data point $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_D)$ matches the *novelty criterion* written as

$$p(\mathbf{x}|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \quad \forall j \tag{6}$$

An instantaneous data point that does not match the novelty criterion needs to be assimilated by the current mixture distribution, causing an update in the values of its parameters due to the information it bears. IGMM follows an incremental version for the usual iterative process to estimate the parameters of a mixture model based on two steps: an estimation step (E) and a maximization step (M). The update process begins computing the posterior probabilities of component membership for the data point, $p(j|\mathbf{x})$, the *estimation* step. These can be obtained through Bayes' theorem, using the current component-conditional densities $p(\mathbf{x}|j)$ and priors $p(j)$ as follows:

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{j=1}^{M} p(\mathbf{x}|j)p(j)} \quad \forall j \tag{7}$$

The posterior probabilities can then be used to compute new estimates for the values of the mean vector $\boldsymbol{\mu}_i^{new}$ and covariance matrix $\mathbf{C}_j^{new}$ of each component density $p(\mathbf{x}|j)$, and the priors $p^{new}(j)$ in the *maximization* step. Next, we derive the recursive equations used by IGMM to successively estimate these parameters.

The parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)^T$, corresponding to the means, $\boldsymbol{\mu}_j$, covariances matrices, $\mathbf{C}_j$, and priors $p(j)$ of a mixture model involving D-dimensional Gaussian distributions $p(\mathbf{x}|j)$, can be estimated from a data sequence of t vectors, $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n, \ldots, \mathbf{x}^t\}$ assumed to be drawn independently from this mixture distribution. The estimates of the parameters are random vectors whose statistical proprieties are obtained from their joint density function. Starting from an initial "guess", each observation vector is used to update the estimates according to a successive estimation procedure.

IGMM follows the Robbins-Monro stochastic approximation method to derive the recursive equation used to successively estimate the priors [8]. For this, in the maximization step the parameters of the current model are updated based on the maximization of the likelihood of the data.

In this case, the *likelihood* of $\boldsymbol{\theta}$ for the given $\mathbf{X}$, $L(\boldsymbol{\theta})$, is the joint probability density of the whole data stream $\mathbf{X}$, given by

$$L(\theta) \equiv p(\mathbf{X}|\theta) = \prod_{n=1}^{t} p(\mathbf{x}^n|\theta) = \prod_{n=1}^{t} \left[\sum_{j=1}^{M} p(\mathbf{x}^n|j)p(j) \right] \tag{8}$$

The technique of maximum likelihood sets the value of $\boldsymbol{\theta}$ by maximizing $L(\boldsymbol{\theta})$.

Although the maximum likelihood technique for estimating the priors is straightforward, it becomes quite complex when applied to estimate the mean vector and the covariance matrix directly from (5). Instead, we follow the natural conjugate technique

to estimate these parameters [15]. When $\boldsymbol{\mu}$ and $\mathbf{C}$ are estimated by the sample mean vector and sample covariance matrix, and $\mathbf{X}$ is a normally distributed random vector, the joint density function $p(\boldsymbol{\mu}, \mathbf{C}|\mathbf{x}^1, \ldots, \mathbf{x}^i, \ldots, \mathbf{x}^n)$ is known to be the reproducible Gauss-Wishart distribution, the natural conjugate density for the model of (5) [15]. In this case, when we estimate both the expected vector and the covariance matrix of a single distribution, starting with a priori distribution with an expected vector $\boldsymbol{\mu}^0$ and covariance matrix $\mathbf{C}^0$, these parameters are transformed through n observations in the following manner [15, 16]:

$$\omega^1 = \omega^0 + n \qquad v^1 = v^0 + n$$

$$\boldsymbol{\mu}^1 = \frac{\omega^0 \boldsymbol{\mu}^0 + n \langle \mathbf{X} \rangle}{\omega^0 + n} \tag{9}$$

$$\mathbf{C}^1 = \frac{\left(v^0 \mathbf{C}^0 + \omega^0 \boldsymbol{\mu}^0 \left(\boldsymbol{\mu}^0\right)^T\right) + n \langle \mathbf{X} \rangle \langle \mathbf{X} \rangle^T - \omega^1 \boldsymbol{\mu}^1 \left(\boldsymbol{\mu}^1\right)^T}{v^0 + n} \tag{10}$$

where ω^0 and v^0 reflect the confidence about the initial estimates of $\boldsymbol{\mu}^0$ and $\mathbf{C}^0$ respectively, corresponding to the number of samples used to compute these initial estimates.

On the other hand, when the probability density of the input data is a Gaussian Mixture Model with M components, an observation $\mathbf{x}^t$ is probabilistic assigned to a distribution j by the corresponding posterior probability $p(j|\mathbf{x}^t)$. In this case, the equivalent number of samples used to compute the parameter estimates of the jth distribution component corresponds to the sum of posterior probabilities that the data presented so far were generated from component j, the so called 0*th-order moment of* $p(j|\mathbf{x})$ *over the data*, or simply the 0*th-order data moment for* j. IGMM stores this summation as the variable sp_j which is periodically restarted to avoid an eventual saturation.

The recursive equations used by IGMM to update the model distributions are:

$$sp_j = sp_j + p(j|\mathbf{x}) \tag{11}$$

$$\boldsymbol{\mu}_j = \boldsymbol{\mu}_j + \frac{p(j|\mathbf{x})}{sp_j} (\mathbf{x} - \boldsymbol{\mu}_j) \tag{12}$$

$$\mathbf{C}_j = \mathbf{C}_j - (\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^{old})(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^{old})^T + \frac{p(j|\mathbf{x})}{sp_j} \left[(\mathbf{x} - \boldsymbol{\mu}_j)(\mathbf{x} - \boldsymbol{\mu}_j)^T - \mathbf{C}_j\right] \tag{13}$$

$$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q} \tag{14}$$

where $p(j|\mathbf{x})$ $\boldsymbol{\mu}_j^{old}$ refers to the value of $\boldsymbol{\mu}_j$ at time $t-1$ (i.e., before updating). One important property of these update equations is the fact that they continuously compute a instantaneous approximation of the parameters that represent the mixture distribution.

The IGMM algorithm has just two configuration parameters, σ_{ini} and τ_{nov}. The σ_{ini} parameter is not critical – its only requirement for σ_{ini} is be large enough to avoid singularities. In our experiments we have simply used $\sigma_{ini} = (\mathbf{x}_{max} - \mathbf{x}_{min})/10$. The τ_{nov} parameter, on the other hand, is more critical and must be defined carefully. It indicates how distant $\mathbf{x}$ must be from $\boldsymbol{\mu}_j$ to be consider a non-member of j. For instance,

$\tau_{nov} = 0.01$ indicates that $p(\mathbf{x}|j)$ must be lower than one percent of the Gaussian height (probability in the center of the Gaussian) for $\mathbf{x}$ be considered a non-member of j. If $\tau_{nov} < 0.01$, few pattern units will be created and the regression will be coarse. If $\tau_{nov} > 0.01$, more pattern units will be created and consequently the regression will be more precise. In the limit, if $\tau_{nov} = 1$ one unit per training pattern will be created.

3 Experiments

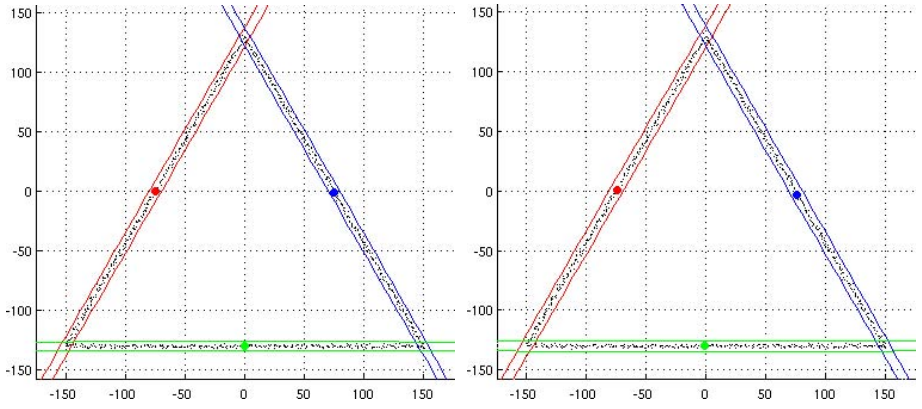
This section describes the experiments accomplished with the proposed model to evaluate its performance and to compare it with the traditional batch-mode EM algorithm. In these experiments, we assess how close a mixture model, generated by IGMM after a single presentation of the training set (just one iteration over the training data), matches the model computed by means of the EM algorithm after 100 iterations over the same data. The configuration parameters used in all experiments are $\tau_{nov} = 0.01$ and $\sigma_{ini} = (\mathbf{x}_{max} - \mathbf{x}_{min})/10$. It is important to say that no exhaustive search was performed to optimize these parameters.

The first experiment simulates the trajectory performed by a mobile robot (position (x, y) at each instant) in an environment with three corridors disposed in a triangular way. Although in real mobile robotics we can rarely count on the actual instant position of the robot, these data are used because the resulting computed models are more understandable than the more realistic multidimensional sensor data of a real robot, used in later experiments. The training data (900 (x, y) pairs describing the robot trajectory in the environment) were added with a standard Gaussian noise vector. Table 1 shows the results obtained in this experiment.

The first lines of Table 1 describe the actual distribution parameters, the following lines the distribution parameters computed by EM algorithm and the last lines show the model parameters computed by IGMM. Observing Table 1 it can be noticed that the

Table 1. Model parameters computed for the triangular dataset

Actual parameters of the distribution						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3333	-74.84	0.41	1885.7	3262.2	5647.7
Cluster 2	0.3333	75.29	-0.41	1879.2	-3257.8	5651.3
Cluster 3	0.3333	-0.07	-129.95	7536.2	-1.0	0.9
Model parameters estimated by EM algorithm						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3324	-75.22	-0.27	1869.1	3232.6	5595.2
Cluster 2	0.3353	75.12	-0.13	1896.0	-3286.5	5700.4
Cluster 3	0.3323	-0.18	-129.94	7467.1	-0.1	0.9
Model parameters estimated by IGMM						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3375	-74.43	1.05	1916.0	3310.7	5726.3
Cluster 2	0.3346	76.70	-2.88	1873.9	-3251.3	5646.4
Cluster 3	0.3279	-0.94	-129.95	7197.1	-0.1	1.2



(a) Actual model (b) IGMM model

Fig. 1. Concepts computed using the triangular dataset

parameters incrementally computed by IGMM are very similar to those computed by the batch-mode EM algorithm.

Fig. 1 shows the data points and the model parameters obtained in this experiment. Fig. 1(a) shows the actual distribution parameters, and Fig. 1(b) shows the distribution parameters computed by IGMM. Each cluster in Fig. 1 was drawn using a different color. It can be noticed that the distributions are very elongated, but again IGMM was capable to align the model components with the actual principal component axes.

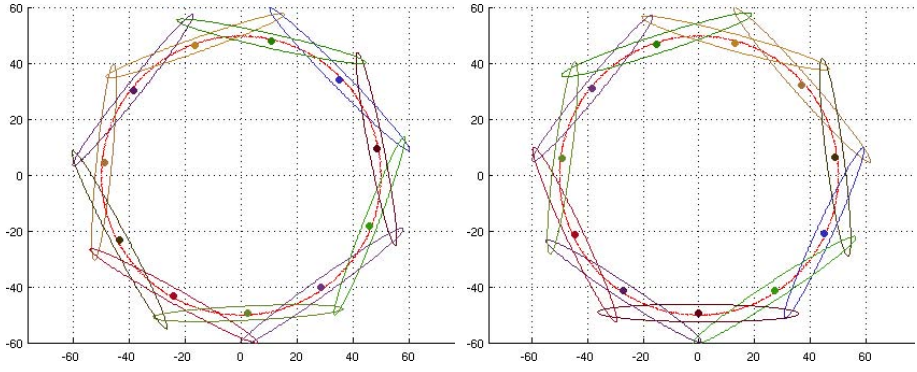
The next experiment describes a circular trajectory of a mobile robot in the $x - y$ plane. This kind of dataset is much more difficult to cluster because: (i) the clusters' boundaries are not clear (there are no abrupt changes in directions); (ii) multivariate Gaussian distributions do not perfectly fit the training data; and (iii) the number of clusters affects the quality of the model (to obtain a good fit, many clusters are necessary, but in this case the generalization may be poor). Thus, it is not possible to know in advance the actual distribution parameters, because the generated clusters depend on the granularity of the model (adjusted using τ_{nov} configuration parameter) and the initial point of the trajectory. The training dataset (1000 (x, y) pairs describing a circular trajectory) was added with a standard Gaussian noise vector. Table 2 shows the model parameters computed in this experiment using EM algorithm and IGMM, and Fig. 2 shows these results graphically.

It is important to say that the clusters computed by IGMM must not necessarily be the same computed by EM algorithm, because the absolute positions of the IGMM clusters will depend on the initial point in the trajectory. The batch-mode EM algorithm, on the other hand, may generate clusters in other absolute positions depending on the initialization. Nevertheless, it can be noticed that the cluster relative positions are very similar in both algorithms. Moreover, IGMM was able to correctly estimate the multivariate data distributions, with model parameters similar to those obtained with the batch-mode EM algorithm.

The next experiment is similar to the previous one, but in this case the robot follows a sinusoidal trajectory. This dataset is interesting because: (i) the direction changes are

Table 2. Model parameters computed for the circular dataset

	Model parameters estimated by EM						Model parameters estimated by IGMM					
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cl. 1	0.0901	10.77	48.13	63.31	-14.06	3.51	0.0907	13.07	47.55	62.22	-16.92	5.13
Cl. 2	0.0929	35.19	34.51	34.67	-34.93	36.06	0.0962	36.97	32.51	33.18	-36.95	42.48
Cl. 3	0.0917	48.38	9.58	2.93	-12.89	66.03	0.0932	48.86	6.43	1.72	-9.00	69.31
Cl. 4	0.0892	45.96	-17.97	8.93	21.91	56.32	0.0877	44.81	-20.68	11.37	23.64	51.69
Cl. 5	0.0942	28.70	-40.03	48.01	34.05	24.85	0.0911	27.08	-41.18	47.80	30.99	20.88
Cl. 6	0.0884	2.36	-49.28	64.06	3.00	0.50	0.0936	-0.26	-49.25	71.50	-0.44	0.54
Cl. 7	0.0893	-24.11	-43.08	49.67	-27.53	15.74	0.0871	-27.05	-41.33	43.13	-27.92	18.72
Cl. 8	0.0937	-43.57	-23.01	16.04	-29.48	56.20	0.0893	-44.58	-21.10	12.55	-25.39	54.12
Cl. 9	0.0922	-49.07	4.82	1.08	6.72	68.57	0.0898	-48.91	6.32	1.64	8.56	65.57
Cl. 10	0.0899	-38.68	30.64	25.79	31.97	40.58	0.0874	-38.20	31.25	26.01	30.99	38.09
Cl. 11	0.0884	-16.50	46.52	56.61	19.94	7.41	0.0939	-15.17	46.88	64.49	20.68	7.24



(a) EM model (b) IGMM model

Fig. 2. Concepts computed using the circular dataset

soft, which requires to solve the stability-plasticity dilemma; and (ii) the x values are linear and the y values are cyclical, and this may difficult the learning process. For this experiment, 1000 training data were generated according to:

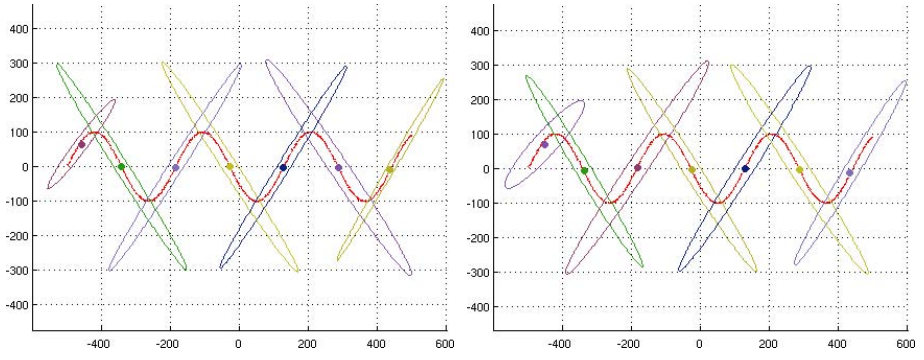
$$x(t) = t - 500$$
$$y(t) = 100\sin(t/50)$$

This training data are added with standard Gaussian noise. Table 3 shows the model parameters computed by EM and IGMM, and Fig. 3 shows these results graphically.

We can see from Fig. 3 that the distributions computed by IGMM are similar to those computed by EM, although IGMM produces larger variances. But considering that EM algorithm computes the distributions through several iterations over all dataset, whilst IGMM computes the model parameters incrementally with just one iteration, the obtained results are very good for an incremental approach.

Table 3. Model parameters computed for the sinusoidal dataset

	Model parameters estimated by EM						Model parameters estimated by IGMM					
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cl. 1	0.0799	-459.7	65.1	532.5	685.6	933.4	0.0934	-452.9	69.8	741.2	751.2	896.3
Cl. 2	0.1540	-342.4	-0.6	1971.8	-3092.5	4920.8	0.1389	-337.9	-7.2	1585.5	-2564.3	4242.4
Cl. 3	0.1562	-187.1	-1.8	2047.9	3164.0	4962.8	0.1617	-184.1	2.6	2407.3	3532.6	5323.6
Cl. 4	0.1613	-28.2	-0.5	2165.3	-3307.4	5135.3	0.1534	-25.0	-4.8	1949.2	-3015.6	4774.0
Cl. 5	0.1497	126.8	-1.8	1878.9	2970.8	4754.9	0.1566	128.5	0.5	2031.1	3115.6	4888.9
Cl. 6	0.1722	287.9	-2.9	2492.0	-3634.4	5431.7	0.1628	287.1	-2.1	2198.6	-3300.2	5084.3
Cl. 7	0.1264	437.3	-7.7	1328.1	2248.6	3834.4	0.1333	434.0	-12.3	1457.6	2371.2	3951.4



(a) EM model (b) IGMM model

Fig. 3. Concepts computed using the sinusoidal dataset

4 Conclusion

In this paper we presented IGMM, an on-line algorithm for modeling data flows derived derived from the classic Robbins-Monro stochastic approximation method [8]. It is rooted in the well established field of statistical learning, using an incremental Gaussian Mixture Model to represent the probability density of the input data flow, and adding new density components to the model whenever a new regularity, or concept, is identified in the incoming data. As one of the main contributions in the development of IGMM, we presented the update recursive equations that are approximate incremental counterparts of the update equations used by the EM algorithm.

The performed experiments have shown that after a single presentation of the training data IGMM generated a model with parameters very close to the ones computed by EM algorithm after 100 iterations over the same data. Moreover, for these data, IGMM needed an initial sequence of just 10 patterns of each concept of the environment to afterwards correctly distinguish the corresponding contexts. Future developments will integrate the proposed model in robotic learning tasks using a real robot Pioneer 3-DX. For this, IGMM will extract concepts from sensory-motor data flows, handling not only sonar signals but also the speed of robot's motors.

Acknowledgment

This work is supported by CNPq, an entity of the Brazilian government for scientific and technological development.

References

1. Arandjelovic, O., Cipolla, R.: Incremental learning of temporally-coherent gaussian mixture models. In: Proc. 16th British Machine Vision Conf. (BMVC), Oxford, UK, pp. 759–768 (2005)
2. Kristan, M., Skocaj, D., Leonardis, A.: Incremental learning with gaussian mixture models. In: Proc. Computer Vision Winter Workshop, Moravske Toplice, Slovenia, pp. 25–32 (2008)
3. Fisher, D.H.: Knowledge acquisition via incremental conceptual learning. Machine Learning 2, 139–172 (1987)
4. Gennari, J.H., Langley, P., Fisher, D.: Models of incremental concept formation. Artificial Intelligence 40, 11–61 (1989)
5. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. Univ. California Press, Berkeley (1967)
6. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Boston (2006)
7. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
8. Robbins, H., Monro, S.: A stochastic approximation method. Annals of Mathematical Statistics 22, 400–407 (1951)
9. Titterington, D.M.: Recursive parameter estimation using incomplete data. Journal of the Royal Statistical Society 46(2), 257–267 (1984)
10. Wang, S., Zhao, Y.: Almost sure convergence of titterington's recursive estimator for mixture models. Statistics & Probability Letters (76) (2001–2006)
11. Neal, R.M., Hinton, G.E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Learning in Graphical Models, pp. 355–368. Kluwer Academic Publishers, Dordrecht (1998)
12. Sato, M.A., Ishii, S.: On-line EM algorithm for the normalized gaussian network. Neural Computation 12(2), 407–432 (2000)
13. Cappé, O., Moulines, E.: Recursive EM algorithm with applications to DOA estimation. In: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Toulouse, France (2006)
14. Cappé, O., Moulines, E.: Online EM algorithm for latent data models. Journal of the Royal Statistical Society (2008)
15. Keehn, D.G.: A note on learning for gaussian proprieties. IEEE Trans. Information Theory 11, 126–132 (1965)
16. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, London (1990)

Bayesian Network Structure Inference with an Hierarchical Bayesian Model

Adriano Velasque Werhli

Centro de Ciências Computacionais - C3,
Universidade Federal do Rio Grande
FURG, Rio Grande, Brazil
werhli@gmail.com

Abstract. Bayesian Networks (BNs) are applied to a wide range of applications. In the past few years great interest is dedicated to the problem of inferring the structure of BNs solely from the data. In this work we explore a probabilistic method which enables the inclusion of extra knowledge in the inference of BNs. We briefly present the theory of BNs and introduce our probabilistic model. We also present the method of Markov Chain Monte Carlo (MCMC) which is used to sample network structures and hyper-parameters of our probabilistic model. Finally we present and discuss the results focusing on aspects related with the accuracy of the reconstructed networks and how the proposed method behaves when provided with sources of knowledge of different quality.

Keywords: Bayesian Networks, Bayesian Hierarchical Model, MCMC.

1 Introduction

Bayesian Networks (BNs) are widely used in a variety of problems and applications. Among the various aspects of BNs the fast and accurate learning of its structure from data is still very challenging. This challenge is particularly difficult when the data sets available for inference are sparse. In this paper we follow a score-based inference scheme where a score is assigned to a particular model (network structure) given some observed data. In order to improve the accuracy of the inferred networks we propose a probabilistic model which enables us to use extra knowledge in the inference. After finding the score the next step is to search for the model that better agrees with the data. In the scenario of sparse data, instead of searching for the best model, it makes more sense to search for a collection of best models and we pursue this with a Markov Chain Monte Carlo (MCMC) method. The paper is organized as follows: In Section 2 the methodology of BNs is briefly introduced and the method for addition of extra knowledge and the sampling scheme are presented. In Section 3 we present how the simulations were set and how the simulated data is obtained. Within this section we also discuss how the simulation results are evaluated. In Section 4 the results are presented and in Section 5 a discussion about the results and possible future directions of research are presented.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 92–101, 2010.

2 Methodology

2.1 Bayesian Networks (BNs)

Bayesian Networks (BNs) are a combination of probability theory and graph theory. Formally a BN is fully specified by a graphical structure $\mathcal{M}$, a family of conditional probability distributions $\mathcal{F}$ and their parameters $\mathbf{q}$, which together specify a joint distribution over a set of random variables of interest. The graphical structure $\mathcal{M}$ of a BN consists of a set of nodes and a set of directed edges. The nodes represent random variables, while the edges indicate conditional dependence relations. The family of conditional probability distributions $\mathcal{F}$ and their parameters $\mathbf{q}$ specify the functional form of the conditional probabilities associated with the edges, that is, they indicate the nature of the interactions between nodes and the intensity of these interactions. A BN is characterized by a simple and unique rule for expanding the joint probability in terms of simpler conditional probabilities. This follows the local Markov property: *A node is conditionally independent of its non descendants given its parents.* Due to this property, the structure $\mathcal{M}$, of a BN has necessarily to be a directed acyclic graph (DAG), that is, a network without any directed cycles. Let $X_1, X_2, ..., X_N$ be a set of random variables represented by the nodes $i \in \{1, ..., N\}$ in the graph, define $\pi_i[G]$ to be the parents of node X_i in graph G, and let $X_{\pi_i[G]}$ represent the set of random variables associated with $\pi_i[G]$. Then we can write the expansion for the joint probability as $P(X_1, ..., X_N) = \prod_{i=1}^{N} P(X_i|X_{\pi_i[G]})$.

2.2 Score-Based Approach Inference

The task of learning a BN structure in a score-based approach consists in devising a BN structure from a given set of training data $\mathcal{D}$. At the end we want to find a DAG structure that better explains the data we have for learning. If we define that $\mathbb{M}$ is the space of all models, the first goal is to find a model $\mathcal{M}^* \in \mathbb{M}$ that is most supported by the data $\mathcal{D}$, $\mathcal{M}^* = \text{argmax}_{\mathcal{M}} \{\mathcal{P}(\mathcal{M}|\mathcal{D})\}$. Having the best structure $\mathcal{M}^*$ and the data $\mathcal{D}$, we can now find the best parameters, $\mathbf{q} = \text{argmax}_{\mathbf{q}} \{P(\mathbf{q}|\mathcal{M}^*, \mathcal{D})\}$. If we apply Bayes' rule we get $P(\mathcal{M}|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{M})P(\mathcal{M})$ where the marginal likelihood implies an integration over the whole parameter space:

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\mathbf{q}, \mathcal{M})P(\mathbf{q}|\mathcal{M})d\mathbf{q} \tag{1}$$

The integral in Equation 1, our score, is analytically tractable when the data is complete and the prior $P(\mathbf{q}|\mathcal{M})$ and the likelihood $P(\mathcal{D}|\mathbf{q}, \mathcal{M})$ satisfies certain regularity conditions [1,2].

According to Equation 1 we have a way to assign a score to a graphical structure given a data set. However, the search for high scoring structures is not trivial. It is impossible to list the whole set of structures because its number increases super-exponentially with the number of nodes. Furthermore, when considering a sparse data set, $P(\mathcal{M}|\mathcal{D})$ is diffuse, meaning that it will not be

properly represented by a single structure $\mathcal{M}^*$. Hence, a Markov chain Monte Carlo (MCMC) scheme is adopted [3], which under fairly general regularity conditions is theoretically guaranteed to converge to the posterior distribution [4]. For more details about this scheme see [5].

2.3 Extra Knowledge

The main aim of this work is to investigate the influence of extra knowledge addition in the inference of BNs structure. First of all we need to define a function that measures the agreement between a given network structure $\mathcal{M}$ and the extra knowledge that we have at our disposal, $\mathcal{B}$. We call this agreement measure 'energy' following the approach proposed in [6].

A network structure $\mathcal{M}$ is represented by an adjacency matrix where each entry $\mathcal{M}_{ij}$ can be either 0 or 1 representing respectively the absence and the presence of an edge between node$_i$ and node$_j$. The prior knowledge matrix, or belief matrix, $\mathcal{B}$, is defined to have entries $\mathcal{B}_{ij} \in [0, 1]$ representing our knowledge about the node interactions. An entry $\mathcal{B}_{ij} = 0.5$ denotes that we do not have any information about the presence or absence of an edge between node$_i$ and node$_j$. If $0 \leq B_{ij} < 0.5$ we have prior evidence the edge between node$_i$ and node$_j$ is absent and the evidence is stronger as B_{ij} is closer to 0. At last, if $0.5 < B_{ij} \leq 1$ we have prior evidence that there is a directed edge pointing from node$_i$ to node$_j$. The evidence is stronger as B_{ij} is closer to 1. It is important to note that the entries in our belief matrix are not proper probabilities and they only express our belief, or knowledge obtained from other sources, about the relationships among nodes.

Having defined how to represent a BN structure, $\mathcal{M}$, and the extra belief, $\mathcal{B}$, the energy of a network is defined as:

$$E(\mathcal{M}) = \sum_{i,j=1}^{N} |\mathcal{B}_{i,j} - \mathcal{M}_{i,j}| \tag{2}$$

where N is the total number of nodes in the studied domain. From Equation 2 it is clear that the energy E is zero for a perfect match between the prior knowledge $\mathcal{B}$ and the actual network structure $\mathcal{M}$, while increasing values of E indicate an increasing mismatch between $\mathcal{B}$ and $\mathcal{M}$.

Following the work of [6] we integrate the prior knowledge expressed by Equation 2 into the inference procedure, and define the prior distribution over network structures, $\mathcal{M}$, to take the form of a Gibbs distribution:

$$P(\mathcal{M}|\beta) = \frac{e^{-\beta E(\mathcal{M})}}{\sum_{\mathcal{M}\in\mathbb{M}} e^{-\beta E(\mathcal{M})}} \tag{3}$$

where the energy $E(\mathcal{M})$ was defined in Equation 2, β is a hyper-parameter that corresponds to an inverse temperature in statistical physics, and the denominator is a normalizing constant that is usually referred to as the partition function. Note that the summation in the denominator extends over the set of all possible

network structures $\mathbb{M}$. The hyper-parameter β can be interpreted as a factor that indicates the strength of the influence of the prior knowledge relative to the data. For $\beta \to 0$, the prior distribution defined in Equation 3 becomes flat and uninformative about the network structure. Conversely, for $\beta \to \infty$, the prior distribution becomes sharply peaked at the network structure with the lowest energy.

For Dynamic Bayesian Networks the summation of the denominator of Equation 3 can be computed exactly and efficiently as discussed in [7] with

$$Z(\beta) = \prod_{n} \sum_{\pi_{\mathcal{M}}(n)} e^{-\beta \mathcal{E}(n, \pi_{\mathcal{M}}(n))}. \tag{4}$$

In this paper we apply our method only to static BNs and thus the summation of the denominator of Equation 3 is in fact an upper bound to the true value. This happens because this summation includes all possible structures and we are only interested in the DAG structures. Furthermore, throughout this paper we use a fan-in restriction of three as has been proposed in several other applications, for instance see [8,9,10]. This fan-in restriction makes the summation over all structures closer to the summation of only the DAGs as it reduces the number of densely connected networks. The partition function approximation has been investigated elsewhere [7,11] and was not found to pose a problem to the proposed method.

2.4 MCMC Sampling Scheme

With the prior probability distribution over network structures defined we now present an MCMC scheme to sample both the hyper-parameters and the network structures from the posterior distribution $P(\mathcal{M}, \beta|\mathcal{D})$.

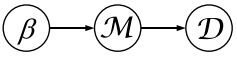


Fig. 1. Probabilistic Model. The probabilistic graphical model represents conditional independence relationships between the data $\mathcal{D}$, the network structure $\mathcal{M}$, and the hyper-parameter of the prior on $\mathcal{M}$. The conditional independence relationships can be obtained from the graphs according to the standard rules of factorization in Bayesian networks, as discussed, e.g., in [12]. This leads to the following expansion: $P(\mathcal{D}, \mathcal{M}, \beta) = P(\mathcal{D}|\mathcal{M})P(\mathcal{M}|\beta)P(\beta)$.

A new network structure $\mathcal{M}_{\text{new}}$ and a new hyper-parameter β_{new} are proposed respectively from the proposal distributions $Q(\mathcal{M}_{\text{new}}|\mathcal{M}_{\text{old}})$ and $R(\beta_{\text{new}}|\beta_{\text{old}})$. This proposed move is then accepted according to the Metropolis-Hastings update rule [4] with the following acceptance probability:

$$A = \min\left\{ \frac{P(\mathcal{D}|\mathcal{M}_{\text{new}})P(\mathcal{M}_{\text{new}}|\beta_{\text{new}})P(\beta_{\text{new}})Q(\mathcal{M}_{\text{old}}|\mathcal{M}_{\text{new}})R(\beta_{\text{old}}|\beta_{\text{new}})}{P(\mathcal{D}|\mathcal{M}_{\text{old}})P(\mathcal{M}_{\text{old}}|\beta_{\text{old}})P(\beta_{\text{old}})Q(\mathcal{M}_{\text{new}}|\mathcal{M}_{\text{old}})R(\beta_{\text{new}}|\beta_{\text{old}})}, 1 \right\} \tag{5}$$

which was expanded following the conditional independence relationships depicted in Figure 1.

In order to increase the acceptance probability which in turn can augment the mixing and convergence of the Markov chain the move is separated into two sub-moves. In the first move a new network structure $\mathcal{M}_{\text{new}}$ is sampled from the proposal distribution $Q(\mathcal{M}_{\text{new}}|\mathcal{M}_{\text{old}})$ while keeping the hyper-parameter β fixed. Next, we sample a new hyper-parameter β from the proposal distribution $R(\beta_{\text{new}}|\beta_{\text{old}})$ for a fixed network structure $\mathcal{M}$. The two sub-moves are iterated until a convergence criterion is satisfied. In this work the test of convergence consisted of repeating two MCMC simulations from independent initializations. Consistency in the marginal posterior probabilities of the edges was taken as indication of sufficient convergence. For more details about this procedure see Figure 2.

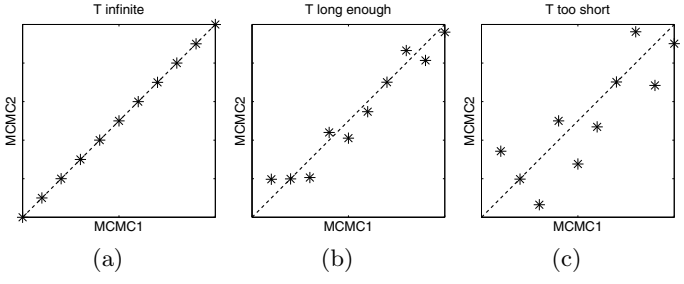


Fig. 2. MCMC simulations convergence test. The marginal posterior probabilities of the edges from two different simulation initializations are plotted. For an infinite time t all the posterior probabilities of the edges for both simulations are going to be the same as exemplified in panel (a). If the time t is long enough we should expect a graph as in panel (b). If the simulations have not properly converged yet the graph should appear as in panel (c). Note that a graph similar to the one shown in panel (a), only provides information that the two simulations have reached the same meta-stable pre-equilibrium, which is a necessary but not sufficient condition for MCMC convergence.

3 Simulations

In this section we explain how we set the simulations in order to explore the addition of prior knowledge in the inference of BNs. In all simulations we assumed that the data comes from a multivariate Gaussian distribution and adopted the BGe score derived in [13] .

3.1 Data Sets

To test the performance of the proposed method we use simulated data. Given a network structure the data sets are obtained with the simulation tool Netbuilder [14,15]. In Netbuilder a sigma-pi formalism is implemented as an approximation

to the solution of a set of Ordinary Differential Equations that model enzyme-substrate reactions. With this approximation simulation of data typical of signals measured in molecular biology is obtained. The data sets obtained with Netbuilder are closer to real data sets where non-linearities and noise are expected, hence, we use Netbuilder generated data to ensure a mismatch between the data sets and the inference model which assumes the data to come from a multivariate Gaussian distribution. We generated five data sets with 100 measurements in each. For more details about the data generation see [16].

The structure we generated data from is presented in Figure 3 taken from [17] where the authors have measured protein concentrations for each of the nodes. The referred network has 11 nodes and a total of 20 edges out of the possible 121. This network is widely studied and, hence, there is a good knowledge about its true structure. These characteristics make this specific network a good benchmark candidate for structure inference algorithms.

3.2 Extra Knowledge

One of the interesting aspects to be investigated is to observe how the proposed algorithm behaves when provided with prior information of different quality. The quality of prior information is the level of agreement between the prior and the data set available. In order to proceed with this investigation we propose three prior belief matrices each with a different agreement level with the true network of Figure 3.

The proposed prior belief, $\mathcal{B}_{100}$, completely agrees with the true structure and, conversely, the prior belief, $\mathcal{B}_0$ completely disagrees with the true structure. In between these two extremes we propose a prior belief, $\mathcal{B}_{50}$, that has half of its entries in agreement and the other half in disagreement with the true structure.

3.3 Evaluation Criteria

Not all of the edge directions in a Bayesian network can always be inferred. This is due to the existence of equivalence classes of networks [18] which may lead to partially directed graphs. In face of the presence of directed and partially directed graphs, to assess the performance of the proposed method we apply two

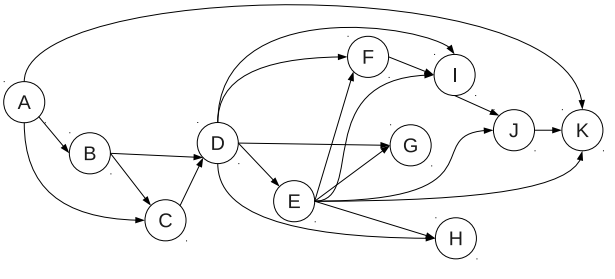


Fig. 3. Raf signalling pathway. The graph shows the currently accepted signalling network, taken from [17]. Nodes represent proteins, edges represent interactions, and arrows indicate the direction of signal transduction.

different criteria. In one of the approaches the information about the edge directions is completely discarded. Whenever two nodes are connected by a directed edge this edge is replaced by an undirected one. This approach is called the undirected graph evaluation (UGE). The other approach considers a predicted undirected edge as the superposition of two directed edges, pointing in opposite directions. This approach is called the directed graph evaluation (DGE). The result of the MCMC simulation is a sample of network structures which leads to a matrix of marginal posterior probabilities associated with the edges in a network. This defines a ranking of the edges. This ranking defines a receiver operator characteristics (ROC) curve, where the relative number of true-positive (TP) edges is plotted against the relative number of false-positive (FP) edges. Ideally we would compare the whole ROC curves but this is impractical. Therefore, we use the area under the ROC curve (AUC). The AUC is a measure of the area under the ROC curve and summarizes the results for all the thresholds. A perfect predictor would produce an AUC value of 1.00 conversely a random predictor would produce an AUC value around 0.50. In general bigger area values represent better predictors.

4 Results and Discussion

In this section we present the results we obtain when using the proposed methodology and how these compare with the standard MCMC where no prior knowledge is used. We ran each of the simulations twice in order to check the convergence as described in Figure 2. First of all we ran standard MCMC simulations where no prior is added. A summary of the results of these simulations is presented in Figure 4 as the horizontal lines. The middle horizontal line is the

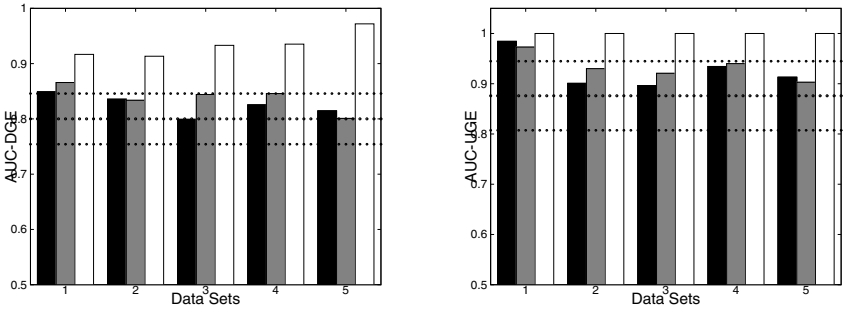


Fig. 4. AUC scores results. The DGE and the UGE evaluation criteria are shown respectively in the left panel and in the right panel. Each group of vertical bars represents one data set. The bar filled in black presents the obtained AUC values when considering the completely wrong prior, $\mathcal{B}_0$. The gray filled bar and the white bar presents respectively the AUC values obtained for the "half" correct prior, $\mathcal{B}_{50}$ and for the completely correct prior $\mathcal{B}_{100}$. The middle horizontal dashed line is the AUC average from the simulations without prior. The upper and lower horizontal represents one standard deviation from the mean.

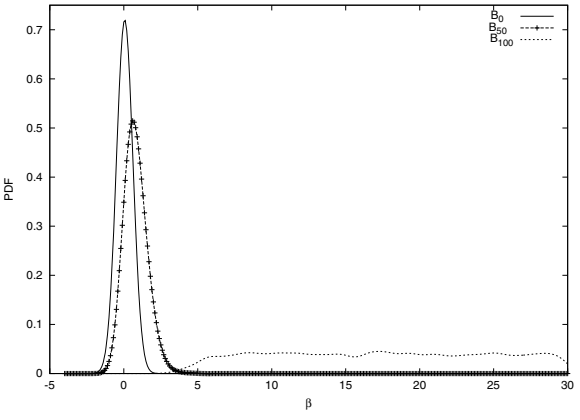


Fig. 5. Posterior distribution of hyper-parameter. For each of the prior knowledge available, $\mathcal{B}_0$, $\mathcal{B}_{50}$ and $\mathcal{B}_{100}$, the posterior distribution of the sampled hyper-parameters for all the data sets are estimated with a kernel estimator. As expected $\mathcal{B}_0$ has the smallest value indicating that this prior is not in agreement with the data and therefore it is practically not used. The prior $\mathcal{B}_{50}$ shows slightly higher values indicating that there is some agreement between the data and the prior. Finally the prior $\mathcal{B}_{100}$ is broadly distributed reaching up large values indicating a good agreement between the data and the prior.

average AUC over the five data sets and the upper and lower lines represent one standard deviation from the mean.

All the resulting AUC scores are presented in Figure 4. The results are separated in the left and the right panel which shows respectively the DGE and UGE evaluation criteria. Within each panel the vertical bars are divided in groups, one for each dataset. In each of these groups there are three bars. The black bar shows the AUC when considering a completely wrong prior, $\mathcal{B}_0$, the gray bar shows the results when a "half" correct prior is used, $\mathcal{B}_{50}$, and the white bar shows the results when a completely correct prior is used, $\mathcal{B}_{100}$.

In Figure 5 the resulting hyper-parameters for each of the three different levels of prior knowledge are shown. In order to summarize the results we present the posterior distribution of the hyper-parameters obtained with a kernel estimator. The posterior distribution was estimated considering the sampled hyper-parameters of the five data sets within each level of prior knowledge.

Observing the results of Figure 4 in conjunction with those of Figure 5 we can note interesting and crucial aspects of the proposed method. When prior knowledge available is not in agreement with the data the inference procedure learns a very low hyper-parameter which in turn practically turns off the use of prior knowledge and therefore the results are similar to the results of the standard MCMC. When the prior knowledge fully agrees with the data the hyper-parameter is sampled in a wide range of large values indicating that the information contained in the prior knowledge should be used. Confirming what is indicated by the hyper-parameter the results of the proposed method are higher

than the standard MCMC in this case. When the prior knowledge is in between the two extremes the sampled hyper-parameter is also in between the completely turned off prior and the completely turned on prior. In this case there is a small indication that the prior information should be used and the results are not consistently higher of those obtained with the standard MCMC.

5 Conclusion and Future Work

The proposed method behaves as expected but it may be improved for the cases where only part of the prior knowledge available is in agreement with the data. When the prior knowledge is in fully agreement with the data the sampled hyperparameter is large and the prior knowledge is used in the inference improving the accuracy of the reconstructed network. When the prior knowledge available is not in agreement with the data the hyperparameter is sampled at very small values practically turning off the inclusion of extra knowledge in the inference. This is a very important feature as it prevents the inference of being contaminated with bad information contained in the prior knowledge. Thus, even if the prior knowledge available is inadequate it will not harm the inference and the obtained result is the same obtained with the standard MCMC.

In practice it is very difficult to have a whole prior knowledge available which is completely correct or completely wrong, hence it is expected that we have something in between these two extremes. In our simulations we used the half correct prior, $\mathcal{B}_{50}$, to illustrate this situation. The results in terms of the network reconstruction for this case were not consistently higher than the ones obtained without any prior.

As future work we can explore some ways of splitting up the prior knowledge available. Ideally we could have one hyperparameter for each possible edge in the network. As this would produce prohibitive computational costs alternative ways of modularization of the prior should be explored. Another alternative is the study of the parallelization of the presented method.

References

1. Heckerman, D.: Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research, Redmond, Washington (July 1994)
2. Heckerman, D.: A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington (1995)
3. Madigan, D., York, J.: Bayesian graphical models for discrete data. International Statistical Review 63, 215–232 (1995)
4. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57, 97–109 (1970)
5. Husmeier, D., Dybowski, R., Roberts, S.: Probabilistic Modeling in Bioinformatics and Medical Informatics. In: Advanced Information and Knowledge Processing. Springer, New York (2005)

6. Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., Miyano, S.: Combining microarrays and biological knowledge for estimating gene networksvia Bayesian networks. In: Proceedings IEEE Computer Society Bioinformatics Conference (CSB 2003), pp. 104–113 (2003)
7. Werhli, A.V., Husmeier, D.: Reconstructing gene regulatory networks with Bayesian networks by combiningexpression data with multiple sources of prior knowledge. Statistical Applications in Genetics and Molecular Biology, Article 15 6(1) (May 2007)
8. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian networks to analyze expression data. Journal of Computational Biology 7(3/4), 601–620 (2000)
9. Friedman, N., Koller, D.: Being Bayesian about network structure. Machine Learning 50, 95–126 (2003)
10. Husmeier, D.: Sensitivity and specificity of inferring genetic regulatory interactionsfrom microarray experiments with dynamic Bayesian networks. Bioinformatics 19, 2271–2282 (2003)
11. Werhli, A.V., Husmeier, D.: Gene regulatory network reconstruction by Bayesian integration of prior knowledge and/or different experimental conditions. Journal of Bioinformatics and Computational Biology 6, 543–572 (2008)
12. Heckerman, D.: A tutorial on learning with Bayesian networks. In: Jordan, M.I. (ed.) Learning in Graphical Models. Adaptive Computation and Machine Learning, pp. 301–354. MIT Press, Cambridge (1999)
13. Geiger, D., Heckerman, D.: Learning Gaussian networks. In: de Mantaras, R.L., Poole, D. (eds.) Uncertainty in Artificial Intelligence, pp. 235–243. Morgan Kaufmann, San Francisco (July 1994)
14. Yuh, C.H., Bolouri, H., Davidson, E.H.: Genomic cis-regulatory logic: experimental and computational analysis ofa sea urchin gene. Science 279, 1896–1902 (1998)
15. Yuh, C.H., Bolouri, H., Davidson, E.H.: Cis-regulatory logic in the endo16 gene: switching from a specificationto a differentiation mode of control. Development 128, 617–629 (2001)
16. Werhli, A.V., Grzegorczyk, M., Husmeier, D.: Comparative evaluation of reverse engineering gene regulatory networks withrelevance networks, graphical Gaussian models and Bayesian networks. Bioinformatics 22(20), 2523–2531 (2006)
17. Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D., Nolan, G.P.: Causal proteinsignaling networks derived from multiparameter single-celldata. Science 308(5721), 523–529 (2005)
18. Pearl, J.: Causality: Models, Reasoning and Intelligent Systems. Cambridge University Press, London (2000)

On the Construction of Synthetic Characters with Personality and Emotion

Ary Fagundes Bressane Neto and Flávio Soares Corrêa da Silva

University of Sao Paulo, Laboratory of Interactivity and Digital Entertainment Technology, Rua do Matao, 1010, Sao Paulo, Brazil
{bressane,fcs}@ime.usp.br

Abstract. An important challenge in the development of interactive systems for digital entertainment is to build immersive virtual environments inhabited by synthetic characters that feature an illusion of life. Such characters should be adaptive to unforeseen situations, and therefore less predictable than deterministically scripted characters. In many cases, the adaptive behavior can be based on models of personality and emotion. In this paper we introduce an architecture for the construction of synthetic characters with personality and emotion based on the BDI model, to which is added an Affective Module. This module consists of three sub-modules, namely Personality, Mood and Emotion, and is used to characterize and influence the cognitive activities of perception, memory and decision making of characters. Our goal is to contribute to the design and implementation of believable synthetic characters for digital entertainment.

Keywords: emotional agents, affective computing, intelligent agents.

1 Introduction

Artificial Intelligence researchers have used systems for Digital Entertainment (e.g. computer games and 3D virtual worlds) as a platform for the development and benchmarking of novel techniques. These systems are particularly challenging due to their stringent requirements of scalability and real time behavior [16]. In turn, developers of systems for Digital Entertainment have resorted to Artificial Intelligence to create characters that feature a variety of convincing and consistent behaviors [13].

Recent studies in cognitive psychology and neuroscience analyze the role of personality and emotions in human cognition, based on the notions of perception, attention, planning, reasoning, creativity, learning, memory and decision making [20]. These notions can be characterized as modules in a formal model to describe the personality and emotions of autonomous agents [2], whose manifestations can be directly dependent upon personality and emotional states.

Many research projects have built agent models based on the notions of personality and emotion [1,11,7,9]. In the vast majority of these projects, the affective states influence the facial expression, voice intonation, gestures, selection

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 102–111, 2010.

of words and sentences of agents. However, few initiatives explore how personality and emotion influence cognitive processes such as perception, motivation, memory and decision making. According to Laird [12], future research in affective computing must explore how emotions interact with other modules in agent architectures (such as memory and decision making), as well as how emotions influence the interactions with other agents.

In this paper we introduce an architecture for the construction of synthetic characters with personality and emotions. This architecture is based on the BDI model, extended with an Affective Module composed by three sub-modules: Personality, Mood and Emotion. The Affective Module influences the cognitive activities of perception, memory and decision making of agents. In section 2 we detail the Affective Module, all it sub-modules and their relationships. In section 3 we present a proof of concept under development, whose empirical analysis shall be published in future work. Finally, in section 4 we present a short discussion about the architecture and future work.

2 The Architecture

Our architecture has been implemented using Jason [8], which is an open-source framework for the development of BDI (belief-desire-intention) intelligent agents based on an agent oriented declarative programming language called AgentSpeak. Jason is comprised by four main components: Belief Base, Plan Library, Events and Intentions. Using this framework, the developer can control the entire life cycles of agents (creation, execution and destruction) and the communication between them.

In order to develop an architecture that allows the construction of a synthetic character with personality and emotion, we have extended Jason to include the Affective Module and three filters, namely filters of perception, memory and plans. The Affective Module is used to influence the other modules and filters, turning the actions of the agents more believable.

Figure 1 shows a diagram of the architecture where rectangles represent the modules, solid arrows the data flow and dotted arrows the queries.

2.1 The Affective Module

The Affective Module influences other modules and filters to generate more diversified behavior for the agent. Internally, the module provides connections between personality, mood and emotion. According to [22] these three types of affection can be categorized by the influence that they have on agent behavior and by their temporal characteristics. The connections are implemented using the model of Mehrabian [15]. This model expresses the relationships between personality, mood and emotion using the parameters of the Big Five Personality Factors [14], the model Mood PAD [15] and the OCC model of emotion [19]. It has been chosen because it defines objective parameters for the data flow and because it has been used successfully in other studies [7,9].

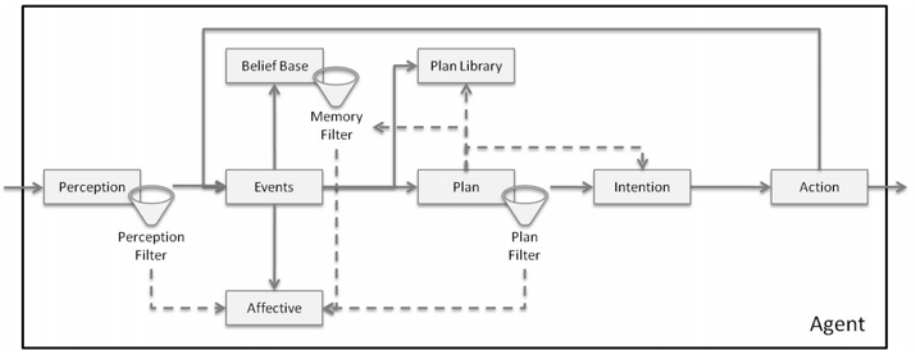


Fig. 1. The architecture and its components

A diagram of the module and the relationship among the sub-modules is shown in Figure 2. In this diagram rectangles represent the modules, solid arrows the data flow and dotted arrows the queries.

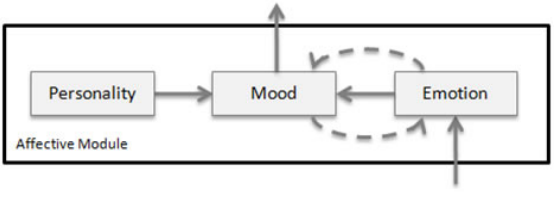


Fig. 2. The affective module

Personality

Personality is what individuates human beings. It is assumed to be stable, and influential on behavior as well as on the intensity of emotions. In our model we have adopted the model of The Big Five Personality Factors [14], whose focus is on description rather than on the understanding of personality. Through a lexical approach in natural language the model was developed, in such way that describes and classifies the human personality in five factors (Openness, Conscientiousness, Extroversion, Agreeableness and Neuroticism), where each factor groups a variety of psychological traits. The approach used to discover those factors presumes that everything that the human personality describes that has some relevance, interest or importance is registered in the natural language. The five factors are described below:

In the affective module, the personality is represented by the vector $P = [O, C, E, A, N]$ *where* $O, C, E, A, N \in [-1, 1]$.

Mood

Mood is a state that results from the cumulative effect of emotions and which strongly influences human cognitive functions [6]. Normally, mood is divided in

three states: good, bad and neutral. Similarly, in our work we use the model defined by Mehrabian [15], which divides mood in pleasure, arousal and dominance.

The model consists of three independent dimensions that are used to describe and to evaluate mood states: P (pleasure/displeasure), A (arousal/tranquility), D (dominance/submission). Pleasure (P) measure the affective quality of the emotional states, arousal (A) describes a composition of physical activity and attention, and dominance is defined in terms of control [15].

A point in the three dimensional space represents a person's mood. In numeric terms, the range of each dimension varies from -1.0 to 1.0, defining eight mood states:: [+P+A+D] Exuberant, [+P+A-D] Dependent, [+P-A+D] Relaxed, [+P-A-D] Docile, [-P-A-D] Bored, [-P-A+D] Disdainful, [-P+A-D] Anxious, [-P+A+D] Hostile.

Mood is represented as a vector in the affective module of the form $M = [m_1, m_2, m_3]$, where $m_i \in [-1, 1]$. The initial mood state of the agent is given by a relationship between the Big Five Personality Factors model and the PAD mood model, where personality is transformed into a point in the three dimensional mood space:

$$m_1 = 0,21 * P[E] + 0,59 * P[A] + 0,19 * P[N]$$
$$m_2 = 0,15 * P[O] + 0,30 * P[A] - 0,57 * P[N]$$
$$m_3 = 0,25 * P[O] + 0,17 * P[C] + 0,60 * P[E] - 0,32 * P[A]$$

Emotion

Emotion is a state of immediate effect and short duration. It is activated by events, actions or specific objects, and normally influences manifestations like facial expressions, gestures and voice intonation. The model of emotions that we have adopted in our experiments is the OCC model [19]. This model describes 22 types of emotions (Admiration, Anger, Disappointment, Distress, Fear, Fear-Confirmed, Gloating, Gratification, Gratitude, Happy-For, Hate, Hope, Joy, Love, Pity, Pride, Relief, Remorse, Reproach, Resentment, Satisfaction, Shame), that are evaluated as positive or negative, and that are result of three types of interaction: events that happen in the environment, objects of the scene and other agents. The model includes a rule-based system to generate each of the emotions. It has become the default model to synthesize emotions in machines [10].

In our architecture, emotions are represented by the vector $E = [e_1, e_2, ..., e_{22}]$, $e_n \in [0, 1]$.

The Relations among Personality, Mood and Emotion

Our architecture is similar to ALMA [7] and the Architecture of Interaction [9] which also use the model of Mehrabian [15] to express the relationship between the parameters of the model of The Big Five Personality Factors [14], of the mood model PAD [15] and the emotion model OCC [19]. Contrasting with those

models, however, our architecture focuses on cognitive responses, instead of the physiological ones.

Initially the personality sets the parameters that determine the mood base (M_{base}). After that, to each cycle of execution of the architecture, the emotion and mood values are updated (the value of personality remains stable along the time because it is considered a long-term affective state). To update the values, we use two functions described in [9], where E_s represents the emotional state, E_a the appraised emotional state, M_{cur} the current mood state and α the mapping (table 1) developed by [15] and extended by [7] that relates the 22 emotions of the OCC model with the three dimensions of the PAD model.

- **Emotion Update** $E_S = E_S + filter(E_a, M_{cur})$, where $filter(E_a, M_{cur}) = E_a + \frac{\sum_{i=1}^{22}\sum_{j=1}^{3}\alpha_{ij} * m_{ij}}{\sum_{i=1}^{22}\sum_{j=1}^{3}\alpha_{ij}}$.
- **Mood Update** $M_{cur} = M_{cur} + UpdateMood(E_s)$, where $UpdateMood(E_s) = \sum_{i=1}^{22}\sum_{j=1}^{3} e_i * \alpha_{ij}$.

2.2 Filters

The influence of the Affective Module in cognitive processes is implemented, in our architecture, as relations among these modules and three filters: perception, memory and plans.

Table 1. Mapping the OCC emotions into PAD dimensions [7]

Emotion	P	A	D
Admiration	0.5	0.3	-0.2
Anger	-0.51	0.59	0.25
Disappointment	-0.3	0.1	-0.4
Distress	-0.4	-0.2	-0.5
Fear	-0.64	0.6	-0.43
Fear-Confirmed	-0.5	-0.3	-0.7
Gloating	0.3	-0.3	-0.1
Gratification	0.6	0.5	0.4
Gratitude	0.4	0.2	-0.3
Happy-For	0.4	0.2	0.2
Hate	-0.6	0.6	0.3
Hope	0.2	0.2	-0.1
Joy	0.4	0.2	0.1
Love	0.3	0.1	0.2
Pity	-0.4	-0.2	-0.5
Pride	0.4	0.3	0.3
Relief	0.2	-0.3	0.4
Remorse	-0.3	0.1	-0.6
Reproach	-0.3	-0.1	0.4
Resentment	-0.2	-0.3	-0.2
Satisfaction	0.3	-0.2	0.4
Shame	-0.3	0.1	-0.6

Perception Filter

There is evidence that the current mood state of a person influences his/her perception. Empirical research has indicated that the intensity of perception of stimuli in individuals varies according to their affective state [18]. Similar observations have related affective state and sensitivity to external events [17].

We have extended Jason, so that "Perception" now receives two new attributes: a vector indicating which emotions are associated to this perception and a vector that determines the intensity of each emotion. We have also implemented a Perception Filter to select perceptions following two criteria: the perception associated to an intensive emotion and whether the result of the appraised emotions (the emotions are predominantly positive or negative) is consistent with the current affective state of the agent.

For the first case we have calculated a summation of the intensity of emotions and, if this value is greater than the value of the minimum intensity (I_{min}) defined by the user, the perception is selected, otherwise it is discarded. For the second case, assessment is made on the number of positive and negative emotions caused by perception. The dominant (positive or negative) type is compared to the current affective state of the agent (for all the filters, the current affective state of the agent is represented by the value of dimension P of mood). If the values are congruent, the perception is selected, otherwise it is discarded. Therefore, the Affective Module influences and increases the perception of the agent. The filter criteria are determined by the user in the source code.

Memory Filter

Recent studies have shown that storage and retrieval of information are strongly influenced by cognitive context [21]. We characterize the cognitive context in our architecture as a function of affective state.

According to Bower [4], there are four theories that match affective states and cognitive processes: (1) memories are more easily retrieved when an individual has the same mood as that of when they were stored; (2) the current affective state influences reasoning and interpretation of events; (3) memories with affective state similar to the current one are easily retrieved; and (4) memories associated to more intense emotions are more promptly retrieved.

In our architecture, the memories of an agent are characterized in its belief base. According to Bower [4], the affective states and their intensities can provide a context to influence the recovery of information. Hence a Memory Filter is built to select only information related to current emotional state of the agent from the belief base. Therefore, together with the information are also stored the emotions associated with it, as well as their intensity and affective state of the agent.

The information from the belief base can be retrieved through the Memory Filter according to the following criteria: the intensity of emotion associated to the information is greater than the minimum intensity defined by the user (I_{min}); the type (positive or negative) dominant of emotion associated with the stored

information is congruent with the current affective state; and the affective state at the time of memory storage is congruent with the current affective state. The parameters of the filter memory should be set in the source code of the agent.

Plan Filter

The influence of affective states in decision making is characterized in our architecture based on the hypothesis of Damasio's Somatic Markers [5]. This hypothesis indicates that before a human being makes a decision based on rational thought, the brain reduces the number of options based on somatic aspects of prior experiences. According to Damasio, to each decision is related a positive or negative marker, depending on the corporal evaluation made at the moment of the action. In our architecture, a marker is connected to each plan. The Plan Filter makes a pre selection of the possible action plans, rejecting those that have a negative marker. Hence, the somatic markers implement a sort of adaptive behavior, by influencing the choice of actions in order to increase the efficiency of the decision making process.

2.3 The Architecture Iteration Cycle

To create an agent using our architecture, it is necessary first to specify an initial set of beliefs, goals, plans, the character personality vector that, as described above, is represented by a vector of five positions which is assigned the values of openness, conscientiousness, extroversion, agreeableness, and neuroticism and all the parameters for the Perception Filter, Memory Filter and Plan Filter. After the initial specifications, the iteration cycle is started. As in standard Jason [3], the iteration cycle of the new architecture (shown in Figure 1) is divided into steps:

Step 1 - Perceiving the Environment: To each captured information from the external environment two characteristics are assigned: the emotions that they generate and with which intensity.
Step 2 - Filter the Perceptions: Information about the external environment passes through the Perception Filter where are selected depending on the parameters of emotion and intensity chosen by the user.
Step 3 - Generate Events: Perceptions coming from Perception Filter generate new events. Each of these events causes the update of the belief base and the affective state of the agent.
Step 4 - Update the Belief Base: After capture and filter the perception information, the belief base is updated. With each new (or updated) information contained in the belief base, they are assigned to three new information: emotion caused by that information, its intensity and affective state on the agent.
Step 5 - Updating the State Affective: When new information comes to the agent, they must have an impact on their emotional state depending on its intensity. Therefore, it is necessary to recalculate the values of mood and emotion of the agent. Besides the two update functions described above, is executed a function of decline. This function is also important to update, because if the

agent does not receive any more information from the perception, their emotional state should return to baseline. As suggested [9] the decline of the affective state is based on an exponential curve and has two characteristics: the intensity of emotion is declining faster than the mood and the intensity of the fall should consider the personality trait neuroticism (for characters with high neuroticism scores, positive emotions should disappear quickly and negative emotions more slowly and the reverse should happen for characters with low score in this trait).
Step 6 - Retrieve all relevant plans: After all the updates, the event is sent to the module "Plan" responsible for selecting the plan to be executed. In this module the plans related to the received event is retrieved from the Plan Library. The context of plans retrieved is checked with information stored in the Belief Base of the agent. The information retrieved from this base pass through the Memory Filter which, depending on the parameters set by the user, filters the information according to specific emotions related, current mood state or intensity of emotion.
Step 7 - Select a Plan: This step is responsible for selecting which plan should be executed. Therefore, initially plans selected in the previous step passes through the Plan Filter that remove all plans that have a negative somatic marker associated. After applied the filter, a query to the set of intentions is executed to determine which plan should be chosen.
Step 8 - Select an Intention for Future execution: The plan selected is inserted in the module Intentions. However this plan is one of the intentions that the agent has. In this step the agent chooses which of the intention to be executed.
Step 9 - Execute action: In the last step, the agent executes the plan (action) related to the intention selected. This action may cause a change in the environment where the agent is inserted or generate internal events that update the belief base, affective state and the somatic marker of the executed plan.

3 Proof of Concept

We have worked on two proofs of concept, for initial assessment of the extent to which our proposed model can contribute to the construction of more believable synthetic characters. The first experiment is a simulation of the problem of the "Prisoner's Dilemma." In this experiment, the problem is treated as a computer game where two players created with the proposed architecture interact in successive rounds. The second experiment is a computerized version of the "Memory Game" in which, differently from the first experiment, a human user participates in the game against an agent.

Our goal in the first experiment is to assess empirically the influence of personality, mood and emotion in the performance of artificial agents (aka synthetic characters, in the jargon of digital entertainment) in the course of solving a social puzzle. Preliminary results have indicated that the agents obtained based on the proposed architecture can provide the "look-and-feel" of human agents, as well as feature acceptable performance levels.

Our goal in the second experiment is more directed to the digital entertainment industry. We hope to be able to show that, by extending the cognitive abilities and behavior of agents with personality, mood and emotion, we can build truly believable agents, whose applicability can go beyond the existing possibilities for the design and implementation of computer games in which human players play against a computer. Preliminary results have indicated that indeed we can build more immersive computer-based opponents for humans.

4 Discussion and Future Work

With the advance of computer technology, users of digital entertainment systems are induced to believe that what they see is a faithful rendition of the real world. For this to happen, believable agents must be built to inhabit these systems, and further empirical analysis of their behavior must be developed to ensure that the constructed agents are indeed believable.

The extension of agent models to include affective states, and to account for how these states affect cognitive states, can be helpful for that.

In this work we have presented an architecture for the development of agents where the affective state influences cognitive activities of perception, memory and decision making. Preliminary tests indicate that the developed architecture can be used in various applications in the areas of games, virtual environment, interactive media and others.

This is a report of a work-in-progress. The proofs of concept developed so far are rather simple and preliminary. They are included in the text to exhibit the computational feasibility of the architecture and the possibility of obtaining believable outcomes from it, in the sense that the computational agent can be made to behave similarly to its human counterparts.

Acknowledgments. This research has been supported by FAPESP and Microsoft Research, grant 08/53977-3.

References

1. Bates, J., Loyall, B.B., Reilly, W.S.: An architecture for action, emotion, and social behavior. In: 4th European Workshop on Modelling Autonomous Agents in a Multi-agent World, Artificial Social Systems, pp. 55–68 (1992)
2. Bates, J.: The Role of Emotion in Believable Agents. Communications of the ACM, 122–131 (1994)
3. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley-Interscience, Chichester (2007)
4. Bower, G.H.: Mood and memory. American Psychology, 129–148 (1981)
5. Damasio, A.R.: Descartes' Error: Emotion, Reason, and the Human Brain. Harper Perennial, New York (1994)
6. Ekman, P., Davidson, R.J.: On emotion, mood, and related affective constructs. In: The Nature of Emotion. Oxford University Press, New York (1994)

7. Gebhard, P.: ALMA: a layered model of affect. In: Proceedings of The Fourth International Joint Conference on Autonomous Agents And Multiagent Systems, pp. 29–36 (2005)
8. Jason, http://jason.sourceforge.net
9. Kasap, Z., Moussa, M.B., Chaudhuri, P., Magnenat-Thalmann, N.: Making Them RememberEmotional Virtual Characters with Memory. IEEE Computer Graphics and Applications, 20–29 (2009)
10. Kasap, Z., Magnenat-Thalmann, N.: Intelligent virtual humans with autonomy and personality: State-of-the-art. Journal of Intelligent Decision Technologies, 3–15 (2007)
11. Kshirsagar, S., Magnenat-Thalmann, N.: A multilayer personality model. In: Proceedings of The 2nd International Symposium on Smart Graphics, pp. 107–115 (2002)
12. Laird, J.E.: Extending the Soar Cognitive Architecture. In: Artificial General Intelligence Conference, pp. 224–235 (2008)
13. Laird, J.E., Lent, M.V.: Behavior Modeling in Commercial Games. AI Magazine, 15–25 (2001)
14. Mccrae, R.R.: An Introduction to the five-Factor model and its applications. Journal of Personality, 175–215 (1992)
15. Mehrabian, A.: Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament. Current Psychology, 261–292 (1995)
16. Nareyek, A.: Intelligent Agents for Computer Games. Computers and Games, 414–422 (2001)
17. Niedenthal, P.M., Halberstadt, J.B., Margolin, J., Innes-Ker, A.H.: Emotional state and the detection of change in facial expression of emotion. European Journal of Social Psychology, 211–222 (2000)
18. Niedenthal, P.M., Setterlund, M.B.: Emotion congruence in perception. Personality and Social Psychology Bulletin, 401–411 (1994)
19. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotion. Cambridge University Press, Cambridge (1988)
20. Picard, R.: Affective Computing. MIT Press, Boston (1997)
21. Sternberg, R.J.: Cognitive Psychology. Wadsworth Publishing, Bemont (2002)
22. Vinayagamoorthy, V., Gillies, M., Steed, A., Tanguy, E., Pan, X., Loscos, C., Slater, M.: Building Expression into Virtual Characters. In: Eurographics 2006 (2006)

Towards Automated Trading Based on Fundamentalist and Technical Data

Carlos Henrique Dejavite Araújo and Paulo André Lima de Castro

Technological Institute of Aeronautics - ITA
São José dos Campos, SP, Brazil
Tel.: +55 12 39476942
carlosdejavite@gmail.com, pauloac@ita.br

Abstract. Autonomous trading is often seen as artificial intelligence applied to finance by AI researchers, but it may also be a way to motivate the development of autonomous agents, just like robot soccer competitions are used to motivate the research in mobile robots. In fact, some initiatives could be observed in recent years, for instance [1] and [2]. In this paper, we present a multiagent system composed by several autonomous analysts that use fundamentalist information in their reasoning process. These fundamentalist information are composed by company profit, dividends, data related to the company economic sector among others. This kind of information is rarely used on autonomous trading, because most of the agents deal only with technical information, which is composed by price and volume time series. Furthermore, we do not find a open source stock market simulator with support to fundamentalist trader agents. We then created a significantly extended version of the open source financial market simulation tool, called AgEx. This designed version provides also fundamentalist information about the trader's assets. As well as, makes more efficient the exchange of messages within AgEx. This efficiency allows traders that may submit orders in very short intervals of just some seconds or even some fraction of second, to use AgEx as a test platform. Using this new version of AgEx, we implemented and tested the multiagent system based on fundamentalist agents, that we call FAS. The achieved results are presented and analyzed.

Keywords: multiagent systems, automated trading, autonomous agents.

1 Introduction

Automated trading [3] may be an interesting environment for the development and test of multiagent systems and autonomous agents. In fact, we may observe the use of several AI techniques in automated trading, for instance: neural networks [4], reinforcement learning [5,6], BDI architectures [7], SWARM approaches [8]. Often, these initiatives relay on time series of price and/or trade volume. Despite the fact that these data are relevant, human experts many times use also economic information about the company (such as profit, dividends policy and so on), about the economic sector (size and growth projections) and general economy (growth projections, volatility analysis, etc.). The analyses based on this kind of data is commonly called fundamentalist, in opposition to the analyses based on price and volume time series, which is called technical analyses.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 112–121, 2010.

We present in this paper a multiagent system composed by autonomous traders that use fundamentalist information in their reasoning processes. These traders were implemented and tested. The results achieved are presented and analyzed. Furthermore, we present a significantly extended version of the open source financial market simulation tool, called AgEx. This extended version provides fundamentalist information about financial assets. The designed extensions also allow AgEx to be used for traders that may submit orders in very short intervals of just some seconds or even some fraction of second. We believe this tool may be useful for others researchers in automated trading and even for researchers that intend to test their systems in a complex and dynamic environment as financial market.

This paper is organized as follows section 2 describes the AgEx architecture highlighting the new features added to previous version [3]. In section 3, we present the FAS multiagent systems based on fundamentalist traders. The experiments and achieved results using FAS under AgEx are presented and discussed in section 4. Finally, we present some conclusions and suggestion to future work in section 5.

2 AgEx Architecture

Figure 1 presents the AgEx architecture [3] with its main components and communications links among trader agents and their human investors. The gray rectangles represent software agents (traders, manager and broker), while the circles represent the owners of the agents and a human administrator of AgEx. The entity represented by a white rectangle is a software module that is too simple to be classified as agent, and performs the actions determined by agents, such as buy and sell order executions. The component AgEx Data is just a database of real operations that took place in some real exchange and it may be used in simulations as described later. The AgEx system is composed by three components, which are described next:

– **Trader Agent:** It is responsible to decide and to submit buy or sell orders to some predefined assets. In fact, these agents use the AgEx as a simulation platform

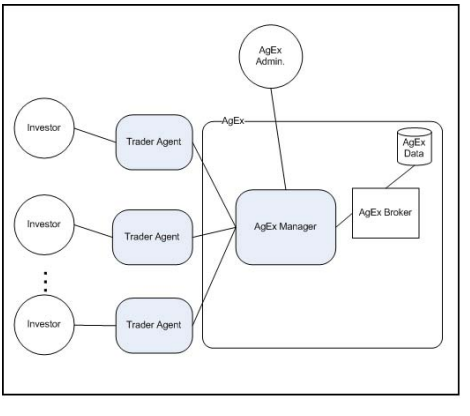


Fig. 1. AgEx architecture

framework for communication and life cycle management. Therefore, they are represented over AgEx border in figure 1. The AgEx system may provide services for many traders simultaneously, as shown in figure 1.

- **AgEx Manager:** This agent is responsible to validate and to process messages addressed to AgEx system. It sends the valid messages to execution that are performed by a software module, called AgEx Broker. The execution results are received by the manager and sent to the traders that submitted the order.
- **AgEx Broker:** It receives and executes buy or sell orders and informs the AgEx Manager about the result of execution.

The trader agents and the AgEx manager agent are synchronized by message exchanges. The manager defines the duration of each cycle (time step) and their transitions. All traders must be able to get the needed information, deliberate and submit orders within the interval of one time step. Whenever a trader does not complete these tasks within one time step, the system raises an overrun exception. One trader agent does not know in advance at which price one market order is executed, just like it happens in real markets. Furthermore, agents are not allowed to access price information beyond the current cycle. These features provide more realism to the simulation and avoid that one trader gets privileged information.

The AgEx database was changed to provide a structure with historic price and volume time series, financial data, financial indicators and macroeconomic data for the various stocks traded in the financial market. This data is transmitted to the agents through messages in Agent Communication Language (ACL) that uses an ontology specially defined for AgEx. This ontology is based in models developed by FIPA [9] and implemented by JADE platform.

3 Fundamentalist Agents System

In this section, it is presented the architecture of the multiagent system FAS (Fundamentalist Agents System). FAS works as one trader agent from the point of view of AgEx, because it manages one portfolio. However, it is composed by of one software module, **growth estimator**, and three kinds of agents, **price analyst**,**indexes analyst** and **manager**, which are briefly describe next.

- **Growth Estimator** is a software module that, through different estimation methods, provides to the *price analysts* and *indexes analysts* the estimative of companies' profit and EBIT growth to be used into the fundamentalist analyses executed by each one of those *analysts*.
- **Price Analyst** is an agent specialized in one of the various types of fundamentalist analysis based on models to estimate the stock fair price. This fair price after being determined is passed to the *manager* and gives support to the decision of buying or selling stocks.
- **Indexes Analyst** is an agent that uses strategies of fundamentalist indicators to determine if a stock is under or overvalued. Its advice is passed to the *manager* and gives support to the decision of buying or selling stocks.

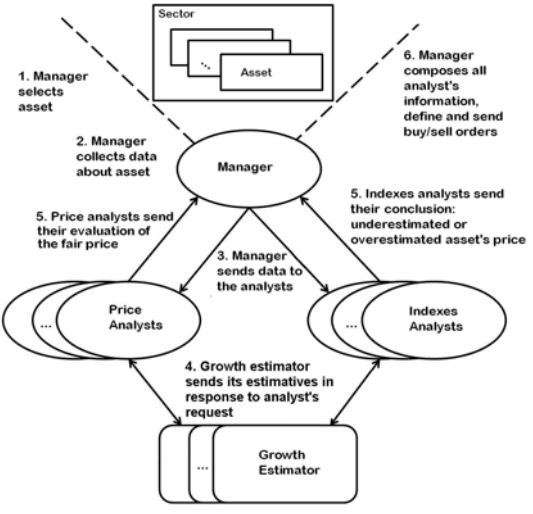


Fig. 2. FAS architecture

– **Manager** is an agent that selects a stock to be analyzed, passes data about this stock to the *indexes* and *price analysts*, gathers the analysis of each one of those analysts, determines the selected stock fair price and executes buy or sell orders according to this fair price. The fair price is a consolidation of the analysis made by each one of the indexes and price analysts balanced by the performance of each one of them in the previous periods.

The FAS architecture is presented in figure 2. FAS works on six sequential steps, as shown in figure 2, that are performed by its components. Each one of FAS components,**growth estimator**, **price analyst**,**indexes analyst** and **manager** are described in details in the following subsections.

3.1 Growth Estimator

The company growth rate estimation is really subjective and contains knowledge of professional analysts about the company and the economic sector which the company belongs to. Therefore, we used simplified versions of some common financial market estimation methods to estimate the perpetual company profit and EBIT growth. In order to achieve those estimates, the *growth estimator* receives as input of an analyst (Indexes or Price Analyst) financial data of the company (payout ratio, ROA, debt, equity, tax rate and interest rate) and macroeconomic data (GDP growth of the company's country). The *growth estimator*, that is assigned with one method of estimation, after receiving the inputs, executes the method of estimation and returns the estimative to the Analyst that sent the inputs. The growth estimator used by the analyst will be defined by what kind of model or strategy the analyst is using. If the model or strategy doesn't have any restrictions about what kind of estimated growth rate can be used (EBIT or

profit growth), the analyst will use all kinds of estimations to obtain from his model or strategy as many inputs as possible for the *manager*. The *growth estimator* is a software module that implements one of the four estimation methods following :

- Estimation by compound annualized growth rate of the last five years of GDP growth of the country that the company is held. This estimation is based on the hypothesis adopted by fundamental analyses that points out that the analysis only applies to companies with profit and EBIT growth similar to the economic growth of the country. The last five years are used to decrease the effects of a recent fall in GDP that would undervalue the potential company growth.
- Estimation by compound annualized growth rate of the last three years of GDP growth of the country that the company is held. This estimation is based on the hypothesis adopted by fundamental analyses that points out that the analysis only applies to companies with profit and EBIT growth similar to the economic growth of the country. The last three years are used to capture the effects of a recent boom in GDP that would leverage the potential growth of the company.
- Estimation of profit growth based on financial data according to these formula

$$\mathrm{EPS} = (1 - \mathrm{Payout})\{\mathrm{ROA} + \frac{D}{E}[\mathrm{ROA} - j(1 - i)]\}. \tag{1}$$

  This growth is based on the relation between financial indicators.
- Estimation of EBIT growth based on financial data according to this formula:

$$\mathrm{EBIT} = (1 - \mathrm{Payout})\mathrm{ROA}. \tag{2}$$

  These estimations were imposed to simplify the process of estimation of the company growth necessary to automate the fundamental analyses.

3.2 Price Analysts

The *price analysts* have as inputs company financial data (EBIT, debt, equity, dividend, outstanding stocks, depreciation, investments, interest rates, tax rates, working capital and beta), macroeconomic data (risk free rate and historic premium) and company profit or EBIT growth rate estimated by the *growth estimator*. The estimated growth rate used depends of the model restrictions about what kind of growth can be used (EBIT or profit growth) and if there is none, the *price analysts* use all to provide as many inputs as possible for the *manager*. The input data is passed by the *manager* according to the stock that the *manager* wants to buy or sell.

From the input data given by the *manager*, the *price analyst* calculates more elaborate financial indicators such as free cash flow, weight average cost of capital, cost of equity, and uses the Gordon model and the Free Cash Flow to Firm model [10] to estimate the fair price to be paid for the stock selected by the *manager*.

The fair price obtained is passed to the *manager* that weights it according to the performance showed by the price analyst in the previous periods.

3.3 Indexes Analysts

The *indexes analysts* have as input company financial data (profit, equity, outstanding stocks, revenue, payout, ROE, stock price and beta), macroeconomic data (risk free rate and historic premium) and company profit or EBIT growth rate estimated by the *growth estimator*. The estimated growth rate used depends of the fundamentalist indicator strategy restriction about what kind of growth can be used (EBIT or profit growth) and if there is none, the *indexes analysts* use all to provide as many inputs as possible for the *manager*. The input data is passed by the *manager* according to the stock that the Manager wants to buy or sell. The *indexes analysts* uses one of the fundamentalist indicator strategies described as following:

- The fundamentalist indicator Price/Profit (PL) has the following strategies: (1) Select stock that has the lowest market PL among the stocks of the same sector. This stock probably is undervalued, (2) Compare the market PL value with the expected PL value according to basic financial data using different profit growth rate estimations. If the market PL is lower, it is probable that the market is undervaluing the company, (3) Compare market dividend yield with risk free rate. If the market dividend yield is higher, it is probable that the stock is undervalued and(4) Compare market dividend yield with the expected dividend yield value according to basic financial data using different profit growth rate estimations. If the market dividend yield is lower, it is probable that the stock is undervalued.
- The fundamentalist indicator Price/Book Value (PBV) has the following strategies: (1) Select stock that has the lowest market PBV among the stocks of the same sector. This stock probably is undervalued, (2) Compare the market PBV value with the expected PBV value according to basic financial data using different EBIT growth rate estimations. If the market PBV is lower, it is probable that the market is undervaluing the company and (3) Compare market PBV value with company ROE. If PBV is the lowest and ROE is the highest among stocks of the same sector, it is probable that the market is undervaluing the price of the stock.
- The fundamentalist indicator Price/Sell (PS) has the following strategies: (1) Select stock that has the lowest market PS among the stocks of the same sector. This stock probably is undervalued, (2) Compare the market PS value with the expected PS value according to basic financial data using different profit growth rate estimations. If the market PS is lower, it is probable that the market is undervaluing the company and (3) Compare market PS value with company profit margin. If PS is the lowest and the profit margin is the highest among stocks of the same sector, it is probable that the market is undervaluing the price of the stock.

It is known that these strategies have different results depending on the year and which stock is being used and that the advices of under or overvalued is not a precise value and do not guarantee with 100% that the stock price rises or falls. So, it is possible to use fuzzy logic [11] to transform the fundamentalist indicator value, a crisp value, into a linguistic value of the linguistic variable "*Indexes analyst price*" in a way to serve

as entrance for a fuzzy inference system. A system that can be used to obtain conclusions about the fair price to be paid by a stock.

This linguistic variable "*Indexes analyst price*" is the advice returned to the Manager and indicates the pertinence level in which the stock chosen by the Manager belongs to the linguistic values "*Overestimated*", "*Underestimated*" and "*Neutral*". The last value, "*Neutral*", indicates that the fundamentalist indicator do not present enough evidences to be possible to affirm if the stock is under or overvalued. The linguistic variable "*Indexes analyst price*" is returned to the *manager* with the pertinence functions of each possible value and with the crisp value of input (fundamentalist indicator value). For each of the described fundamentalist indicator strategy, these pertinence functions are update dynamically at each new asset selection.

3.4 Manager

The *manager* of a sector is responsible for observe the companies of that sector, gather financial and macroeconomic data necessary to the analysts elaborate their advices and analyses, receive the fair price from the *price analysts* and the advice (linguistic variable) indicating the pertinence level of under and overestimated of the *indexes analyst* and aggregate everything to come to a unique fair price for the selected stock. With this fair price, is possible to buy and sell stocks of the companies in the sector of his responsibility. Therefore, the *manager* manages a portfolio of companies' stock of a sector and uses the analysts to improve his portfolio performance.

During this analysis aggregation process, it is important to consider how trustful those analyses had proven to be. In order to the final fair price to have a major weight of trustful analyses and a minor weight of those who are not. The confidence level of an analysis is measured by the gain that the analyst had with his analyses during the period of evaluation. This period of evaluation is a system parameter and allows to adequate the system to be more or less reactive to the extern environment. However, to determine if a analysis is trustful or not it is also a imprecise value and therefore needs to be made fuzzy to be used as an input in the fuzzy inference system and to come to a conclusion about the final fair price. The *analyst* gain is measured by the gain that the *manager* would have if the decisions of buying and selling the chosen stock were to be taken based only on the analyses about the stock of the evaluated analyst.

This way, if the analysis of the evaluated *analyst* indicates that the stock is "*Underestimated*", the *manager* would have bought the stocks, therefore the position is saved as +1 and the buy price P is also saved. If the analysis indicates that the stock is "*Overestimated*", the *manager* would have sold the stocks, therefore the position is saved as -1 and the sell price P is also saved. If the analysis indicates "Neutral", the positions and prices are kept. The position and price only change if the analysis changes from "*Underestimated*" to "*Overestimated*" or vice-versa. Therefore at the end of the evaluation period, price and positions saved are weight summed to obtain the gain/performance of the *analyst* in the period. Analyst's open positions are closed (positions sum must be zero) at the end of the evaluation period by doing an opposite trade with the market price in order to equalize all *analysts*. During the evaluation period, *analysts* are permitted to sell short in order to have an appropriate performance and fair comparison.

4 Simulated Experiments and Results

In this section, the results of FAS simulation are described and analyzed. In subsection 4.1, the parameters used in FAS simulation, details about the database and the used AgEx configurations are presented. In subsection 4.2, the results obtained by the simulations are shown and analyzed.

4.1 Simulation Characteristics

We performed simulations over a set of five different sectors, where four of those have five companies and one have three. The sectors and the companies chosen were among the biggest american companies of each of the biggest american economy sectors listed in Fortune 500-2008. These companies were selected because of the availability of financial data and to be giants in its sectors which ensures the hypothesis used by the fundamental analyses used in this work. The companies and sectors selected are listed below:

- Oil&Gas: Exxon, Chevron, ConocoPhillips, Valero Energy and Marathon Oil
- Metals: Alcoa Inc., United States Steel Corp, Nucor Corp, Commercial Metals and AK Steel Holding Corp
- Automobiles and Parts: Ford Motor, Johnson Controls and Goodyear Tire & Rubber
- Computer Software: Microsoft, Oracle, Symantec, CA Inc and ADOBE Systems
- Computers and Office Equipments: Hewlett-Packard, Dell, Apple, Xerox and Sun Microsystems

The financial data were obtained in Bloomberg from 2001 to 2008. This database had a bunch of incomplete or missing data. This data were estimated or completed before being inserted into the AgEx database for the simulation. The estimative realized for the six missing data (investments, beta, ROA, ROE, tax rate and interest rate) were :

1. The investment were estimated as: $\Delta\mathrm{GrossFixedAsset} - \Delta\mathrm{AccumlatedDepreciation}$, where Δ is the variation occurred from last year to the current year
2. The beta was calculated from linear regression of the last three years of monthly returns of company stocks against the market index S&P500. This estimative is used by YahooFinance that was used as a comparison base. It is important to keep in mind that beta was calculated for all years from 2001 to 2008 dated from the company financial statement launch.
3. ROA was estimated according to the formula: $\mathrm{EBIT}(1-i)/\mathrm{Assets}$, where EBIT is the earnings before interest and taxes, i is the tax rate and Assets are the total assets owned by the company
4. ROE was estimated according to the formula: $\mathrm{Profit}/\mathrm{BV}$, where BV is the book value of the company equity.
5. Tax Rate was estimated accordingto the formula: $\mathrm{TaxExpense}/\mathrm{PretaxIncome}$, where Pretax Income is the direct profit coming from the company's activities before the payment of taxes.
6. Interest Tax was estimated according to the formula: $\mathrm{InterestExpense}/\mathrm{Debt}$, where Debt is the company long-term debt.

The AgEx system was simulated in historic mode with prices and volume time series from 01/02/2001 to 10/30/2009 obtained in YahooFinance. The *manager* starts with zero stocks for each company in the sector and with one million in cash. This money is equally distributed between the companies of the sector and is exclusive for trades in each company stock.

4.2 Simulation Results and Analyzis

The table 1 shows the consolidate FAS portfolio and compare it with different sector and financial market benchmarks. The results of each *fundamentalist analyst* are compared among themselves for the eight years of simulation in order to validate the implementation and performance of the analysis. Furthermore, those analyses are compared with the *manager* performance in order to validate if the *manager* really benefits from the analyses composition.

Table 1. Accumulated return, risk and Sharpe ratio achieved by FAS and sector portfolios

	2002	2003	2004	2005	2006	2007	2008	2009	Risk	Sharpe ratio
FAS	3%	17%	41%	48%	61%	76%	55%	63%	24,6%	3,0
Metals	-16%	19%	124%	130%	218%	346%	70%	89%	115,4%	0,9
Oil&Gas	-17%	12%	89%	131%	166%	182%	123%	128%	70,6%	2,0
Software	-20%	6%	14%	16%	28%	29%	8%	22%	15,6%	2,1
Automobile e Parts	-37%	-8%	-4%	-12%	25%	26%	-14%	-2%	20,5%	0,4
Comp. and Office Equip.	-29%	5%	66%	89%	126%	183%	88%	171%	74,1%	2,4
SP500	-22%	-2%	4%	12%	24%	20%	-24%	-11%	18,3%	0,0
Total Sector	-24%	7%	58%	71%	113%	153%	55%	82%	55,7%	1,7
Risk Free Asset	5%	10%	14%	19%	24%	30%	36%	41%	-	-

We also compare the *manager* performance with the performances of the stocks in the sector and benchmark market index such as S&P500 and the sector portfolio, compound of equal portion of the stocks in the sector, in order to validate the *manager* performance against the financial market. These performances are compared using a well known index from finance theory, the Sharpe ratio [12], where the return of the risk free asset is the return of a benchmark (in the work S&P500 was used as benchmark).

The achieved results show that FAS has the best (highest) Sharpe ratio when compared to sector portfolios, table 1. However, FAS is just the fifth in accumulated return from 2002 to 2009, nevertheless it got 63% in that period. As explained in section 4.2, Sharpe ratio is defined by the return above a benchmark return divided by the risk, measured as standard deviation of returns. Therefore, the best position in Sharpe ratio can be explained, due to the good performance in risk . FAS presents the second best (lowest) risk, see table 1.

5 Conclusions and Future Work

The AgEx tool presented in this paper is a special-purpose software agent platform for simulation of financial markets, with support to traders that rely on fundamentalist

analyses. It is open source and allows market simulation with prices from real markets. It makes available a market ontology that simplifies communication. AgEx provides facilities to launch traders from several computers over the net and to analyze their performances.

We presented a multiagent trader system, FAS, developed with AgEx and based on fundamentalist analysis. FAS has shown to be a promising system for the automate asset portfolio administration using fundamentalist analysis. The system showed that can maximize the index Sharpe with an asset portfolio balanced between risk assets, stocks, and risk free assets, treasury bonds. Besides, it showed consistency in its long-term returns due to a lower exposure of market oscillations. Finally, we believe that AgEx new version can be very useful for others researchers trying to develop new trading strategies based on technical or fundamentalist analyses.

References

1. Kearns, M., Ortiz, L.: The penn-lehman automated trading project. IEEE Intelligent System 18(6), 22–31 (2003)
2. Azevedo, S.C., Teixeira, M., Costa, A.D., Borsato, B., Soares, F.A., Lucena, C.J.P.: Multiagent system for stock exchange simulation - masses. In: Proceedings of the Fourth Workshop on Software Engineering for Agent-oriented Systems - SEAS, Campinas, Brasil (2008)
3. Castro, P.A., Sichman, J.S.: Agex: A financial market simulation tool for software agents. In: Proceedings of 11th International Conference on Enterprise Information Systems, ICEIS., Milan, Italy, pp. 704–715 (2009)
4. Kendall, G., Su, Y.: A multi-agent based simulated stock market - testing on different types of stocks (2003)
5. Sherstov, A., Stone, P.: Three automated stock-trading agents: A comparative study. In: Proceedings of the Agent Mediated Electronic Commerce (AMEC) Workshop - AAMAS 2004, New York (2004)
6. Feng, Y., Nevmyvaka, Y., Kearns, M.: Reinforcement learning for optimized trade execution. In: Proceedings of the 23rd International Conference on Machine Learning- ICML 2006, Pittsburgh, Pennsylvania (June 2006)
7. Feng, X., Jo, C.-H.: Agent-based stock trading. In: Proceedings of the ISCA CATA-2003, Honolulu, Hawaii(March 2003)
8. Kendall, G., Su, Y.: A particle swarm optimisation approach in the construction of optimal risky portfolios. In: Proceedings of the 23rd IASTED International Multi-Conference Artificial Intelligence and Applications, Innsbruck, Austria, pp. 140–145 (February 2005)
9. FIPA: Fipa. the foundation for intelligent physical agents (2009)
10. Damodaran, A.: Avaliação de investimento: ferramentas e técnicas para a determinação do valor de qualquer ativo. Qualitymark Ed., Rio de Janeiro (1997)
11. Aguiar e Oliveira Jr., H.: Inteligência Computacional aplicada à administração, economia e engenharia. Thomson, São Paulo (2007)
12. Sharpe, W.F.: The sharpe ratio. Journal of Portfolio Management 13(3), 227–286 (1994)

Developing a Consciousness-Based Mind for an Artificial Creature

Ricardo Capitanio Martins da Silva and Ricardo Ribeiro Gudwin

DCA-FEEC-UNICAMP
Av. Albert Einstein 400, 13.083-852 Campinas, SP
{martins,gudwin}@dca.fee.unicamp.br

Abstract. This work describes the application of the Baars-Franklin Architecture (BFA), an artificial consciousness approach, to synthesize a mind (a control system) for an artificial creature. Firstly we introduce the theoretical foundations of this approach for the development of a conscious agent. Then we explain the architecture of our agent and at the end we discuss the results and first impressions of this approach.

1 Introduction

The scientific study of consciousness has improved dramatically in the last ten years [1]. A technological offspring of these studies is the field of artificial consciousness [2,3,4]. In this work we concentrate in what we call here the Baars-Franklin architecture (BFA). The BFA is a computational architecture being developed by the group of Stan Franklin, at the University of Memphis [5,2,6], based on the model of consciousness given by Bernard Baars, called Global Workspace Theory [7].

The BFA has already been applied to many different kinds of software agents. The first application of BFA was CMattie [5,2], an agent developed by the Cognitive Computing Research Group (CCRG) at the University of Memphis, whose main activities were to gather seminar information via email from humans, compose an announcement of the next week's seminars, and mail it to members of a mailing list. Through the interaction with human seminar organizers, CMattie could realize that there was missing information and ask it via email.

The overall BFA received major improvements with subsequent developments. One remarkable implementation of it was IDA (Intelligent Distribution Agent) [6], an application developed for the US Navy to automate an entire set of tasks of human personnel agent who assigns sailors to new tours of duty. IDA is supposed to communicate with sailors via email and, in natural language, understand the content and produce life-like messages.

The BFA was also used outside of Franklin's group. Daniel Dubois from University of Quebec developed CTS (Conscious Tutoring System) [8], a BFA-based autonomous agent to support the training on the manipulation of the International Space Station robotic control system called Canadarm2.

Nevertheless, up to our knowledge, BFA was never used to implement a mind (a control system) for an artificial virtual creature. Artificial Creatures are a

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 122–132, 2010.

special kind of agents, embodied autonomous agents which exists in a certain environment, moving itself in this environment and acting on it [9]. Artificial creatures may be real or virtual. Examples of real artificial creatures are robots acting in the real environment. Virtual Artificial Creatures are software agents living in a virtual world, where they are able to sense and actuate by means of an avatar (a virtual body). One example of a virtual artificial creature is an intelligent opponent in a computer game, where an intelligent control system must decide the actions to be performed by the agent in order to foster a good entertainment to the system user, simulating with realism the behavior of a human opponent. Other examples of virtual artificial creatures include ethological simulation studies, in artificial life, where tasks such as foraging and sheltering are very common.

Virtual artificial creatures pose some interesting research problems when compared to other kinds of software agents where BFA has already been tested. In the original applications where BFA was tested, the perception system is based on the exchange of e-mail messages (the case of CMattie and IDA), and interactions in a HCI (human-computer interface), in the case of CTS. In a virtual artificial creature, perception must rely on remote (e.g. visual, sonar, etc) and/or local (e.g. contact) sensors, capturing properties of the scenario and interpreting them in order to create a world model. The behavior generation module is also different, as the agent must act on itself (its body) and over things on the environment. The main motivation for the research reported in this work is though to investigate how the use of BFA may impact the control of a virtual artificial creature, and what are the benefits which can be expected.

In the next section, we introduce briefly Baars' theory of consciousness, Global Workspace Theory, and then we describe how we customized BFA in order to deal with virtual artificial agents. After that, we introduce CAV (Conscious Autonomous Vehicle), the artificial creature we used in our study and its environment, and a brief analysis of the results of our simulations using CAV.

2 Global Workspace Theory and BFA

Bernard Baars has developed the Global Workspace Theory (GWT) [7] inspired by psychology and based on empirical tests from cognitive and neural sciences. GWT is an unifying theory that puts together many previous hypothesis about the human mind and human consciousness.

Baars postulates that processes such as attention, action selection, automation, learning, meta-cognition, emotion, and most cognitive operations are carried out by a multitude of globally distributed unconscious specialized processors. Each processor is autonomous, efficient, and works in parallel and high speed. Nevertheless, in order to do its processing, each processor may need a set of resources (mostly information of a specific kind), and at the same time, will generate another set of resources after its processing. Specialized processors can cooperate to each other forming coalitions. This cooperation is by means of supplying to each other, the kinds of resources necessary for their processing. They exchange resources by writing in and reading from specific places in

working memory. Coalitions may form large complex networks, where processors are able to exchange information to each other. But processors within a coalition do have only local information. There may be situations, where the required information is not available within the coalition. To deal with these situations, and allow global communication among all the processors, there is a global workspace, where processors are able to broadcast their requirements to all other processors. Likewise, there may be situations where some processor would like to advertise the resource it generates, as there may be other processors interested in them. They will also be interested in accessing the global workspace and broadcasting to all other processors. In the broadcast dynamics, only one coalition is allowed to be within the global workspace in a given instance of time. In order to decide which coalition will go to the global workspace in a given instant of time, a whole competition process is triggered. Each processor has an activation level, which expresses its urgency in getting some information or the importance of the information it generates. A coalition will also have an activation level which is the average of activation levels of its participants. At each time instant, the coalition with the highest activation level will win the access to the global workspace. Once a coalition is within the global workspace, all its processors will broadcast their requests and the information they generate. The broadcast mechanism do allow the formation of new coalitions, and also some change in working coalitions.

For Baars, consciousness is related to the working of this global workspace. Processors are usually unconscious, having access only to local information, but in some cases they may require or provide global information, in which case they request access to consciousness, where they will be able to broadcast to all other processors. This is the case when they have unusual, urgent, or particularly relevant information or demands. This mechanism supports integration among many independent functions of the brain and unconscious collections of knowledge. In this way, consciousness plays an integrative and mobilizing role. Moreover, consciousness can be useful too when automatized (unconscious) tasks are not being able to deal with some particular situation (e.g. they are not working as expected), and so a special problem solving is required. Executive coalitions, specialized in problem solving will be recruited then in order to deal with these special situations, delegating trivial problems to other unconscious coalitions. In this way, consciousness works like a filter, receiving only emergencial or specially relevant information.

Inspired by Baars description of his theory of consciousness, and also by previous work in the computer science literature, Franklin proposed a framework for a software agent which realized Baars theory of consciousness, in terms of a computational architecture, constituting so what we are calling here the Baars-Franklin architecture. In specifying BFA, Franklin used the following theories as background, among others not detailed here: Selfridge's Pandemonium [10] and Jackson's extension to it [11], Hofstadter and Mitchell Copycat [12] and Maes' Behavior Network [13].

From Hofstadter's Copycat, Franklin borrowed the notion of a "Codelet" (and also the Slipnet, for perception). He noticed that these *codelets* were more or less the same thing as Selfridge's "demons" in Pandemonium theory and also a good computational version for Baars *processors*. Jackson's description of an arena of *demons* competing for selection will fit as well Baars description of processors competing in a *Playing Field* for access to consciousness. Using these similarities, Franklin set up the basis of BFA: cognitive functions are performed by coalitions of codelets working together unconsciously, reading and writing tagged information to a Working Memory. Each codelet has an activity level and a tagged information. A special mechanism, the *Coalition Manager* will manage coalitions and calculate the activity level of each coalition. Another special mechanism, the *Spotlight Controller*, will be evaluating each coalition activity level, and defining the winning coalition. Also, the *Spotlight Controller* will be responsible for performing the broadcast of the tagged information of each codelet in the winning coalition, to all codelets in the system. The agent behavior is decided using a Behavior Network, whose propositions are related to the tagged information in the Working Memory.

Unfortunately, a full description of BFA is beyond the space available in this text. We refer the interested reader to [2,6,8,14], where a more detailed description of BFA is available.

3 Our Implementation of BFA

In our experiment, we developed an artificial mind (a control system), which we call CAV - *Conscious Autonomous Vehicle*, to control an artificial creature in a virtual environment (see figure 1). The creature and its environment were originally presented in [15] (where more details on its characteristics can be obtained) and were adapted for our current studies. In this environment, the creature is equipped with sensors and actuators, which enable it to navigate through an environment full of objects with different characteristics. An object can vary in its "color" and each color is linked to: a measure of "hardness" which is used in the dynamic model as a friction coefficient that can slow down the creature's movement (or completely block it), a "taste" which can be bad or good, and a feature related with "energy" which indicates that the object drains/supplies energy from/to the creature's internal rechargeable battery.

The creature connects to its mind through sockets. In this sense, the artificial mind is a completely separate process, which can be run even in a different machine. So, different minds can be attached to the creature and tested for the exact same situation.

When the simulation is started, the creature builds an incremental map of the environment based on the sensory information. Our agent adds landmarks to this map and uses them to generate movement plans. It has two main motivations: it should navigate from an initial point up to a target point, avoiding collisions with objects; and it should keep its energetic balance, taking care of the energy level in the internal batteries.

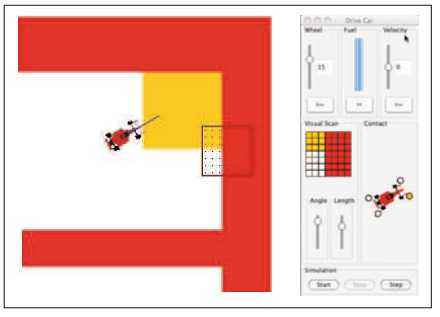


Fig. 1. Sensory-motor structure of the creature

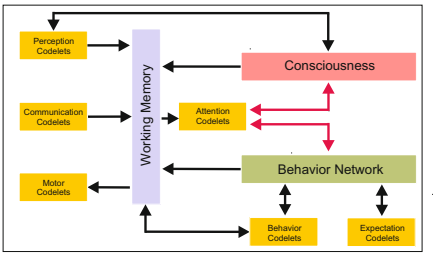


Fig. 2. *CAV*'s Architecture

Our architecture (see figure 2) is essentially rooted in the BFA implementation as in [2] (consciousness) and [6] (behavior network). *CAV* brings some modifications in the implementation related with the application domain, and the interaction among consciousness and behavior network. The following sections contain a brief description of CAV's modules.

3.1 Codelets

CAV is heavily dependent on small pieces of code running as separate threads called codelets (BFA borrows this name from Hofstadter's Copycat). Those codelets correspond pretty well to the specialized processors of global workspace theory or demons of Jackson and Selfridge.

BFA prescribes different kinds of codelets such as attention codelets, information codelets, perceptual codelets and behavior codelets. In addition to that, it is possible to create new types of codelets depending on the problem domain. CAV's domain does not require string processing as do most other BFA applications. Instead of that, the creature state is well divided in registers at the working memory. It is possible to have access to all variables anytime. Because of this, CAV does not use information codelets which in BFA are used to represent and transfer information. We have two kinds of behavioral codelets: the behavior codelets, linked with the nodes of the Behavior Network and responsible for "what to do", and motor codelets, which know "how to act" on the environment. With this in mind CAV has the taxonomy of codelets presented at Table 1.

3.2 Working Memory

The working memory consists of a set of registers which are responsible for keeping temporary information. The major part of the working memory is related to the creature status. The communication codelet constantly overwrites the registers like speed, wheel degree, sensory information and creature position. CAV's working memory works also as an interface among modules, for example,

Table 1. *CAV's Codelets Taxonomy*

Type	Role
Communication	Perform the communication with the simulator, bringing novel simulation information
Perception	Give an interpretation to what the agent senses from its environment
Attention	Monitor the working memory for relevant situations and bias information selection
Expectation	Check that expected results do happen
Behavior	Alter the parameter of the motor codelet
Motor	Act on the environment

between consciousness and the behavior network. Some codelets, including attention codelets watch what is written in the working memory in order to find relevant, insistent or urgent situations. When they find something, they react in order to compete for consciousness. Whenever one of then reaches consciousness, its information will influence the agent's actions.

3.3 Consciousness Mechanism

The consciousness mechanism consists of a *Coalition Manager*, a *Spotlight Controller*, a *Broadcast Manager* and attention codelets which are responsible for bringing appropriate contents to "consciousness" [2]. In most of the cases, codelets are observing the working memory, looking for some relevant external situation (e.g. a low level of energy). But some codelets keep a watchful eye on the state of the behavior network for some particular occurrence, like having no plan to reach a target. More than one attention codelet can be excited due to a certain situation, causing a competition for the spotlight of consciousness. If a codelet is the winner of this competition, its content is then broadcast to the registered codelets in the broadcast manager. We have three main differences between standard BFA and CAV, related to this module. The first one is that we don't use information codelets. The second is that not all of the codelets are notified like in BFA, just the registered ones. Finally, some codelets can be active outside of the playing field. In this case their contents will never reach consciousness.

3.4 Behavior Network

CAV's behavior network is based on a version of Maes' architecture [13] modified by Negatu [6]. Negatu adapted Maes' behavior network so each behavior is performed by a collection of codelets. Negatu's implementation also divided the behavior network in *streams* of behavior nodes.

The behavior network works like a long-term procedural memory, a decision structure and a planning mechanism. It coordinates the behavior actions through an "unconscious" decision-making process. Even so it relies on conscious broadcasts to keep up-to-date about the current situation. This is called "consciously mediated action selection" [6].

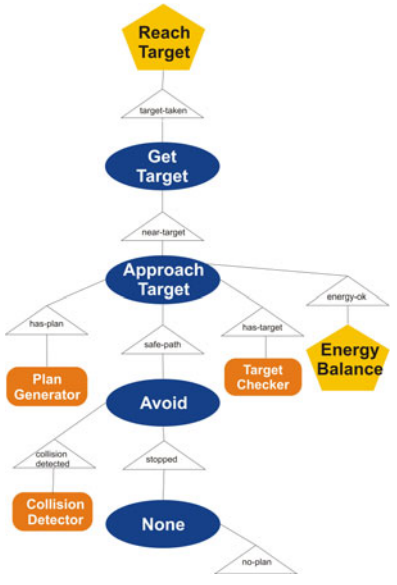


Fig. 3. Behavior Network - Target Stream

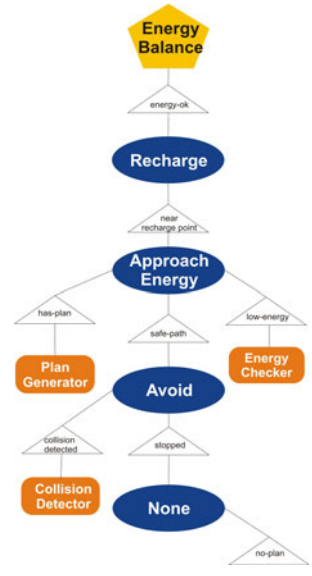


Fig. 4. Behavior Network - Energy Stream

CAV uses two main behavioral streams, the *Target* stream and the *Energy* stream, as in figures 3 and 4.

4 A Brief Analysis of CAV's Implementation

A running simulation of CAV's performance is illustrated in figure 5. The main experiment worked as expected. The creature was able to pursue its main objectives: to avoid collision with obstacles while exploring the environment, and at the same time maintaining an energy balance. While exploring the environment, if the energy level decreased to a critic limit, CAV correctly postponed its exploratory behavior, looked for the closest source of energy and traced a route to it to feed itself. After refreshing its batteries, it returned to its exploratory behavior. As we said before, though, our main goal was not simply related to the achievement of these tasks (something which could be achieved by more traditional methods, as e.g. in [15]), but understanding how "consciousness" could be used in such an application.

By applying BFA to this application, we would like to evaluate the value of "consciousness" (as in BFA) to the construction of a new generation of cognitive architectures to control artificial creatures. Pragmatically, we would like to understand what exactly it is this "consciousness" technology, and what the benefits to expect while applying it as a mind to an artificial creature. This goal was also achieved while we had the experience of studying BFA and applying it to the current application. Our findings are summarized in the next subsections.

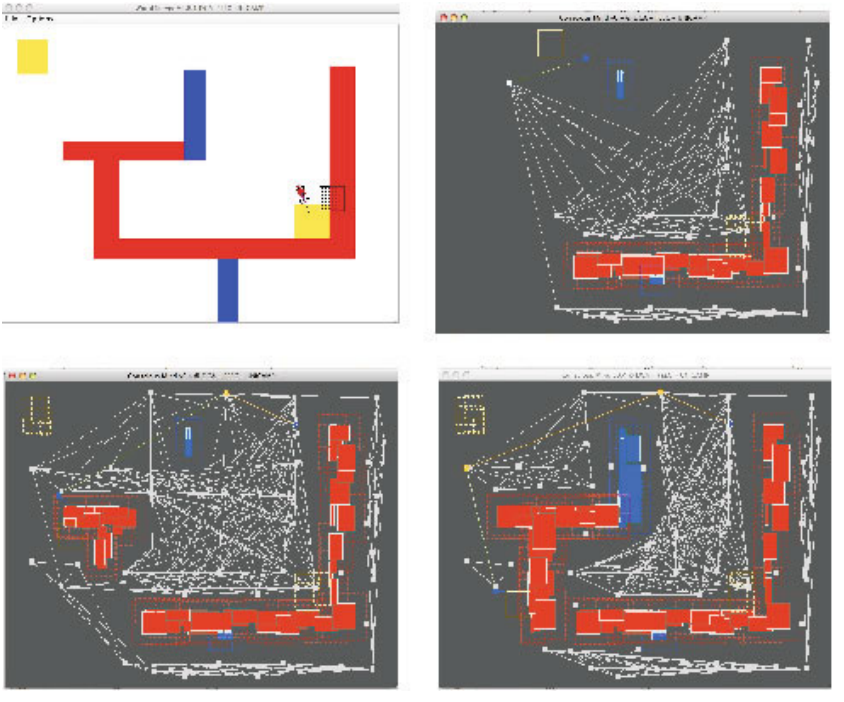


Fig. 5. Example of Simulation

4.1 A Qualitative Analysis

Our implementation of BFA as a mind of our artificial creature allowed us to better understand what is the role of consciousness in BFA and what are its main benefits as a technology. First let us make it very clear what is consciousness (in the context of BFA). The philosopher Daniel Dennet has already stated that: "Human consciousness (...) can best be understood as the operation of a '*Von Neumannesque*' virtual machine *implemented* in the parallel architecture of a brain". This is what BFA provides. It implements a (virtual) serial machine on top of a parallel machine. The overall structure of codelets reading and writing on the *Working Memory* configures a fully parallel multi-agent system. The constraints of the *SpotlightController* and the broadcast mechanism implements on top of it the emergence of a serial stream which is the consciousness. But this serial stream is not just any serial stream. It focuses attention on the most important kind of information in each time step. It builds what Koch called an *executive summary* of information [16]. This is one of the main advantages of this technology: to focus attention on what is most important and spreading this to all agents in the multi-agent system. This simple understanding opens a large set of opportunities to future research. Now we are able to improve this main idea and check other uses for such a technology.

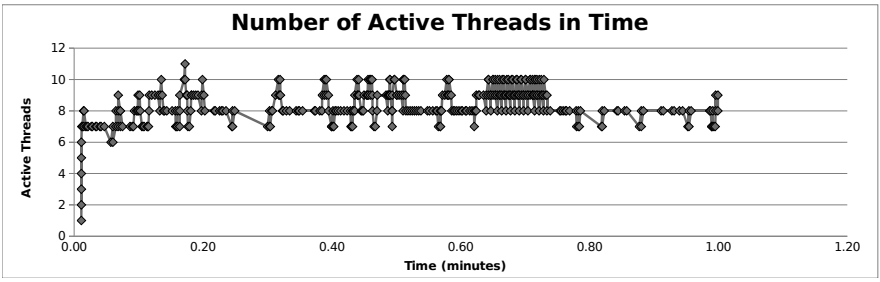


Fig. 6. Number of Active Threads in Time

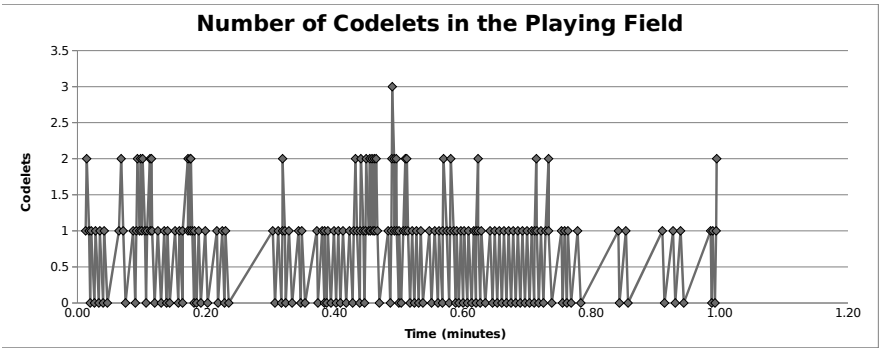
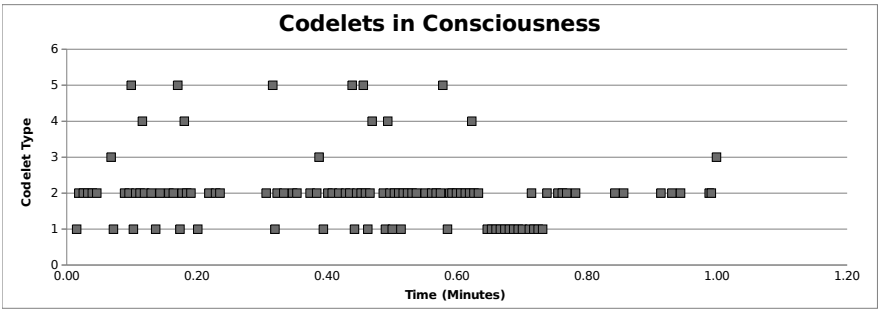


Fig. 7. Number of Codelets in the Playing Field



1 - PlanGenerator 2 - ObstacleRecorder 3 - TargetCarrier
4 - CollisionDetector 5 - PathChecker

Fig. 8. Types of Codelets in Consciousness

4.2 A Quantitative Analysis

Some data related to the experiment can be viewed in figures 6, 7 and 8.

Figure 6 shows the number of active threads at each instant of time. We can see that an average of 8 threads are working at the same time. Figure 7 shows the number of codelets running at the same time at the playing field. An average

of 1 or 2 codelets were at the playing field at the same time. The maximum of codelets at the playing field at the same time was 3. Finally, figure 8 shows the different types of codelets accessing the consciousness at each time. We can see that most of the time the codelet *ObstacleRecorder* was at consciousness. The second more frequent was *PlanGenerator*. The other three, *TargetCarrier*, *CollisionDetector* and *PathChecker* were less frequently at the consciousness.

These data refer to 1 minute of simulation. The subsequent instants of time show more or less the same behavior. Other codelets, like e.g. *LowEnergy*, also appear from time to time, but they didn't appear in the time-frame shown in the figure.

5 Conclusion

BFA is shown to be a very flexible and scalable architecture, due to its consciousness and behavior network mechanisms implemented through independent codelets. Newer features can be easily included by means of newer codelets performing new roles. Consciousness mechanism makes possible a deliberation process that enables the perception of most relevant information for the current situation, building what Koch called an executive summary of perception. Much work remains to be done, especially related to a better model formalization and a better understanding of the overall role of coalitions. However, seen as an embryo of a conscious artificial creature, the first results of this study show the feasibility of such techniques, motivating our group to continue on this line of investigation.

References

1. Blackmore, S.: Consciousness - A very short introduction. Oxford University Press, Oxford (2005)
2. Bogner, M.B.: Realizing "Consciousness" in Software Agents. PhD thesis, The University of Memphis (December 1999)
3. Chella, A., Manzotti, R.: Artificial Consciousness. Imprint Academic (2007)
4. Gamez, D.: Progress in machine consciousness. Consciousness and Cognition 17, 887–910 (2008)
5. Franklin, S., Graesser, A.: A software agent model of consciousness. Consciousness and Cognition 8, 285–301 (1999)
6. Negatu, A.S.: Cognitively Inspired Decision Making for Software Agents: Integrated Mechanisms for Action Selection, Expectation, Automatization and Non-Routine Problem Solving. PhD thesis, The University of Memphis (August 2006)
7. Baars, B.J.: A cognitive theory of consciousness. Cambridge University Press, Cambridge (1988)
8. Dubois, D.: Constructing an Agent Equipped with an Artificial Consciousness: Application to an Intelligent Tutoring System. PhD thesis, Université du Québec à Montréal (August 2007)
9. Balkenius, C.: Natural Intelligence in Artificial Creatures. Lund Univ. Cognitive Studies 37 (1995)

10. Selfridge, O.G.: Pandemonium: a paradigm for learning. In: Mechanism of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory, pp. 513–526. HMSO, London (November 1958)
11. Jackson, J.V.: Idea for a mind. ACM SIGART Bulletin xx(101), 23–26 (1987)
12. Hofstadter, D.R., Mitchell, M.: The copycat project: A model of mental fluidity and analogy-making. In: Holyoak, K.J., Barnden, J.A. (eds.) Advances in Connectionist and Neural Computation Theory, vol. 2, pp. 31–112 (1994)
13. Maes, P.: How to do the right thing. Connection Science Journal 1, 3 (1989)
14. da Silva, R.C.M.: Análise da arquitetura baars-franklin de consciência artificial aplicada a uma criatura virtual. Master's thesis, DCA-FEEC-UNICAMP (July 2009)
15. Gudwin, R.R.: Contribuições ao Estudo Matemático de Sistemas Inteligentes. PhD thesis, DCA-FEEC-UNICAMP (1996)
16. Koch, C.: The Quest for Consciousness - A Neurobiological Approach. Roberts & Company Publishers, Englewood (2004)

Simulating the Emergence of Social Relationship Networks in Groups of Believable Agents: The X-BARIM Model

Pablo Barbosa, Danielle Silva, Geber Ramalho, and Patricia Tedesco

Centro de Informática, UFPE, Cidade Universitária, Recife/PE, Brasil
{psb,drds,glr,pcart}@cin.ufpe.br

Abstract. Believable agents are intelligent agents designed to emulate characteristics such as personality, emotions and relatioships in order to exhibit an ilusion of life to human observers. Hence, designers should not only cater for the individual behaviour, but also consider their social interactions. In this light, this paper describes a model for interaction and relationships in groups of believable agents. The model proposed, X-BARIM, was based on consolidated Psychology studies and aims at enhancing the social simulation credibility. Experiments performed with the model have shown that it is possible to simulate well known phenomena such as Leadership Emergence and Coalition Formation even for larger groups of agents.

Keywords: Believable Agents, Social Interactions, Social Simulations.

1 Introduction

Believable Agents arose from the need to simulate aspects of human behavior in virtual environments inhabited by animated characters (e.g. games). In such environments, it is often desired that virtual characters exhibit a behavior that is not necessarily optimal in terms of problem-solving, but similar to the behavior that humans would present in similar situations, thus providing an illusion of life.

In order to build believable agents, one must consider elements such as personality, emotions and relationships, since they ensure the agents' behavior credibility. Human personality may be seen as a set of features that would be used to describe somebody (for instance, some people are outgoing whereas others are shy). Emotions are intense feelings directed to someone or something. Relationships are connections that arise between individuals who live and interact with each other for a certain period of time. Although several studies about the use and modeling of believable agents (e.g. [1],[2],[3],[4]) have been conducted, most of them focus only on the first two elements (i.e. the individual aspects), leaving aside the relationships.

Relationships represent the history of interactions between characters [4]. Therefore, in simulations where interactions between agents or between them and the player are needed, the lack of a relationship model leads to a significant reduction in the agents' credibility, since agents cannot respond consistently with respect to past interactions. Thus, in a game, for example, an agent that does not have a relationship

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 133–142, 2010.

model is not be able to discern which players interacted with him in a hostile or friendly way, making it difficult to decide which are potential allies or enemies.

In this scenario, BARIM [5], acronym for Believable Agents Relationship and Interaction Model, arose as a model for simulating the emergence of social relationships. Since it is based on Social Psychology theories, Barim has achieved important goals with respect to group simulation. However, many important concepts found in Social Psychology, such as multiple relationships[6] and interactions of influence, that allow better understanding of leadership[7], asymmetry[8] and structural balance[8], were ignored. These concepts model characteristics inherent to human relationships and are useful for more sophisticated and realistic simulations. Thus, this paper presents X-BARIM, a model that extends BARIM with all those concepts. The experiments performed with X-BARIM demonstrated that the model is suitable for a larger and more sophisticated set of simulations when compared to BARIM, allowing for the simulation of phenomena such as Leadership Emergence and Coalition Formation.

The remainder of this paper is organized as fallows. Section 2 outlines the background knowledge on Interactions and Relationships. In Section 3 we present the state of the art, together with BARIM, the model we have based X-BARIM on. In Section 4 we propose a new Interaction and Relationship model for believable agents. The experiments used to validate the new model are presented in Section 5. Finally, our conclusions and suggestions for further work are presented in Section 6.

2 Interactions and Relationships

There are basically two groups of relationships in interpersonal environments [9]. In the first group, are those defined by the social system, with very specific names, (e.g. family relationships such as father, son, uncle, nephew, etc.) or hierarchical relationships, such as leadership and subordination within a company. In such cases, individuals have a role that clearly defines the relationship between them, their obligations and expected behavior. These are called *formal* or *named* relationships.

In the second group are the so-called informal relationships, which are those not established by the social system and that generally have looser definitions, such as friendship, trust, respect, etc. When someone, for example, sees a young man and an older person by his side, they may find that the two have a relationship of the first group - father and child or teacher and student - or of the second - they are just friends. In general, formal and informal relationships tend to be linked, since a formal relationship entails some kind of interaction between individuals. And through these interactions informal relationships between individuals arise.

Social interactions can occur in several ways, using different codes and languages. This interpersonal communication can lead to the formation, maintenance and dissolution of multiple human relationships. When a person perceives an interaction, they react according to the classification they, internally, give to the interaction, changing their social relations accordingly [4]. For instance, if Mary tells Peter a joke, and he considers it offensive for some reason, he may like Mary less or change his relationship with her from "like" to "dislike". Hence, without a relationship and interaction model, the credibility of a believable agent in a group simulation could be greatly impaired, thus reducing the quality of the overall simulation.

3 Related Work

Several research groups attempted to develop models for believable agents, such as: Virtual Theater Project [10], Project GBM [11], Cathexis [12], PM / SME [13], μSic [2], the model based on valence of emotions and personality [14], the extended model PPP [15], the Par / PARSYS project [16], X-PCSA [1] and the SGD model [4]

All of these use Psychological theories to model personality and/or emotions, but few model social interactions between agents. Even in works that study groups of believable agents, the focus is on the individual actor, consistency of the story [10] or user participation as a member of the group [4]. Very little attention (if any at all) is given to the social networks that emerge from interactions.

A relationship and interaction model based on consolidated studies of Psychology would greatly enhance the credibility of the believable agent group simulation. That way it is possible to use known phenomena in psychology, as leadership emergence and coalition formation, to validate the feasibility of the model.

BARIM [5] (Believable Agents Relationship and Interaction Model) emerged to fill in this gap. It is a model of interactions and relationships for multi-agent systems, capable of simulating both the emerging relationships between agents (artificial or human). This model is composed of two main elements: a set of interactions and a relationship structure, as follows:

- **Interactions:** The agents communicate with each other through a set of interactions that impact agents' relationships directly and indirectly. The types of interactions are defined according to the application domain. Each interaction is classified as positive, negative or neutral. These classifications are called socio-emotional functions. This model also strongly suggests categorizing the interactions according to the Interaction Process Analysis (IPA) defined in [17].
- **Relationships:** Since relationships are affected by interactions, it is possible to say that the relationships between agents function as a record of their past interactions, and consistent behavior should take them into account. There are three levels of relationships: positive, neutral and negative. Every interaction that is not neutral affects relationships. To guide indirect interactions - those that are perceived by the actors, although not directed to them - the Balance Theory [18] was adopted.

3.1 Open Issues

A tool for group simulation was also built, demonstrating that BARIM [5] is suitable for simulating groups of believable agents. However, it is impossible to use it to map certain characteristics inherent to human relationships, such as multiplicity, symmetry [8] or influence [7], which add realism to simulation. This model also has shortcomings with regards to the notion of balance, which is not appropriate for groups, and to the assignment of the interaction value (it is not possible to assign different values to interactions that occur in different situations). Hence, we felt motivated to build a new model for believable agents' interaction and relationships, considering the issues not tackled by BARIM.

4 X-BARIM

X-BARIM is a novel model of interaction and relationship of believable agents, which arose from the refinement and extension of BARIM. The proposed improvements may entail a greater credibility in the final simulation, allowing it to be used in contexts unavailable for BARIM. Such contexts include: (1) according to [8], the concept of balance used by BARIM is not suitable for social simulations. X-BARIM uses Structural Balance [8] to solve this problem; (2) BARIM just considers a single relationship among agents, and does not specify the considered relationship. X-BARIM works with multiple relationships, thus providing a more realistic model [6]; and (3) the signal of the interaction value is fixed in BARIM, so a specific interaction always has positive or negative impact on relationship. In X-BARIM the interaction signal is defined by the agent, and it may have different effects over different relationships.

Like BARIM, this model is composed of two main elements: a structure of relationships and a set of interactions that modify these relationships, detailed below.

4.1 Relationships

Just as in BARIM, agents' relationships reflect the history of their past interactions. A social network is seen as a social structure made of nodes, generally individuals or organizations, which are tied together by one or more types of interdependency [6], such as values, visions, ideas, etc. Social networks can be represented as graphs.

The first extension to BARIM is related to relationship **multiplicity**. BARIM only considers global relationships. Therefore, for any two agents, if the relationship between them is positive, it means that they had a greater number of positive interactions than negative in the past. But, in fact, more realistic networks consider multiple connections between nodes [6]. Thus, in a working relationship between supervisor and assistant, both trust and respect can exist. Hence, in X-BARIM, we consider the possibility of multiple relationships between agents.

For each relationship there are three ranges of values, indicating whether it is positive, negative or neutral. Considering two agents A1 and A2, the *like* relationship could be positive. However, the *trust* relationship could be negative. This refinement made X-BARIM suitable for use in more credible simulations, since agents may use the information contained in each specific relation for their decision making. For instance, an agent could decide not to request help for a task to another agent with whom it has a negative trust relationship, but it may have a positive like relationship with the same agent and invite it for a coffee.

Another change was related to the network nodes' symmetry. In BARIM the value of the relationship is an attribute common to the connection nodes, i.e. all connections are symmetric. In other words, if we consider two agents A1 and A2, one can say that the link between them is positive, negative or neutral. But it is not possible to say that A1 likes A2 and A2 dislikes A1. But in Psychology, attitudes are evaluative statements related to objects people or events and reflect how the individual feels about something [19]. In X-BARIM attitudes are attributes that belong to each agent. This means that the links are not necessarily symmetrical.

And finally, the last change in the relationship network was regarding to its balance. BARIM adopts the Balance Theory [18], to guide the agents' perceived interactions and which are not directed to them. However, the definition of balance in that theory is applicable only to a limited number of situations. Two problems of its use should be noted: (1) **Non-symmetric relations**. In Balance Theory, there is no definition of balance to cover non-symmetrical relations; and (2) **Networks with more than three entities**. The theory refers to units of three entities. For more realistic simulations, we need to consider social balance for larger networks.

So we suggest the use of structural balance theory [8], which solves the problems described above. The structural balance is a generalization of the Balance Theory, more associated with groups. Using the concepts of digraph, weighted graph and sign of cycle, [8] showed that a weighted digraph is balanced if all its cycles are positive. This formalization of balance is more appropriate for X-BARIM's relationship networks, since we work with groups of agents and asymmetric networks.

4.2 Interactions

In a closed set of people, which initially do not know each other and which will interact over a period of time, connections will arise spontaneously [20]. In BARIM, interactions and relationships are certainly interconnected, because the relationships result from agents' interactions. When agents interact, the relationships between them are modified according that interactions' value.

The first point of discussion about interactions is related to the rating value associated to them. In BARIM this value is set *a priori*, once the interaction is defined. There is also a suggestion to group interaction according to the categories defined by [17] (e.g. looks friendly and agree, which are always have positive rating), and a rating is assigned to each category, so all interactions of a category have a same value. However, let us consider a situation where a person decides to interact using something in the category "reduce tension", such as telling a joke at the beginning of a meeting. In BARIM the influence of this category is considered positive. But what if, in the context presented, the boss, also present the meeting, is just about to report that the company is in financial difficulties, and that there will be layoffs? In this case, it is possible that he does not understand the interaction of that person as positive. Indeed, it is possible that someone perceives it as negative, simply because they do not like the author of the joke.

Another important point is that the same interaction may have different ratings in different relationships. Thus, if a person fails to perform a task, you can lose confidence in this person's work, but not necessarily cease to like them.

Because of situations like that, we argue that it is the agent that perceives the interaction that should rate it. A function that calculates the rating value for the interaction could consider as inputs, for example, the state of the agents who perceive the interaction, their relationship with the interaction's author, the network balance, and the interaction itself. All that is required is to calculate the rating value of the interaction perceived by them. This value should be calculated for each relationship between agents, and then update their relations according to this value.

4.3 Discussion

X-BARIM can be used to guide new developers that use groups of believable agents in their applications. The first proposed extension to BARIM was the addition of multiplicity in social relations. Multiple networks are inherent of real environments, hence, a model considering them is more realistic. Relations also became an individual agent attribute, which allows, for example, the visualization of phenomena such as asymmetric tension [8].

Another change related to relationships model was the revision of the balance concept. The balance theory presented in BARIM does not have a definition of balanced groups, and groups of agents are the focus this work. So, we presented the concept of structural balance, which extends the notion of stable states for groups, thus being more consistent with the work's context.

These changes have brought a new model of interactions to light. So now the same interaction may influence different types of relationships. These interactions also no longer have a predefined positive or negative value. This value is the result of the agent´s cognition, which increases the credibility of the agents' interactions.

5 Experiments and Results

A set of simulations was conducted to validate the concepts underlying X-BARIM. From the analysis of agents' social relations we identified phenomena such as leadership emergence and coalitions. In this light, the actor that developed more relationship connections was considered the group leader [21], and coalitions were identified by checking the network partitioning [8].

The first set of simulations was designed to test interaction and symmetry concepts. The second set was focused on the multiplicity of relationships. For each experiment were performed groups of fifty simulations, and graphs shown below represent the average relationship value. We use a function that calculates the value of social-emotional interaction considering: 1) the state actors who perceive the interaction, 2) their relationship with the author's interaction, 3) and the interaction itself.

5.1 Simulation Scenario

We considered a scenario where actors work cooperatively to complete a given task. Each task has a size and information on how much work has been done. A certain number of tasks is assigned to each actor. The simulation runs until all actors have completed their tasks. In each turn the agent tries to perform a task, and may have three possible results: success, error or failure. In the case of success, we increment the parameter *completed work*. In the case of error, the parameter *completed work* is decremented, meaning that the group as a whole will have extra work to do. And finally, if the result is a fail, the *completed work* parameter remains unchanged. This scenario ensures that there is an interdependence between the actors, as the result of an individual influences the outcome of the group.

The agents have only one personality trait, extroversion. This is defined as the agent's initiative and influences how often it will participate in social interactions. The

agents also have the ability attribute that represents their competence in performing a task. The higher the skill, more likely the result of its execution will be a success.

5.2 Interactions and Relationships

We used a turn based protocol. During each turn actors try to accomplish their task, and may engage in interactions with other actors When the result of a task execution is an error or a failure, the agent can ask help from another agent, who either denies it, or leaves their task to help. So, one of the interactions available to the agent is *ask for help*. The interactions *help* and *deny help* may be answers to the first interaction.

These interactions will influence the agents "trust" relationship. We are also considering the emergence of friendship between agents, represented by the "like" attitude. For simplicity, just the interactions *friendly*, with positive effect, and *unfriendly*, with negative effect influenced the *like* relationship,. Each agent actively searches for friends and works to maintain friendships. Actors with high extroversion have a greater tendency to form friendship networks than those with low extroversion.

5.3 Experiment: Interaction and Symmetry

The objective of this simulation is to demonstrate how the concept of symmetry can be used to make more realistic simulations. In this experiment we used the "trust" relationship and the interactions related to this relationship, which are *ask for help*, *help* and *deny help*. Figure 1 represents the resulting agent networks. Each agent is represented by a circle and relationships are the links between them.

The simulations had three agents. Agents A and B were initialized with a skill level of 90, while C had a skill of 60. Figure 1.1 represents the network state after 15% of all tasks were completed. "C", which has the lowest skill, has used a lot of help from "A" and "B", however when these two needed help from "C", it was denied because "C" is late with his tasks. As "A" and "B" still maintain a neutral stance with "C", they are still helping. But the low negative "trust" values from "A" and "B" to "C", generate a symmetry tension [8], due to “C”s refusal to aid them.

The final result of the simulation is shown in Figure 1.2. Agents A and B have formed a coalition and left C isolated. With BARIM, the "help" interaction would be always positive. Since C often asks for help, the "trust" relationship would be always positive. Consequently, A and B would keep asking C for help.

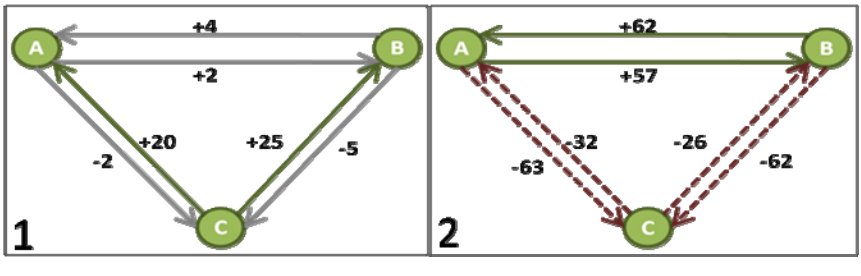


Fig. 1. 1)Trust relationship between agents A, B and C after 15% of the simulations.2) Final simulation result.

5.4 Experiment: Multiple Relationships

For this simulation we used five agents A, B, C, D and E that were initialized with the skills (80, 90, 80, 70, 70), and extroversion values (70, 80, 70, 30, 60). We chose high and low skills and medium value extroversion to ease the observation of leader emergence and coalition formation. Figure 2 shows the agents trust graph. All agents trust B with an average of 76.75, the highest in the group. Considering that the agent with the best relationship network can be considered the group leader ([5,19,21]) B, stands out as leader of the group due to its competence. Figure 2 shows the group friendship graph. Due to the low extroversion of D, this agent was eventually removed from the friendship group formed by A, B, C and E, resulting in the formation of subgroups. Due to being the more outgoing, B has the highest average positive links, with an average of 51.75. Thus, B was elected leader by reference. B, the more skilled and sociable, stood out as innate leader.

Each of the experiments presented in this section showed specific features of the new model, such as multiple relationships, asymmetry and new balance. In particular, in the second experiment it was possible to observe the phenomena of competence and referent leadership, and the formation of coalitions. Although the leader of competence and reference was the same agent, this was due to the fact it has the highest values for ability and extroversion.

Most of these concepts were not handled by BARIM, which prevents this type of simulation for those using that model only. Thus, X-BARIM appears to be more suitable to be used in a more complex and realistic experiments with groups of agents.

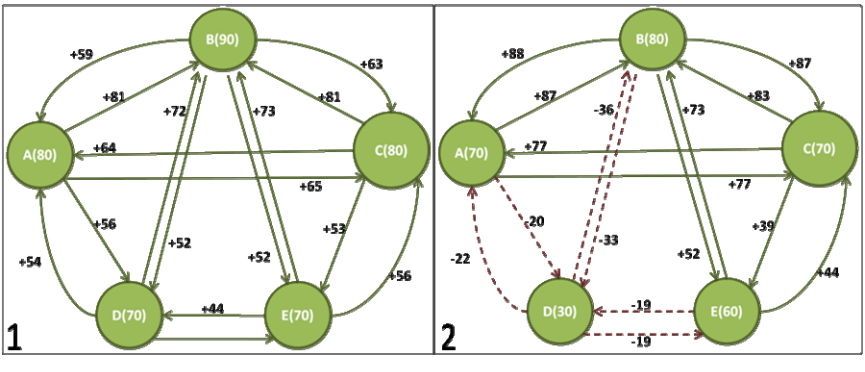


Fig. 2. 1) Final state for trust network. 2) Final state for friendship network.

6 Conclusions and Further Work

Believable agents are intelligent agents designed to emulate characteristics such as personality and emotions in order to exhibit credible behavior. Several studies in the area (e.g. [1], [2], [4], [10]) have used psychological models in an attempt to increase the realism of these actors. However, most of them focus only on personality and emotions, ignoring the relationships between agents.

Given this scenario, BARIM [5] was presented as a model of interaction and relationships, based on studies of Organizational Psychology. Despite its simplicity, it was possible to observe phenomena of coalition and leadership in the experiments carried out to validate the simulation model.

However, there were a few drawbacks, which undermine the credibility of the simulation, such as the use of the concept of balance, which is inappropriate for groups [6] and the assignment of the interaction value, where it is not possible to assign different values to the same interaction that occurs in different situations. Thus, X-BARIM was presented as a new interaction and relationships model for believable agents, which fixes BARIM problems and extends it with other theories from Social Psychology, such as multiplicity of social relations that are inherent to real environments [6], balance definition extended to groups [8], ability to simulate new phenomena, like symmetry tension[8], interactions which its value is the result of the agent cognition. We aimed a believable agent whose behavior is close to a human behavior, and furthermore, it must have consistency of its behavioral states and some degree of predictability [22]. In our experiments, for example, agents with high extroversion developed more friendship connections and agents with high skill developed more trust connections, which demonstrates the consistency of the simulation. Then it was shown through simulations that the model proposed is suitable to be used in a larger and more sophisticated set of simulations than BARIM.

In the near future, we intend to work on extending the current model to aggregate more features, such as to combine this model with personality and emotions models as X-PCSA[1] model and to add attraction to the model.

References

1. Silva, D.R.: Atores Sintéticos em Jogos Sérios: Uma Abordagem Baseada em Psicologia Organizacional. Ph.D. Thesis. CIn - UFPE. Recife, Brazil (2009)
2. Namee, B.M., Cunningham, P.: The μ-SIC System: A Connectionist Driven Simulation of Socially Interactive Agents. Technical Report. Trinity College Dublin, Department of Computer Science. Dublin, Ireland (2002)
3. Bates, J.: The Role of Emotion In Believable Agents. Communications of the ACM 37(7), 122–125 (1994)
4. Prada, R.: Teaming Up Humans and Synthetic Characters. Ph.D. Thesis. Universidade Técnica de Lisboa, Portugal (2005)
5. Dominoni, E.: Simulação de relacionamentos sociais emergentes em grupos de atores sintéticos: o modelo BARIM. In: Centro de Informática, UFPE, Recife, Brasil (2007)
6. Kadushin, C.: Introduction to Social Network Theory, `http://home.earthlink.net/~ckadushin/`
7. Chiavenato, I.: Gestão de Pessoas. Editora Campus, São Paulo (2009)
8. Cartwright, D., Harary, F.: Structural balance: A generalization of Heider's theory. Psychological Review 63(5) (1956)
9. Bates, J.: The Role of Emotion in Believable Agents. CACM 37(7), 122–125 (1994)
10. Doyle, P., Hayes, B.: Guided Exploration of Virtual Worlds. In: Network and Netplay: Virtual Groups on the Internet. MIT Press, Massachusetts (1998)
11. Rizzo, P., Veloso, M.V., Miceli, M.: Personality-Driven Social Behaviors in Believable Agents. In: Fall Symposium on Socially Intelligent Agents, Massachusetts, USA (1997)

12. Velasquez, J.D.: When Robots Weep: Emotional Memories and Decision-Making. In: Proceedings of the 15th National Conference on AI, Wisconsin, USA (1997)
13. Egges, A., Kshirsagar, S., Thalmann, N.: Generic personality and emotion simulation for conversational Agents. Technical Report. MIRALab, University of Geneva. Geneva, Switzerland (2004)
14. Ventura, R.: Emotion-based Agents. In: Workshop of the Third International Conference on Autonomous Agents, Seattle, USA (2000)
15. Beale, R., Wood, A.: Agent-Based Interaction. In: Proceedings of the HCI, Glasgow, UK (1994)
16. Allbeck, J., Badler, N.: Toward Representing Agent Behaviors Modified by Personality and Emotion. In: Embodied Conversational Agents at AAMAS 2002, Bologna, Italy (2002)
17. Bales, R.F.: Interaction Process Analysis. The University of Chicago Press, Chicago (1950)
18. Heider, F.: Attitudes and Cognitive Organization. The Journal of Psychology 21, 107–112 (1956)
19. Robbins, S.P.: Fundamentos do Comportamento Organizacional. Pearson Prentice Hall, São Paulo (2004), ISBN 85-87918-64-8
20. Zeggelink, E.: Dynamics of structure: An individual-oriented approach. Social Networks 16, 83–110 (1994)
21. Castilho, A.: Liderandos Grupos, um enfoque gerencial. Editora Quality (2005)
22. Ortoni, A., Clore, G., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge (1988)

Using Jason to Develop Normative Agents

Baldoino F. dos S. Neto[1], Viviane Torres da Silva[2], and Carlos J.P. de Lucena[1]

[1] PUC-Rio, Informatics Department, LES, Rio de Janeiro - Brazil
[2] Universidade Federal Fluminense, Computer Science Department, Niterói - Brazil
{bneto,lucena}@inf.puc-rio.br, viviane.silva@ic.uff.br

Abstract. Norms have become one of the most promising mechanisms of social control to ensure a desirable social order in open multi-agent systems where autonomous, heterogeneous and independently designed entities can work towards similar or different ends. Norms regulate the behaviour of agents by defining permissions, obligations and prohibitions, and by stating stimulus to their fulfilment while defining rewards and discouraging their violation while pointing out punishments. Since goal-oriented agents' priority is the satisfaction of their own desires, they must evaluate the positive and negative effects of the fulfilment or violation of the norms before choosing to comply or not with them. In this context, we present the new functions of the Jason platform defined to support normative reasoning, i.e, to build agents able to deal with desires and norms. Agents are then able to check if they should adopt or not the norm, evaluate the effects of the fulfilment or violation of the norm on their desires, detect and solve conflicts among norms, and select desires and plans according to their choices of fulfilling or not a norm. We demonstrate the applicability of such new functions through a non-combatant evacuation scenario.

1 Introduction

Open multi-agent systems are societies in which autonomous, heterogeneous and independently designed entities can work towards similar or different ends [7]. In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, those systems establish a set of norms that is used as a mechanism of social control to ensure a desirable social order in which agents work together [7].

Such norms regulate the behaviour of the agents by defining obligations (indicating that agents are obligated to accomplish something in the world), permission (indicating that agents are permitted to act in a particular way) and prohibitions (indicating that they are prohibited to act in a particular way)[9]. Moreover, norms may give stimulus to their fulfilment by defining rewards and may discourage their violation while stating punishments [12].

Over the last years, several approaches have been proposed on the specification and implementation aspects of norms, such as [4] and [12]. Others have focuses on the definition of parts of an infrastructure to be used by BDI agents [11] to reasoning on norms, such as [8] [6]. However, there is still a need to define an agent-oriented platform able to guide the implementation of goal-oriented normative agents, i.e., agents that have the main purpose of achieving their desires while trying to fulfil the system norms.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 143–152, 2010.

From the set of main used agent-oriented platform such as [1] [5], none provides support to build normative agents.

In this context, we present the new functions of the Jason platform [1] defined to support normative reasoning, i.e, to build agents able to deal with desires and norms. The original Jason platform already provides support to the implementation of BDI agents and a set of hot-spots that enable the implementation of normative functions. By using the new functions being proposed, it is possible to build BDI agents able to check if they should adopt or not a norm, evaluate the effects, on their desires, of the fulfilment or violation of the norm, detect and solve conflicts among norms, and select desires and plans according to their choices of fulfilling or not a norm.

We demonstrate the applicability of the new functions we have defined through a non-combatant evacuation scenario where the tasks related to adopt, evaluate, and comply norms are shown. The paper is structured as follows. In Section 2 we outline the background about norms. In Section 3 the Jason platform is explained, in Section 4 the scenario used to present the work is presented and in Section 5 we present the normative Jason platform and use the scenario to exemplify it. Section 6 summarizes relevant related work and, finally, Section 7 concludes and presents some future work.

2 Norms

In this work, we follows the norm representation described in [12], as shown below: *norm* (*Addressee*, *Activation*, *Expiration*, *DeonticConcept*, *State*, *Rewards*, *Punishments*) where *Addressee* is the agent or role responsible for fulfilling the norm, *Activation* is the activation condition for the norm to become active, *Expiration* is the expiration condition for the norm to become inactive, *Rewards* are the rewards to be given to the agent for fulfilling a norm, *Punishments* are the punishments to be given to the agent for violating a norm, *DeonticConcept* indicates if the norm states an obligation or a prohibition[1], and *State* describes the set of states being regulated. In this paper we are only dealing with norms that restrict the achievement of a given state. We are not considering norms that directly regulate the execution of actions since in Jason it is not possible to apply unification between internal actions.

3 Jason Platform

Jason is an interpreter for an extended version of AgentSpeak proposed by Rao [10] that gives support to the creation of BDI agents. Figure 1[2] (reproduced from [1]) illustrates the MAS platform provided by Jason. Sets (of beliefs, events, plans, and intentions) are represented as rectangles, diamonds are used to represent selection functions (of one element from a set) and circles to represent some of the processes involved in the interpretation of AgentSpeak programs.

Each interpretation cycle updates the list of events according to the perception coming from the environment, to the messages the agent receives and to the information

[1] In this paper we assume that everything is permitted unless a prohibition is stated.

[2] The dark, internal diamonds were defined in the normative version of the platform (NRF, UN, EN, DSC, ADP, Sg, Sp).

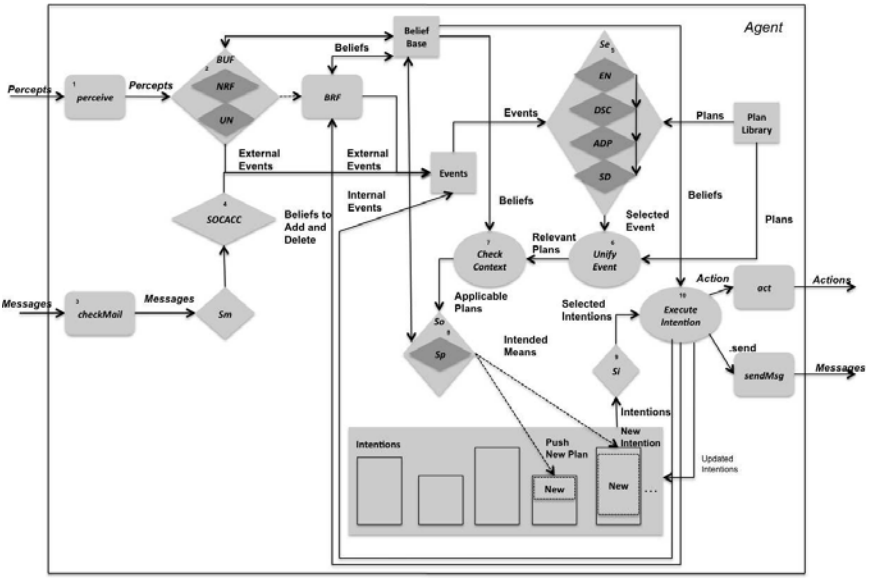


Fig. 1. Normative Jason Platform

coming from the agent's own execution of a plan. In the *Belief Update Function (BUF)* the perceptions and actual agent's beliefs are used to update the *Belief Base* and to update the set of events to be carried on. From the set of messages received, the *Message Selection Function (SM)* selects one to be handled. The *SocAcc function* can filter the messages based on the characteristics of the sender. The *Belief Review Function (BRF)* revises the *Belief Base* with a literal to be added or deleted, and the intention structure that required the belief change. A single event is selected in the *Event selection function (SE)* that is unified with triggering events in the heads of plans by the *Unify Event cycle* generating a set of all relevant plans. The context of such plans are verified according to the *Belief Base* by the *Check Context cycle* generating a set options. The *Option Select Function (SO)* selects a single applicable option from the set of options, which becomes the intended means for handling the selected event. The option either pushes the plan on the top of an existing intention (if the event was an internal one), or creates a new intention in the set of intentions (if the event was external, i.e., generated from perception of the environment). The *Intention Select Function (SI)* selects one of the agent's intentions that is executed by the *Execute Intention cycle*. When all formula in the body of a plan have been executed, the whole plan is removed from the intention list, and so is the achievement goal that generated it. This ends a cycle of execution, and AgentSpeak starts all over again, checking the state of the environment after agents have acted upon it, generating the relevant events, and so forth.

4 Scenario: Rescue Operation

Our implementation is based in the simplified non-combatant evacuation scenario, presented in [2]. In such scenario agents are responsible to plan the evacuation of members

of a Non-Governmental Organisation (NGO) that are in hazardous location. The agents operate in different areas with different resources, such as: *(i)* different Autonomous Unmanned aerial Vehicles (AUVs), deployed with sensors to provide information about the enemies like types of weapons used by them, location of their bases and strategies, and information about the area operated by each agent like information about the weather, *(ii)* helicopters, *(iii)* troops and *(iv)* land-based helicopters. Considering that such resources are limited, we have a *Commander Agent* that is responsible to regulate the behaviour of the agents and the use of the resources according to the following norms:

Norm1
Addressee: *Rescue Entity*
Activation: NGO workers are stranded in a hazardous location
Expiration: NGO workers are stranded in a safe location
DeonticConcept: Obligation
State: To evacuate NGO workers
Rewards: The *Commander Agent* gives more troops to *Rescue Entity*.
Rewards: The *Commander Agent* gives land-based helicopters to *Rescue Entity*.
Punishments : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their troops.

Norm 2
Addressee: *Rescue Entity*
Activation: The weather is bad
Expiration: The weather is good
DeonticConcept: Prohibition
State: To evacuate NGO workers
Punishments : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their helicopters. **Punishments** : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their land-based helicopters.

Norm 3
Addressee: *Rescue Entity*
Activation: The weather is bad
Expiration: The weather is good
DeonticConcept: Prohibition
State: To use helicopters.
Rewards: The *Commander Agent* gives more troops to *Rescue Entity*.
Rewards: The *Commander Agent* gives land-based helicopters to *Rescue Entity*.
Punishments : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their troops.

5 The Normative Jason Platform

The implementation of the Jason platform proposed in this paper aims to help agents on reasoning about the system norms. Norm is considered a primary concept that influences the agent while reasoning about its beliefs, desires, plans and intentions. In a nutshell, the extended Jason platform modifies the original Jason platform by including the following functions[3], as illustrated in Figure 1: (Norm Review Function - **NRF**) this function helps the agent on recognizing its responsibilities towards other agents by incorporating the norms that specify such responsibilities. That is, the main goal of this

[3] The normative Jason platform together with its new functions are available at http://wiki.les.inf.puc-rio.br/index.php/NormativeAgent

function is to update the set of adopted norms taking into accounting the perceptions and the information in the belief base; (Updating Norm - **UN**) after reviewing the adopted norms and the beliefs, some norms' activation conditions and deactivation ones can be trigged. Therefore, the main task of this step is dedicated to update the set of activated and adopted norms; (Evaluating Norm - **EN**) this step helps the agent on selecting, from the set of activated norms, the norms that it has the intention to fulfil and the ones it has the intention to violate; (Detecting and Solving Conflicts - **DSC**) this function checks and solves the conflicts among the norms; (Annotating Desires and Plans- **ADP**) after deciding the norms to be fulfilled, the desires and the plans are annotated with a priority level. The desires that positively influence the fulfilment of the norm and the plans able to fulfil the norms receive highest priority; (Selecting Desires - **SD**) the main goal of this step is to select the desires that will become intentions taking into according their priorities. By default the desire with highest priority is selected; (Selecting Plans - **SP**) this function chooses a single applicable plan from the *Plan Library* according to their priorities. By default the plan with highest priority is selected.

5.1 Norm Review Function (NRF)

This function recognizes from the set of receiving perceptions the ones that describe norms. After recognizing the norms, such function reviews the set of adopted norms applying the following verifications: (i) it checks if the new norm unifies with one of the norms already adopted, i.e., if the incoming norm already exists in the agent *Belief Base*, and (ii) it verifies if the agent is the addressee of the norm, i.e., if the field *Addressee* of the new norm unifies with the agent role or agent name, also stored as a belief in the *Belief Base*. Finally, such function updates the set of adopted norms in the *Belief Base* if the new norm does not already exist and the agent is the addressee of the norm.

With the aim to exemplify the use of this function, let's consider the scenario presented in Section 4 where two groups of agents are leaded by *Agent A* and *Agent B* playing the role *Rescue Entity*. When these entities receive information about the three system norms, the NRF function is executed comparing the addressee information with the role being played by the agents and checking if the norms are not stored yet in the agent's belief base.

5.2 Updating Norm (UN)

UN function updates the set of activated and adopted norms checking if the fields *Activation* and *Expiration* of the norm unifies with the beliefs of the agent. If the activation conditions unify with the beliefs, the adopted norm is activated. In the expiration conditions unify with the beliefs, the norm is deactivated and stored as an adopted norm. Note that only the norms that are active must be fulfilled.

Following the example above, if the weather of the area operated by one of the two rescue entities is bad, both norms 2 and 3 are activated, since the activation condition of both norms is "The weather is bad". If the norms are activated, the rescue entity must not rescue NGO workers and must not use helicopters. Both norms are deactivated when the expiration condition unifies with the information about a good weather stored in the agent's belief base.

5.3 Evaluating Norm (EN)

This function evaluates the benefits of fulfilling or violating the norms, i.e., it checks how close the agent gets of achieving its goals if it decides to fulfil or if it decides to violate the norms. In order to do this, the following steps are performed by considering the norms stored in the *belief base*:

1. In case of obligations, i. e., if the field *DeonticConcept* of the norm is equal to obligation, it checks if the state described in the norm is equal to one of the states the agent has the desire to achieve (including the desires stored in the *event library* and the desires that compose the options of the *plan library*). In affirmative cases, it considers that the obligation contributes positively to help the agent on achieving its desires. In any other case, the obligation will not contribute or will contribute negatively.
2. In case of prohibitions, i. e., if the field *DeonticConcept* of the norm is equal to prohibition, it also checks if the state described in the norm is equal to at least one of the states the agent has the desire to achieve. It means that the agent has the desire to achieve a state that it is being prohibited to. In affirmative cases, it considers that the prohibition contributes negatively since it disturbs the achievement of the agent's desires. In any other case, the prohibition will contribute neutrally.
3. After analysing the state , this step considers the influence that the rewards have to the achievement of the agent's desires. We consider that a reward can never influence the agent negatively but always positively or neutrally. In case the reward really helps the agent on achieving its desires, its influence is positive but there may be cases that the reward is useless.
4. Finally, the punishments are evaluated in order to check if they will influence the achievement of the agent's desires negatively or positively.
   a In case the punishment states a change in the agent beliefs, it can influence the agent neutrally or negatively but probably never positively since it is a punishment.
   b In case the punishment states a prohibition and the state being prohibited is one of the agent desires, the punishment will influence negatively since it will disturb the agent of achieving one of its goals. If it is not, the punishment will not influence.
   c In case the punishment states an obligation and the state being obliged is one of the agent desires, the punishment will influence positively (or neutrally) since such state will already be achieved by the agent. Otherwise, the punishment will influence negatively since the agent has not the desire to achieve such state.

Such considerations are summarized in Table 1. Note that instances of this table must be created for each norm in order to individually check its contribution to the achievement of the agent's desires.

In the end, the *EN* function groups the activated norms in two sub-sets: norms to be fulfilled and norms to be violated. Where by default the norm is added to the sub-set *Fulfil* of the activated norms if the contribution for fulfilling the norm is greater than or equal to the contribution for violating the norm. Otherwise, it is selected to be added to the sub-set *Violate*;

Table 1. Evaluating the norms. D means that the agent has the desire/intention, C that the rewards contribute to achieve the desire/intention, O that the beliefs obscure the achievement of the desire/intention, and O that there is not a case of such contribution.

1	*Norm*	*Contribution*		
2		**positive**	**neutral**	**negative**
3	**Obligation**	D	not D	not D
4	**Prohibition**	Ø	not D	D
5	**Reward**	C	not C	Ø
6	***Punishments***			
7	**belief**	Ø	not O	O
8	**Prohibition**	Ø	not D	D
9	**Obligation**	D	not D	not D

Table 2. Evaluating norm 1

1	*Norm*	*Contribution*		
2		**positive**	**neutral**	**negative**
3	**Obligation**	1	0	0
5a	**Reward**	1	0	Ø
5b	**Reward**	1	0	Ø
6	***Punishments***			
9	**Obligation**	0	0	1

Table 3. Evaluating norm 3

1	*Norm*	*Contribution*		
2		**positive**	**neutral**	**negative**
4	**Prohibition**	0	0	1
5a	**Reward**	1	0	Ø
5b	**Reward**	1	0	Ø
6	***Punishments***			
9	**Obligation**	0	0	1

Table 4. Evaluating norm 2

1	*Norm*	*Contribution*		
2		**positive**	**neutral**	**negative**
4	**Prohibition**	0	0	1
6	***Punishments***			
9a	**Obligation**	0	0	1
9b	**Obligation**	0	0	1

In order to exemplify the applicability of this function, let's consider the rescue operation scenario. The evaluation of the benefits of fulfilling and violating the 3 norms are shown in the Tables 2, 4 and 3 that indicates the contribution of each norm element to the achievement of the agent goals. In order to simplify the example, we consider that any norm element generates the same contribution that is 1.

So, from the contribution of the norms shown in the Tables 2, 4 and 3, Norm 1 is included in the set of norms to be fulfilled since the contribution for fulfilling it is equal to "+3" and greater than the contribution for violating it that is equal to "-1". Norm 2 is also included in the fulfil set since the contribution for fulfilling it is equal to "-1" and greater than the contribution for violating it that is equal to "-2". And, finally, Norm 3 is included in the fulfil set since the contribution for fulfilling it is equal to "+1" and greater than the contribution for violating it that is equal to "-1". It indicates that the agent has the intention to fulfil the three norms.

5.4 Detecting and Solving Conflicts (DSC)

The goal of the DSC function is to check and solve conflicts between norms. Such conflicts can happen if two different norms (one being an obligation and the other one a prohibition) specify the same state, both norms have been selected to be fulfilled or to be violated and both norms are active at the same time.

If the agent intends to fulfil the obligation but does not intend to fulfil the prohibition, these norms are not in conflict. The same can be said if the agent intends to fulfil the prohibitions and to violate the obligation. On the other hand, if the agent intends to fulfil both norms or to violate both norms, they are in conflict and it must be solved.

By default, in case of conflicts between two norms that the agent intends to fulfil or violate, this function proposes to select the one with highest contribution to the achievement of the agent's desires. That is, if the contribution coming from the fulfilment of the first norm plus the contribution coming from the violation of the second norm is greater than or equal to the contribution coming from the fulfilment of the second norm plus the contribution coming from the violation of the first norm, the first norm is selected to be fulfilled and the second to be violated.

Considering the norms contribution evaluated in the *EN* function, a conflict between Norm 1 and 2 is detected and should be solved. The conflict is solved by selecting Norm 1 to be fulfilled and Norm 2 to be violated since the contribution coming from the fulfilment of the first norm (+3) plus the contribution coming from the violation of the second norm (-3) is greater than the contribution coming from the fulfilment of the second norm (-1) plus the contribution coming from the violation of the first norm (-4).

5.5 Annotating Desires and Plans (ADP)

The ADP function annotates the desires and plans according to the deontic concept associated with it (obligated or forbidden). In order to do this, each desire or plan is annotated with a level of priority following two steps: **(Events Verification)** The main goal of this step is to annotate the desires in the *Event Library* taking into according the norms the agent wants to fulfil. In others words, if the agent has a desire to achieve a state and there is a norm that oblige the agent of achieving such state, the desire priority is increased according to the importance of the norm. If the agent has a desire to achieve a state and there is a norm that prohibit the agent of achieving such state, the desire priority is decreased according to the importance of the norm. If there is not any norm related to the desires, its priority is not modified. For example, considering that a "Rescue Entity" has the intention to fulfil Norm 1, the desire "Evacuating the stranded workers to a safe location" is annotated with priority equal to 1.**(Plans Verification)** In this case, if the state described by an obligation norm is equal to one of the states achieved by the plan, the norm increases the priority of such plan and if the state described by a prohibition norm is equal to one of the states of the plan, the norm decreases the priority of such plan. For example, considering that a "Rescue Entity" has intention to fulfil the Norm 3, the priority of plans that uses helicopters are decreased of -1.

5.6 Selecting Desires (SD)

The SD function is responsible for selecting the desires with highest priority. By applying this function to our example, the goal "Evacuating the stranded workers to a safe

location" is selected because such goal has highest priority since it receives a positive influence of Norm 1.

5.7 Selecting Plans (SP)

The SP function is the one responsible for selecting the plan with highest priority. Let's consider that the desires of the agents in our example with highest priority is "Evacuating the stranded workers to a safe location", that the agent has the intention to fulfil Norm 3, and that the priority of plans that uses helicopters has decrease due to the execution of ADP function. When SP function is executed it selects the plan with highest priority that tries to rescue the NGO workers and that will not use helicopters to do so.

6 Related Work

We have summarized the related work in three groups: **(Norms Specification and Implementation)** Works in this group, such as [4] and [12], focus on the specification and operationalization of the norms and not on the implementation of the normative agents. Such works typically contribute with the formalization of norms and with engines that specify and explicitly manage the states of a norm. **(BDI Agents Implementation)** Although there are several of approaches to build BDI agents, for example [1] and [5], none provides support to implement normative agents. **(Normative Agents Implementation)** In this case, there are works, such as: (i) [6] that proposes an architecture to build norm-driven agents whose main purpose is the fulfilment of norms and not the achievement of their goals. In contrast, our agents are desire-driven entities that take into account the norms but are not driven by them; (ii) [3] presents concepts, and their relations, that are necessary for modelling autonomous agents in an environment that is governed by some (social) norms. Although such approach considers that the selection of desires and plans should be based on their priorities and that such priorities can be influenced by norms, it does not present a complete strategy with a set of verification in the norm review process, and strategies to evaluate, identify and solve conflicts between norms such as our work does; and (iii) [8] where the authors provide a technique to extend BDI agent languages by enabling them to enact behaviour modification at runtime in response to newly accepted norms. However, they do not answer the following questions: Why does an agent adopt a norm? How does an agent evaluate the positive and negative effects of these norms on its desires? and How does an agent detect and solve conflicts between the norms?

7 Conclusion and Future Work

This paper proposes an implementation to the Jason platform to build goal-oriented agents that can reason about norms. The implementation helps agents *(i)* on checking if they should adopt or not the norm; *(ii)* on evaluating the effects of the fulfilment or violation of the norm on their desires/intentions; *(iii)* on identifying and solving conflicts among norms selected to be fulfilled and among the ones selected to be violated; and *(iv)* on selecting desires and plans according to their choice of fulfilling or not a norm.

The applicability of such implementation can be verified by using the scenario presented in Section 4, where agents are responsible to plan the evacuation of people that are in hazardous location. The agents, built according to the proposed implementation, were able to reasoning about the norms they would like to fulfil, to solve conflicts among those norms, and to select plans following their intention of fulfilling or violating the norms. We are in the process of defining an experimental study in order complete the evaluation of our approach. It is also our aim to study others BDI architectures and platforms with the aim to investigate the possibility to extend then to build BDI normative agents.

References

1. Bordini, R., Hubner, J., Wooldridge, M.: Programming Multi-agent Systems in AgentSpeak Using Jason. Wiley, Blackwell (2007)
2. Burnett, C., et al.: Agent support for mission planning under policy constraints. In: Proc. of 2nd Int. Technology Alliance (2008)
3. Dignum, F.: Autonomous agents and social norms. In: Proc. of the Workshop on Norms, Obligations and Conventions (1996)
4. García-Camino, A., Rodríguez-Aguilar, J., Sierra, C., Vasconcelos, W.: Norm-oriented programming of electronic institutions. In: AAMAS 2006 (2006)
5. Jadex home page, http://jade.tilab.com/
6. Kollingbaum, M.: Norm-Governed Practical Reasoning Agents. PhD thesis, University of Aberdeen (2005)
7. Lopez-Lopez, F.: Social Power and Norms: Impact on agent behavior. PhD thesis, University of Southampton (2003)
8. Meneguzzi, F., Luck, M.: Norm-based behaviour modification in bdi agents. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (2009)
9. Oren, N., Luck, M., Norman, T.: Argumentation for normative reasoning. In: Proc. Symp. Behaviour Regulation in Multi-Agent Systems, pp. 55–60 (2008)
10. Rao, A.S.: Agentspeak(l): Bdi agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS, vol. 1038. Springer, Heidelberg (1996)
11. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a bdi-architecture. In: Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (1991)
12. Silva, V.: From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. In: JAAMAS, pp. 113–155 (2008)

Improving Space Representation in Multiagent Learning via Tile Coding

Samuel J. Waskow and Ana L.C. Bazzan

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{sjwaskow,bazzan}@inf.ufrgs.br

Abstract. Reinforcement learning is an efficient, widely used machine learning technique that performs well in problems that are characterized by a small number of states and actions. This is rarely the case in multiagent learning problems. For the multiagent case, standard approaches may not be adequate. As an alternative, it is possible to use techniques that generalize the state space to allow agents to learn through the use of abstractions. Thus, the focus of this work is to combine multiagent learning with a generalization technique, namely *tile coding*. This kind of method is key in scenarios where agents have a high number of states to explore. In the scenarios used to test and validate this approach, our results indicate that the proposed representation outperforms the tabular one and is then an effective alternative.

1 Introduction

Reinforcement learning (RL) techniques are based on estimates of values for pairs state-action. These values may be represented as a table with one entry for each state or each state-action pair. This works well in single agent problems and/or when the number of state and actions is small. In [7] two drawbacks of this approach are discussed. First, a lot of memory is necessary to keep large tables. Second, a long time is required to fill such tables accurately. Authors then propose that generalization techniques may help in addressing this so-called *curse of dimensionality*.

In multiagent learning, this problem is even more severe. Representation of the states is a key factor that may limit the use of the standard reinforcement learning algorithms in problems that involve several agents that must somehow learn in a coordinated and/or joint way. Moreover, in continuous scenarios, estimation of the state value by means of tabular Q values may not be feasible.

Therefore, generalization methods are even more important in multiagent reinforcement learning (MARL) problems. Using this approach, the value to be approximated at time t, say V_t, is not represented as a table but as a parametrized functional form with parameters being represented as a vector $\vec{\theta}_t$. In this paper we use *tile coding* [8] for generalization in MARL. To the best of our knowledge, this is the first attempt to tackle the dimensionality problem in MARL by means of function approximation.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 153–162, 2010.

This method allows agents to deal with states never exactly experienced before, which occurs frequently in problems where the representation of states is continuous. In such problems, the only way to learn efficiently is to generalize from previously (similar, close) experienced states. This is the case in two domains we are closely interested, namely prey-predator and traffic control. In both the number of states is huge since they are continuous in nature. Due to space limitations we restrict ourselves here to the latter scenario, comparing our results with the standard tabular representation. For the former, see [9].

This paper is organized as follows: the next section presents the background and related works on multiagent learning, focusing on approaches that deal with state abstraction; section 3 introduces the model of state space representation; section 4 presents the scenario, experiments, results and a comparison with other MARL approaches; the last section discusses the conclusions and future work.

2 Background and Related Work

The standard reinforcement learning (RL) framework for a *single agent* can be modeled as a Markov decision process (MDP). At each time step, the agent perceives the environment current state s and selects an action a. The environment responds with a reward signal r and a new state s'. This is formalized as an experience tuple $\langle s, a, s', r \rangle$.

As a commonly used RL technique, Q-Learning (QL) is an algorithm that estimates state-action values, the Q-values, which are numerical estimators of the quality of a given pair of state and action. The $Q(s, a)$ values are updated as the agent acts in a given environment, using an update rule given in Equation 1, where α is the learning rate and γ is the discount for future rewards.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \Big[r + \gamma \, max_{a'}(s', a') - Q(s,a) \Big] \tag{1}$$

Regarding action selection, a simple, well-known rule that can be used is ε-greedy, which states that the greedy action is selected for a proportion $1 - \varepsilon$ of the trials, while with proportion ε a random action is selected.

In order to improve the convergence rate of QL, Q(λ) [10] has been proposed to integrate the TD(λ) with the QL update rule. The λ factor refers to the use of elegibility traces that is a temporary record of the occurrence of an event, such as visiting a state or selecting an action. More details on eligibility traces can be found in [7].

The tabular representation of the Q values, indexed by state and action, quickly becomes impractical for a large number of entries. This problem can be tackled by a variety of existing function approximation methods. However, we choose the *tile coding* method because, being a linear method, it has a good convergence behavior.

Tile coding is a linear method indicated for practical applications of RL where the sets of states and actions tend to be huge or infinite. This technique generalizes the state space into partitions called tilings, each composed of sub-partitions called tiles. Thus, this method applies a mapping function $\phi(s)$ that transforms

the state s in a column vector of n binary features $[\phi_1, ..., \phi_n(s)]^T$ (same number of components as $\vec{\theta}_t$).

At each time step, the value V is obtained by $\phi(s) \times \vec{\theta}$, where $\vec{\theta}$ is a vector of n parameters associated with the action a at state s, and s is represented by the feature vector $\vec{\phi}(s)$. To approximate the function that determines the value of a state, an overlapping of tilings is used.

In a *multiagent* scenario, the problem of tabular representation is not different. Rather, it is even more severe. Here, learning usually means dealing with large, often continuous state and action spaces and this hinges on effective function approximation [6]. To see why this is the case, consider the tuple (J, S, A, R, T) that formalizes a multiagent learning problem via a multiagent Markov decision process (MMDP) i.e. a generalization of a MDP for $|J|$ agents. In this tuple:

$J = 1, ..., i, ...|J|$ is the set of agents
S is the discrete state space
$A = \times A^i$ is the discrete action space (set of joint actions)
R^i is the reward function (R determines the payoff for agent i as $r^i : S \times A^1 \times \ldots \times A^{|J|} \rightarrow \Re$)
T is the transition probability map (set of probability distributions over the state space S).

If all agents keep mappings of their joint states and actions, this would imply that each agent i needs to maintain tables whose sizes are exponential in the number of agents: $|S^1| \times ... \times |S^{|J|}| \times |A^1| \times ... \times |A^{|J|}|$. Consider $|J| = 2$ agents with $|S| = 10$ states and $|A| = 4$ actions each. The size of the Q table to represent the *joint* space of state and actions would be: $10^2 \times 10^2 \times 4^2 \times 4^2 = 2.560.000$. And this is, by no means, a real-world problem.

One solution is to let agents ignore others, i.e. agents do not consider joint states and joint actions. In fact, Claus and Boutilier [2] use an independent learner (IL) i.e. an agent that ignores the existence of other agents, and compare to what they call joint action learners (JAL), those that consider all observations and actions from every agent in the environment. These JAL's have a huge state representation, since they model all actions from all agents. However, neither IL nor JAL solves the problem. The former is known to converge to sub-optimal solutions while the latter cannot cope with real, big problems.

There has been some propositions in the literature to address such drawbacks. Due to lack of space we refer here just to sparse multiagent QL [4], a reinforcement-learning technique that models context-specific coordination requirements created for cooperative multiagent scenarios. The basic idea is to label each state of the system either as a coordinated or an uncoordinated state. This approach uses coordination graphs [3] thus it is required that the whole coordination mechanism be explicitly pre-defined.

We then see that there is a need for multiagent reinforcement learning (MARL) approaches that are robust enough to deal with problems of the real-world such as traffic control. In this particular domain, [1] presents a survey that shows that multiagent learning has not been solved to a satisfactory level. [5] presents

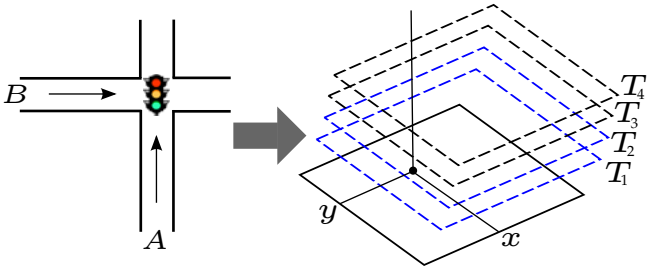


Fig. 1. Example of a mapping of features in a UTC scenario

a study of cooperative reinforcement learning applied on the traffic control scenario and analyzes how traffic-light agents can benefit from sharing information. However, their method still uses a tabular representation, which requires that states are discretized.

3 Multiagent Learning with Function Approximation

The MARL method proposed in this paper departs from the mapping state-action to value, which uses a traditional tabular mapping, such as QL. Rather, it uses a function approximation (here tile coding) and is based on the algorithm Watkins's $Q(\lambda)$. Given a $Q(s, a)$ value, the method retrieves features $\phi(s)$ and updates weights $\vec{\theta}$. For every action a_i in state s, there is a feature vector $\vec{\phi}(s)$ that represents the features present in state s. For each feature ϕ_i there is a tiling vector $\vec{T}$ that represents the set of layers of T_n tilings that are overlapped. For each tiling T_i, there is a tile vector $\vec{t}$.

The representation of state space is done by abstracting agents' interactions and is particular of each domain. For sake of illustration, Figure 1 depicts the state space mapping of a single agent (a traffic light). Assuming that this agent controls the junction between streets A and B, the state of the queue in street A was mapped into feature ϕ_x and the queue in street B was mapped into feature ϕ_y. Thus, the agent computes the $Q(s, a)$ value of a given state by means of the features ϕ_x and ϕ_y. These features are determined, in the case of this figure, by the overlapping of 4 tilings (T_1 to T_4).

4 Experiments and Results

4.1 Basics on Traffic Control via Traffic Lights

In this paper we use one of the most common way to control traffic, namely the use traffic lights at street junctions. Traffic lights work by creating or selecting signal plans for given traffic patterns. Therefore, traffic agents must learn to select an action (here a signal plan) according to the state of the lanes (here we consider the queues). A signal plan splits a cycle time (length of the plan)

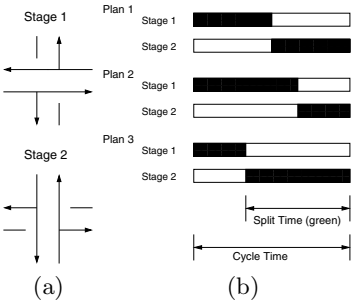


Fig. 2. Stages movement (a) and basic signal plans specification (b)

among all non-conflicting stages (traffic movements). In Figure 2b there are 3 different signal plans for the 2 possible stages (Figure 2a).

4.2 Basic Modeling as an MMDP

Modeling the urban traffic control problem as an MMDP is essential for solving it using reinforcement learning mechanisms. Below we present its main characteristics:

- J is the set of agents (junctions in charge of a traffic light)
- S is the set of states. This is a typical problem where the space of states is continuous: one usually wants to associate the state of an agent with the length of the queues. Moreover, such queues have to be computed over a given time frame because, since the traffic light imposes delays in the flow of vehicles, it makes no sense to associate state with instantaneous queue. Thus, for each traffic light, the state is associated with the average of stopped vehicles in each upstream link over 1 cycle of the signal plan, i.e. 60 time steps;
- A is the set of actions, which is discrete: actions are associated with signal plan as depicted in Figure 2b. Each agent selects a plan at each 120 time steps;
- the reward function assigns $R = -1$ if the queue occupies more than 95% of link capacity; $R = -0.01$ if the queue occupies between 30% and 95% of link capacity; and $R = 1$ if the queue occupies less than 30% of the link.

4.3 Scenarios

Experiments were conducted using ITSUMO, an open-source microscopic traffic simulator. Our scenario consist of two traffic networks depicted in Figure 3, where the edges represent the streets and the circles represent the traffic lights. The first scenario refers to the 3x3 grid depicted in the highlighted rectangle. The second scenario is similar but larger: the grid is 9x9. Both scenarios have an agent controlling each traffic light (A1 to I9 for the 9x9 grid).

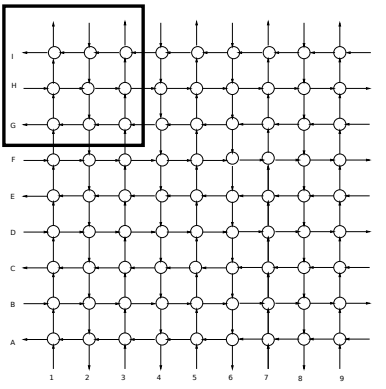


Fig. 3. Traffic network scenarios: 9x9 grid and 3x3 grid

Three kinds of experiments were performed, as detailed in Table 1. In experiments I and III the aim is to perform a comparison between possible values for the $|\vec{T}|$ (number of tilings) and $|\vec{t}|$ (number of tiles per tiling). This is an important analysis concerning the effect of different values for these parameters.

Experiments II and IV aim at comparing methods: IL and JAL (as defined by [2]) and our method that uses tile coding for function approximation.

Finally, experiment V aim to testing the performance of tile coding and IL in an episodic situation in which the rate of vehicles in both directions (horizontal and vertical) change along time.

All traffic lights in Experiments I, II, III, and IV have 2 signal plans (two actions). In the Experiment V, all traffic lights have 3 signal plans (3 actions).

More specifically, the learning algorithms used in experiments were: QL and $Q(\lambda)$. QL was applied using IL and JAL flavors. The learning parameters used in all experiments were learning step-size $\alpha = 0.5$ for QL and $\alpha = 0.01$ for $Q(\lambda)$, the reward factor $\gamma = 0.9$, the backup trace-size was $\lambda = 0.9$, and $\varepsilon = 0.05$ for the action selection rule (ε-greedy). These values were chosen both based on our experimentation and according to observations made in [7] (e.g. about the necessity and advantages of using a low value for α in $Q(\lambda)$).

Table 1. Description of experiments

Experiments	Methods	Number of agents	$\|\vec{T}\|$	$\|\vec{t}\|$	Horizontal ins. rate	Vertical ins. rate
I	Tile coding	9	1,5,9	1,5,9	1008	1008
II	Tile coding, IL and JAL	9	5	1	1008	1008
III	Tile coding	81	1,5,9	1,5,9	1008	25
IV	Tile coding and IL	81	9	9	1008	25
V	Tile coding and IL	81	9	9	1008 1008 25	25 1008 1008

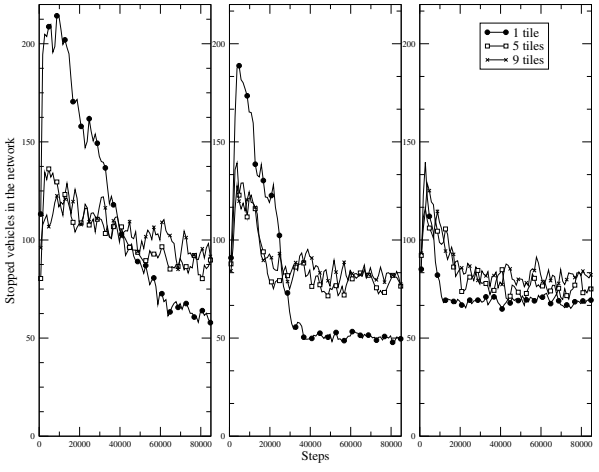


Fig. 4. Experiment I - 3x3 grid - (a): 1 tiling (b): 5 tilings (c): 9 tilings

As mentioned, Experiment I is a calibration of *tile coding* parameters, here for the smaller grid. Figure 4a shows that the number of stopped vehicles does not stabilize when $|\vec{T}| = 1$. Rather, as demonstrated in Figure 4b, the best result was achieved when $|\vec{T}| = 5$ and $|\vec{t}| = 1$. Figure 4c ($|\vec{T}| = 9$) indicates over-fitting because even using a high number of partitions the number of stopped vehicles was greater than the values showed in Figure 4b.

Experiment II still refers to the 3x3 grid. Here we have applied three learning methods (IL, JAL and tile coding) using $|\vec{T}| = 5$ and $\vec{t}| = 1$. As depicted in Figure 5, IL yielded the worst result because it could not represent the state space in an efficiently way.

Intuition says that the JAL method was supposed to bring the best result; however its large action space ($2^9 = 512$ i.e 2 actions for each agent) was not been fully searched during the simulated horizon. Because of this drawback, the tile coding based method showed the best result in this scenario. This is an indication that its representation method can cope with the variation in the queue values on the lanes. Next we discuss whether this holds for the 9x9 network.

Results of Experiment III are depicted as the three inner graphs in the Figure 6 and show the calibration of the tile coding parameters for the larger network. It is possible to see that it is necessary to use more tilings and more tiles than it was the case in the smaller scenario. Now $|\vec{T}| = 9$ and $|\vec{t}| = 9$ are the values that provide the best performance, meaning that the discretization in terms of tilings and tiles needs to be finer.

The outer graph on Figure 6 depicts Experiment IV where IL and tile coding were used with $|\vec{T}| = 9$ and $|\vec{t}| = 9$. It was not possible to use JAL due the combinatorial explosion caused by the size of joint states and actions in this larger scenario. Using IL results in a high number of stopped vehicles because

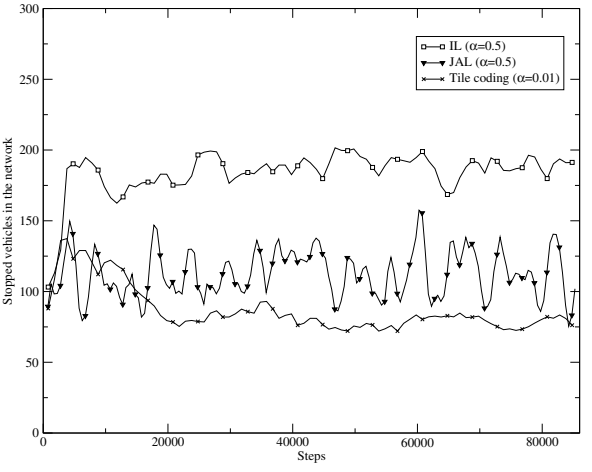


Fig. 5. Experiment II - 3x3 grid: comparison of methods

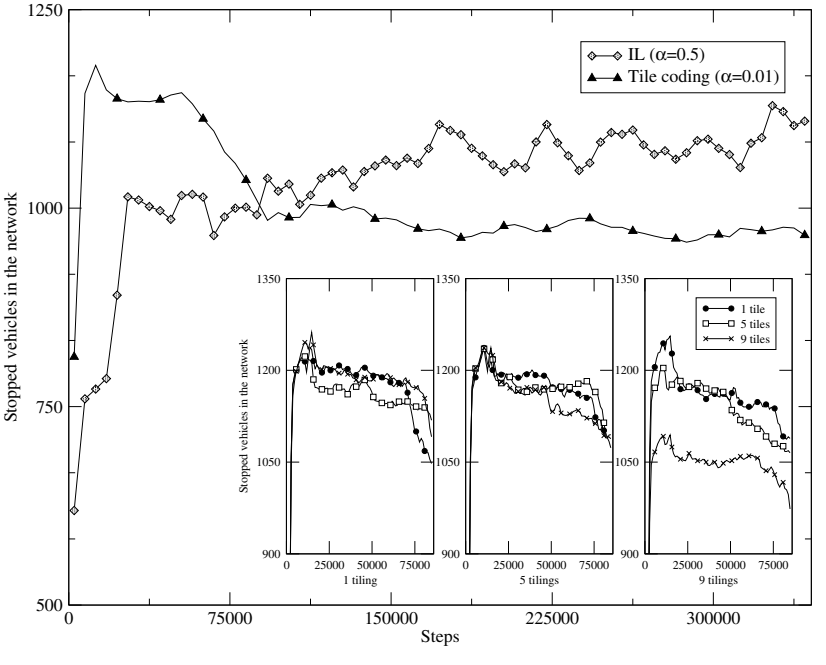


Fig. 6. Experiment III and IV - 9x9 Grid: calibration and comparison of methods

the tabular representation does not map all interactions in a satisfactory way. Frequently, agents take the same action but receive different rewards because the states are not represented correctly.

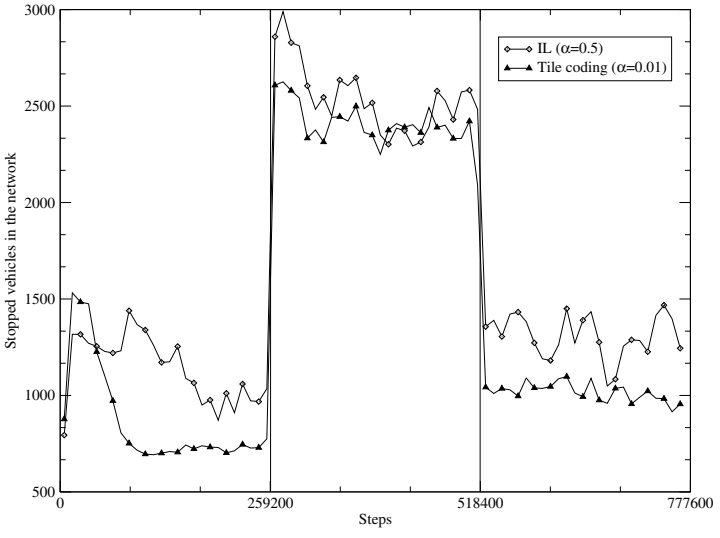


Fig. 7. Experiment V - 9x9 Grid: episodic case

It is possible to notice that the number of stopped vehicles increased suddenly around time step 25000 when using IL. This occurred because the queue reached the saturation point in some lanes. When using the tile coding model, we notice that at the beginning of the simulation, the number of stopped vehicles is high but decreases along the simulation. The explanation is that initially the large space ($|\vec{T}| = 9$ and $|\vec{t}| = 9$) has to be searched.

Figure 7 depicts Experiment V, where different vehicles insertion rates were used in each episode (see last 3 lines of Table 1). The plot shows that using tile coding also leads to better results than using IL, especially in the first and third episodes. In the second episode, the nearly equivalent performance can be explained by the insertion rate that is too high and quickly saturates all streets. This poses difficulties for the method to extract the necessary information for the states representation. This basically means that there is nothing to learn. We are working on a different configuration in which the insertion rate is not so high so that the selectiveness of all methods shall improve.

5 Conclusion

In this paper we make the point that the choice of features that compose a state is one of the most important requirements for efficient learning in complex multi-agent domains. The main contribution of this paper is the use of a representation method for RL (tile coding) that allows generalization when the space of states is continuous and/or high, which is the case in the traffic light control scenario that we have used to illustrate the use of tile coding in MARL.

The results indicated that the proposed representation outperforms the tabular representation, even when tabular one of *joint* actions is possible (as in Experiment II).

As future work, we intend to automate the choice of values for the tile coding parameters according to the agent perception, as well as to let agents modify values during the learning process. This has been proposed in [11] (for a mono agent case) and may bring even higher gain because the adaptive tile coding could cope with dynamic scenarios.

References

[1] Bazzan, A.L.C.: Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents and Multiagent Systems 18(3), 342–375 (2009)
[2] Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence, pp. 746–752 (1998)
[3] Guestrin, C., Lagoudakis, M.G., Parr, R.: Coordinated reinforcement learning. In: Proceedings of the Nineteenth International Conference on Machine Learning (ICML), pp. 227–234. Morgan Kaufmann, San Francisco (2002)
[4] Kok, J., Vlassis, N.: Collaborative multiagent reinforcement learning by payoff propagation. Journal of Machine Learning Research 7, 1789–1828 (2006)
[5] Oliveira, D.d., Bazzan, A.L.C.: Multiagent learning on traffic lights control: effects of using shared information. In: Bazzan, A.L.C., Klügl, F. (eds.) Multi-Agent Systems for Traffic and Transportation, pp. 307–321. IGI Global, Hershey (2009)
[6] Sherstov, A.A., Stone, P.: Improving action selection in MDP's via knowledge transfer. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (July 2005)
[7] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
[8] Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coding. In: Touretzky, D., Mozer, M., Hasselmo, M. (eds.) Advances in Neural Information Processing Systems, vol. 8, pp. 1038–1044. MIT Press, Cambridge (1996)
[9] Waskow, S.J., Bazzan, A.L.C.: Reinforcement learning methods: Generalizing joint tasks. In: Proceedings of the 35th Latin-American Informatics Conference, CLEI, Pelotas, Brazil (September 2009)
[10] Watkins, C.: Learning from Delayed Rewards. PhD thesis, University of Cambridge (1989)
[11] Whiteson, S., Taylor, M.E., Stone, P.: Adaptive tile coding for value function approximation. Technical Report AI-TR-07-339, University of Texas at Austin (2007)

Factored Translation between Brazilian Portuguese and English

Helena de Medeiros Caseli and Israel Aono Nunes

Department of Computer Science
Federal University of São Carlos (UFSCar)
Rod. Washington Luís, Km 235, CP 676
13565-905 São Carlos-SP
helenacaseli@dc.ufscar.br, israelaono@gmail.com

Abstract. Factored translation is an extension of the state-of-the-art phrase-based statistical machine translation (PB-SMT). The main difference in factored translation approach is that a word is not only a token (its surface form) but a vector composed of different information such as lemma, part-of-speech or morphologic/syntactic tags. In this paper we present some experiments carried out to train and test factored translation models on Brazilian Portuguese and English texts. Using part-of-speech and morphological information, the factored models showed better results than the baseline (a PB-SMT), but the same gain in performance was not reached when flat syntactic tags were considered.

1 Introduction

Machine Translation (MT) is one of the oldest and most important areas of Natural Language Processing (NLP). Since its beginnings several methods and paradigms have been proposed ranging from the basic level — in which MT is performed by just replacing words in a source language by words in a target language — to more sophisticated ones — which rely on manually created translation rules (Rule-based Machine Translation or RBMT) or automatically generated statistical models (Statistical Machine Translation or SMT).

In the last years we have experienced the revival of the statistical approaches for MT. Nowadays, according to the automatic evaluation measures BLEU [1] and NIST [2], the phrase-based SMT systems (PB-SMT) as [3,4] are considered the state-of-the-art for MT.

The defenders of SMT point out a lot of advantages of this approach compared with others. First of all, SMT can be applied to perhaps almost any language pair and corpus type. Furthermore, it is an inexpensive, easy and language independent way for detecting recurrent phrases that form the language and translation models. However, the traditional PB-SMT approach also has some disadvantages such as their limitation to map small text chunks without any explicit use of linguistic information (morphological, syntactic or semantic) [5]. For example, in a SMT model, related words such as "write", "writes" and "written" are treated as completely different ones. In this case, if the

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 163–172, 2010.

SMT model has been trained to translate only the word "write" it would not be able to generalize and also translate the word "written". This problem becomes bigger in morphologically rich languages such as the Brazilian Portuguese.

Aiming at to overcome this limitations, [5] proposed an extension for the traditional PB-SMT in which a word is not represented only by a token (its surface form) but a vector of *factors* one for each different level of annotation: surface form, lemma, part-of-speech, morphological features, syntactic tags and go on. This new statistical approach for MT is called *factored translation* and has shown promising results as the ones presented in this paper regarding the factored translation from Brazilian Portuguese to English and vice-versa.

Thus, this paper briefly explains the factored translation approach (section 2) and some related work carried out with other language pairs since, to the best of our knowledge, this paper presents the first experiments on factored translation with Brazilian Portuguese (section 3). Then, we present the results of the experiments developed with Brazilian Portuguese and English parallel texts to train and test some factored translation models (section 4). Finally, we finish this paper with some discussion and proposals for future work (section 5).

2 Factored Translation

Factored translation, as stated by [5], is an extension of PB-SMT models since the main difference between them lies in: (i) the way training data is preprocessed and (ii) the learned models. In PB-SMT, the source and target words in the translation examples (parallel sentences) are grouped into phrases, that is, in sequences of two or more consecutive words not necessarily forming a syntactic phrase. For instance, Figure 1[1] brings an example of how a phrase-based model performs the translation. In this example it is possible to see that the input sentence is segmented into phrases that are, then, translated into an English phrases (such as "*of course*" or "*fun with the*") which may be reordered.

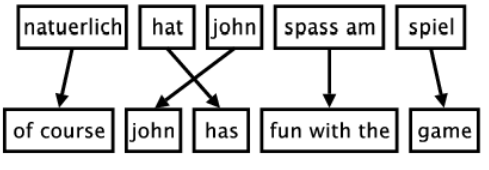


Fig. 1. Example of a PB-SMT process

Similarly to the traditional PB-SMT systems, factored models are built by means of a training process performed based on a sentence-aligned parallel corpus were words are grouped into phrases as shown in the previous example. However, to be able to perform the factored training, the parallel sentences have to be

[1] http://www.statmt.org/moses/?n=Moses.Background

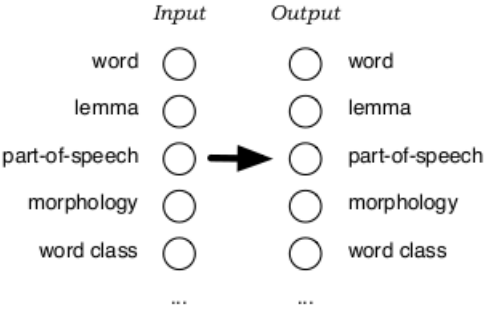


Fig. 2. Example of input and output words with additional factors besides the surface forms (word) [5]

enriched with extra factors in addition to the surface forms presented in Figure 1. Usually, this additional factors are obtained by running automatic tools such as part-of-speech taggers and syntactic parsers and it is possible to use as many factors as needed according to the designed experiments.

As in traditional PB-SMT models, the training of factored models involves the word alignment —typically performed by GIZA++[2] in both translation directions (source–target and target–source) and, then, combined by means of a symmetrization process [6]— of the parallel sentences and the extraction of all phrase-pairs that are consistent with this word alignment. Figure 2 shows an example of additional factors used in factored translation models.

Different from PB-SMT, in the factored models, the phrase mappings are extracted for each factor. These phrase mappings are, then, scored based on relative counts and word-based translation probabilities [5]. So, the translation of the input factored representation to the output factored representation is broken into a sequence of mapping steps that either translate input factors in output factors or generate additional output factors form already existing ones.

It is important to say that while the translation steps are performed considering the phrase level, the generation steps are done for each word. Furthermore, each step is modeled by one or more feature functions which are combined by means of a log-linear model.

3 Related Work

In [5], the authors used the open-source toolkit Moses[3] [7] to train and evaluate several factored translation models. Moses has components for: (i) preprocessing the training data, (ii) training language and translation models, (iii) optimizing these models using the minimum error rate (MER) training proposed by [8] and (iv) evaluating the automatically generated translations using metrics such as BLEU [1] and NIST [2].

[2] http://www.fjoch.com/GIZA++.html
[3] http://www.statmt.org/moses/

These authors carried out several experiments using the factored translation model framework based on linguistic information and also automatically generated word classes for English–German, English–Spanish, English–Czech and English–Chinese language pairs. The sizes of the parallel corpora used for training the factored translation models varied from 20,000 to more than 750,000 sentences and came from law and journalistic domains. In spite of so different corpora, the factored translation models outperformed the baseline PB-SMT trained based only on the surface forms in all of the experiments. The gain over the traditional PB-SMT was reported in terms of BLEU [1] (up to 2% BLEU) and also as a measure of grammatical coherence.

In [9] factored SMT was reported also with English–Czech language pair, but using a different corpus two times bigger than the one used by [5]. The results reported by this author also show that the factored translation models outperforms the traditional PB-SMT. A similar gain is reported in [10] using a corpus even bigger.

Following the ideas on these related works, in this paper we report experiments carried out to test factored translation between Brazilian Portuguese and English. To the best of our knowledge this is the first time that such approach is applied to this language pair.

4 Experiments and Results

The experiments described in this paper were carried out using a parallel Brazilian Portuguese and English corpora (section 4.1) and some standard parameters (section 4.2). The factored models trained according to different configurations are presented in the last three sections (4.3, 4.4 and 4.5). Furthermore, the SMT models were evaluated based on the automatic measures BLEU [1] and NIST [2] and the statistical significance was measured using bootstrapping with 99% of confidence and 10,000 samples [11].

4.1 Training and Test Parallel Corpora

The experiments described in this paper were carried out using a training parallel corpus composed of articles from the online version of the Brazilian scientific magazine *Pesquisa Fapesp*[4] written in Brazilian Portuguese (the original version) and English (the translated version). This corpus has 17,397 sentence pairs and more than 1 million tokens (494,391 in Portuguese and 532,121 in English).

The parallel sentences in this training corpus were preprocessed as explained below. First, they were PoS-tagged using the morphological analyser and the PoS tagger available in the Apertium open-source machine translation platform.[5] The morphological analysis provides one or more lexical forms or analyses

[4] http://revistapesquisa.fapesp.br/

[5] The open-source machine translation platform Apertium, including linguistic data for several language pairs and documentation, is available at http://www.apertium.org

English	Brazilian Portuguese
the|the|det|det.def.sp faults|fault|n|n.pl of|of|pr|pr the|the|det|det.def.sp spheres|sphere|n|n.pl	os|o|det|det.def.m.pl defeitos|defeito|n|n.m.pl das|de+o|pr+det|pr+det.def.f.pl esferas|esfera|n|n.f.pl

Fig. 3. English and Brazilian Portuguese parallel sentences with 4 factors to train the factored models

(information on lemma, lexical category and morphological inflection) for each surface form using a monolingual morphological dictionary.[6] The PoS tagger chooses the best possible analysis based on a first-order hidden Markov model (HMM) [13].

The test corpus consists of 667 parallel sentences from the same scientific magazine but different from those sentences in the training corpus. The reference corpus used to evaluate the output translations by means of automatic measures (such as BLEU [1] and NIST [2]) was created from the corresponding parallel sentences in the test corpus. For example, when translating from Portuguese to English, the Portuguese sentences in the test corpus were used as the input test sentences for the trained models and the output sentences were compared with the English ones in this test/reference corpus.

Figure 3 brings parallel sentences from the training corpus. For each token, the following factors are separated by "|" and enumerated from 0 to 3 in this order: surface form (#0), lemma (#1), part of speech (#2) and part of speech plus morphological information (#3).

4.2 Standard Training Configuration

All the experiments described in this paper were carried out using the Moses' training standard configuration and additional parameters as follows:

`-giza-option m1=5,m2=0,mh=5,m3=3,m4=3` = GIZA++'s alignment is performed with 5 iterations of model IBM1, 3 of IBM3 and IBM4 [14] and 5 iterations of HMM [15],
`-alignment grow-diag-final-and` = alignment option,
`-max-phrase-length 7` = maximum phrase length defined as 7 tokens,
`-reordering msd-bidirectional-fe` = reordering model definition.

These parameters were added to the Moses' standard configuration to train a baseline PB-SMT model. The baseline model was trained based only on the surface forms in the training corpus, that is, considering just the first factor (#0) in Figure 3. BLEU [1] and NIST [2] values achieved by the baseline model on the test corpus (see section 4.1) are shown in Table 1.

[6] To improve the coverage of the morphological analyser, the original morphological dictionaries were enlarged as explained in [12].

Table 1. BLEU and NIST values for the baseline PB-SMT model trained based only on surface forms (#0)

	Portuguese–English		**English–Portuguese**	
	BLEU	**NIST**	**BLEU**	**NIST**
baseline	0.3903	8.3008	0.3589	7.8312

Although higher performance have already been reached in PB-SMT experiments with European Portuguese and English texts, as described in [16],[7], we trained our own PB-SMT: (i) to have a state-of-the-art PB-SMT model trained for Brazilian Portuguese and English language pairand (ii) to use it as a baseline for comparison with the factored models.

4.3 Factored Model Based on Part-of-Speech and Morphology

Our first hypothesis was to investigate the impact of an additional language model in the translation performance. To do so, we tested language models generated only from part-of-speech (the factor #2 in Figure 3) and also part-of-speech+morphological information (the factor #3 in Figure 3).

Additional Moses' parameters were used to train this kind of factored model. For instance, we trained a language model for factor #3 with the maximum phrase length defined as 7 tokens (`factor3.lm`) and used the following additional parameters to train the factored translation model:

`--lm 3:7:/work/lm/factor3.lm:0` = the additional language model being used is the one generated for factor #3 (factor3.lm),

`--translation-factors 0-0,3` = the translation is, first, performed using source surface forms (#0); then, the target surface forms (#0) are mapped into target part-of-speech+morphological information (#3).

Table 2 shows the BLEU [1] and NIST [2] values for the factored translation with additional language models generated from factors #2 (+LM#2) and #3 (+LM#3).

Table 2. BLEU and NIST values for the factored translation with additional language models from factors #2 (+LM#2) and #3 (+LM#3)

	Portuguese–English		**English–Portuguese**	
	BLEU	**NIST**	**BLEU**	**NIST**
baseline	0.3903	8.3008	0.3589	7.8312
+LM#2	0.3917	8.3643	0.3542	7.7500
+LM#3	**0.3923**	**8.3926**	**0.3703**	**7.9699**

[7] In [16], the experiments were carried out using a parallel corpus more than 100 times bigger than the one used in this paper, for legal texts written in European Portuguese and English, with BLEU values of 0.60 for Portuguese–English and 0.55 for English–Portuguese.

According to Table 2, the additional language model generated only from the part-of-speech factor (+LM#2) brought some improvement in the translation from Brazilian Portuguese to English but not in the other translation direction. Indeed, the difference between the baseline and the +LM#2 was not statistically significant in any translation direction, according to bootstrapping (with 99% of confidence and 10,000 samples) [11]. This fact indicates that the part-of-speech by itself does not have impact in the selection of the best translation.

On the other hand, the additional language model trained from part-of-speech+morphological information (+LM#3) brought a statistically significant improvement in the English–Portuguese translation according to bootstrapping (with 99% of confidence and 10,000 samples) [11]. What is important in this result is that the morphological information indeed helped the MT system to translate into a morphologically richer target language (Brazilian Portuguese).

4.4 Factored Model Based on Decomposition of the Translation

The next factored model evaluated here was defined based on [17] and their statement that the translation process becomes more robust when it is decomposed in morphological analysis and generation. To investigate this statement we divided the translation process in the following steps:

`--translation-factors 1-1+3-3+0-0,3` = the translation is performed in:
1. one step for the translation between lemmas (1-1),
2. one step for the translation between part-of-speech+morphological (3-3),
3. one final step for the translation between surface forms and part-of-speech+morphological (0-0,3).

`--generation-factors 1-3+1,3-0` = the generation is performed as follows:
1. one step sets possible part-of-speech+morphological to lemmas (1-3),
2. another step maps part-of-speech+morphological and lemma to surface forms (1,3-0).

`--decoding-steps t0,g0,t1,g1:t2` = three translation steps (t0, t1 and t2) and two generation steps (g0 and g1) are performed. An extra translation table is created (0-0,3) and applied as a different decodification step (:t2).

Following this configuration, the values of BLEU [1] and NIST [2] were better than the ones achieved in the experiments described so far, as shown in Table-3. However, this improvement does not prove to be statistically significant according to bootstrapping (with 99% of confidence and 10,000 samples).

4.5 Factored Model Based on Parsing Factors

Finally, we also carried out experiments using parsing factors for training factored translation models. To do so, we mapped the structural information present in the parse trees, as presented in Figure 4, to flat syntactic tags as shown in Figure 5: surface form (#0), part of speech (#1) and syntactic information (#2). By doing so, we lost the rich structural information present in the parse

Table 3. BLEU and NIST values for the factored translation with decomposition (+LM#3DE)

	Portuguese–English		**English–Portuguese**	
	BLEU	**NIST**	**BLEU**	**NIST**
baseline	0.3903	8.3008	0.3589	7.8312
+LM#3	0.3923	8.3926	0.3703	7.9699
+LM#3DE	**0.3932**	**8.4421**	**0.3713**	**7.9813**

trees. For example, the prepositional phrase *of the spheres* is not taken into account since there is another phrase, the NP *the spheres*, inside of it. We decided to not represent both of them to avoid overspecialization, undesirable in machine learning techniques.

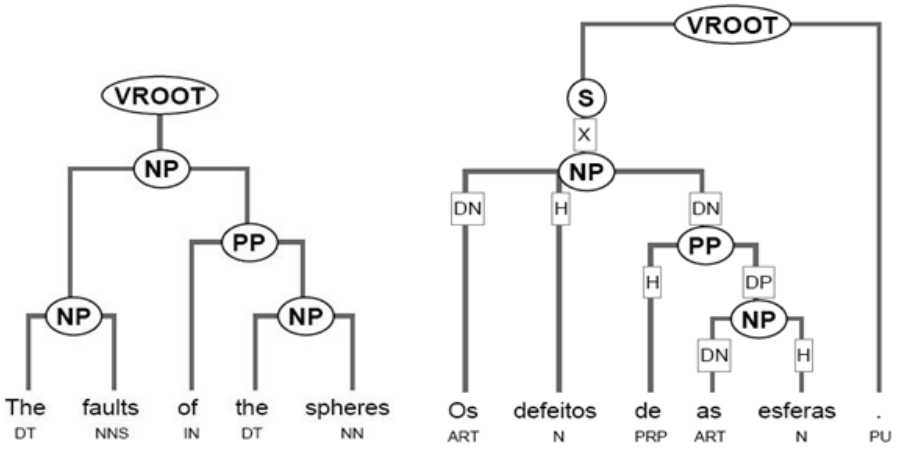


Fig. 4. Example of English and Brazilian Portuguese parse trees

English	**Brazilian Portuguese**
the|dt|np faults|nns|np of|in|pp the|dt|np spheres|nn|np	os|art|np defeitos|n|np de|prp|pp as|art|np esferas|n|np

Fig. 5. English and Brazilian Portuguese parallel sentences with parsing factors

When translating from Brazilian Portuguese to English, the syntactic tags used in this flat way lead to a performance similar to the baseline. However, in the translation from English to Portuguese, the factored model based on syntactic tags decreased the baseline's performance in 4% BLEU and 2.5% NIST. The main explanation for this fact is that the factored SMT does not take into account the structural information represented in the syntactic trees. To take advantage of the rich structural information in the parse trees, the syntax-based approaches for MT should be considered.

5 Conclusions and Future Work

As stated by [5], the traditional PB-SMT approach suffers from the limitation of mapping small text chunks without any explicit use of linguistic information (morphological, syntactic or semantic). By doing so, the traditional PB-SMT models are not able to deal with different forms of the same word. This problem becomes even bigger when morphologically rich languages such as the Brazilian Portuguese are involved.

Factored translation was proposed to try to overcome this limitation and has already proved its value in morphologically richer languages. In the experiments described in this paper, we concluded that the morphological information indeed helped the MT system to translate into a morphologically richer target language (Brazilian Portuguese).

However, if we intend to go further and use syntactic information in the training, the structural information present in the parse trees is not taken into account since they are reduced to flat tags. So, as future work and aiming at exploring the whole power of syntactic information we intend to investigate syntax-based SMT models such as those proposed in [18] and [19].

Acknowledgements

We would like to thank the financial support of the PIADRD (Programa Integrado de Apoio ao Docente Recém Doutor) PUIC/UFSCar and FAPESP for its support in building the parallel corpus.

References

1. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual meeting of the Association for Computational Linguistics (ACL 2002), pp. 311–318 (2002)
2. Doddington, G.: Automatic evaluation of machine translation quality using n-gram cooccurrence statistics. In: Proceedings of the Human Language Technology Conference (HLT 2002), pp. 128–132 (2002)
3. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Proceedings of the Human Language Technology (HLT/NAACL 2003), pp. 127–133 (2003)
4. Och, F.J., Ney, H.: The Alignment Template Approach to Statistical Machine Translation. Computational Linguistics 30(4), 417–449 (2004)
5. Koehn, P., Hoang, H.: Factored Translation Models. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Association for Computational Linguistics, Prague, pp. 868–876 (June 2007)
6. Och, F.J., Ney, H.: Improved statistical alignment models. In: Proceedings of the 38th Annual Meeting of the ACL (ACL 2000), Hong Kong, China, pp. 440–447 (2000)

7. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open Source Toolkit for Statistical Machine Translation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007), Prague, Czech Republic, pp. 177–180 (June 2007)
8. Och, F.J.: Minimum error rate training for statistical machine translation. In: Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL) (2003)
9. Bojar, O.: English-to-Czech Factored Machine Translation. In: Proceedings of the Second Workshop on Statistical Machine Translation, ACL, Prague, pp. 232–239 (June 2007)
10. Bojar, O., Hajič, J.: Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In: Proceedings of the Third Workshop on Statistical Machine Translation, ACL, Columbus, Ohio, USA, pp. 143–146 (June 2008)
11. Zhang, Y., Vogel, S., Waibel, A.: Interpreting Bleu/NIST scores: How much improvement do we need to have a better system? In: Proceedings of LREC 2004, Lisbon, Portugal (May 2004)
12. Caseli, H.M., Nunes, M.G.V., Forcada, M.L.: Automatic induction of bilingual resources from aligned parallel corpora: application to shallow-transfer machine translation. Machine Translation 20, 227–245 (2006)
13. Armentano-Oller, C., Carrasco, R.C., Corbí-Bellot, A.M., Forcada, M.L., Ginestí-Rosell, M., Ortiz-Rojas, S., Pérez-Ortiz, J.A., Ramírez-Sánchez, G., Sánchez-Martínez, F., Scalco, M.A.: Open-source Portuguese-Spanish machine translation. In: Vieira, R., Quaresma, P., Nunes, M.d.G.V., Mamede, N.J., Oliveira, C., Dias, M.C. (eds.) PROPOR 2006. LNCS (LNAI), vol. 3960, pp. 50–59. Springer, Heidelberg (2006)
14. Brown, P.F., Pietra, V.J., Pietra, S.A.D., Mercer, R.L.: The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics 19, 263–311 (1993)
15. Vogel, S., Ney, H., Tillmann, C.: HMM-based word alignment statistical translation. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996), pp. 836–841 (1996)
16. Koehn, P., Birch, A., Steinberger, R.: 462 Machine Translation Systems for Europe. In: Machine Translation Summit XII (2009)
17. Koehn, P., Federico, M., Shen, W., Bertoldi, N., Bojar, O., Callison-Burch, C., Cowan, B., Dyer, C., Hoang, H., Zens, R., Constantin, A., Moran, C., Herbst, E.: Open Source Toolkit for Statistical Machine Translation: Factored Translation Models and Confusion Network Decoding. Technical report, Johns Hopkins University – Center for Speech and Language Processing (September 2007)
18. Galley, M., Graehl, J., Knight, K., Marcu, D., Deneefe, S., Wang, W., Thayer, I.: Scalable inference and training of context-rich syntactic translation models. In: ACL, pp. 961–968 (2006)
19. Nguyen, T.P., Shimazu, A., Ho, T.B., Nguyen, M.L., Nguyen, V.V.: A tree-to-string phrase-based model for statistical machine translation. In: CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning, Manchester, England, Coling 2008 Organizing Committee, pp. 143–150 (August 2008)

Question Answering for Portuguese: How Much Is Needed?

Rodrigo Wilkens[1] and Aline Villavicencio[1,2]

[1] Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)
[2] Department of Computer Sciences, Bath University (UK)
{rodrigo.wilkens,avillavicencio}@inf.ufrgs.br

Abstract. The task of a Question Answering (QA) system is to automatically answer a question in natural language, searching for information in a given data source, such as structured databases or unstructured natural language documents. In this paper we investigate how much is needed in terms of tools and resources for a QA system for Brazilian Portuguese. In particular we assess the impact of shallow and deep methods and the contribution of different tools and resources in terms of the performance of the system in the task. We argue that a combination of deep and shallow strategies results in optimal performance, but that shallow methods alone may already give adequate results.

1 Introduction

The task of a Question Answering (QA) system is to automatically answer a question in natural language, searching for information in a given data source, such as structured databases or unstructured natural language documents (e.g. corpora from a given domain, newspaper texts). This is a challenging task as question types are varied and can include facts, lists, definitions, while answers may come from sources of information ranging from small document collections, to the World Wide Web, and these determine the level of processing that can be feasibly employed for analyzing the question and data sources. Moreover, the difficulty of the task is also inuenced by whether the questions are restricted to a particular domain (e.g. sports, genes) or not, which additional sources of information are available for a given language (e.g. ontology's of domain-specific knowledge, general ontology's), their coverage, and which tools can be used to help the task (e.g. named entity recognizers', parsers, word sense disambiguation tools). Such systems have the potential to make written information more easily accessible to wider audiences, through questions in natural language, especially if the interface allows voice communication.

The Question Answering process can be divided into four stages, as follows: (1) question analysis, (2) identification of candidate documents, (3) generation of candidate answers and (4) computation of answer score. These stages require the following tasks: (1) identification of the topic of the question; (2) translation of user question to a query for an information retrieval tool and search using

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 173–182, 2010.

tool; (3) identification of candidate answers through processing of the documents retrieved; and (4) ranking of the candidates answers in terms of decreasing order of relevance to question.

For resource rich languages like English many such systems have been developed involving different levels of processing, such as Javelin [1], QuALiM [2], TrueKnowledge[1], LAMP [3], Ephyra [4] and AnswerBus [5]. Javelin, for instance, presents a combination of modules for analysis, sources of information, speech synthesis and user responses. Likewise the amount of resources and prior knowledge differs from system to system. QuALiM explores the use of lexical resources such as FrameNet, and PropBank VerbNet in QA and considers two different complementary methods that use these resources. TrueKnowledge is a system based on knowledge, which uses a large database of facts about general topics. For inference about facts, a knowledge generator or an external knowledge supplier can be used. For example, LAMP has as distinctive feature the use of snippets returned by Google, while Ephyra (or OpenEphyra) is the first open framework for QA that retrieves accurate responses to questions in natural language using the Web. Some systems also allow multilingual QA, as is the case of AnswerBus which does open domain QA on using sentence level information retrieval.

The relevance of research in QA is demonstrated by the number of initiatives and conferences devoted to the task. For example, several conferences have had special tracks for evaluating Question Answering technology and systems, such as Text REtrieval Conference (TREC - http://trec.nist.gov) and Cross Language Evaluation Forum (CLEF - http://www.clef-campaign.org). In addition, the interest in the task is also reflected by the inclusion of languages other than English in these evaluations: since 2004 CLEF has included, among others, Portuguese as one of the languages accepted, both as the language of the questions and as the target language containing the answers, and evaluating both mono and multilingual systems. However, there still only a few systems for Portuguese, and even less for Brazilian Portuguese. Among the systems taking part in CLEF we can see a development in terms of performance, as they were evaluated on the same data, over the years. Examples of QA systems for Portuguese are FOX (REPOSA), Priberam and Sphinx (ESFINGE), discussed in section 2. However, there is no consensus as to the amount of resources and tools that are needed in order to build a working QA system with reasonable performance.

In this paper we analyze the contribution brought by different combinations of resources and tools and evaluate how much they affect the performance of each stage of a QA task. The remainder of this paper is structured as follows. In section 2 we discuss QA systems for Portuguese and do a comparison of their particular configurations and performances. Section 3 presents the resources and system configurations in terms of tools and resources used, while section 4 discuss the results obtained. Section 5 finishes with conclusions and future work.

[1] http://www.trueknowledge.com/

2 Question Answering Systems for Portuguese

In this section we discuss the main characteristics of some of the QA systems for Portuguese and compare their performances: University of Évora QA, RAPOSA(FOX), Priberam, Esfinge (Sphinx).

The approach of the University of Evora [6] is based on selecting for each question the most relevant documents, following an information retrieval task. The information contained in each document is analyzed and the query is processed over the knowledge base of each text for the system to return the first answer found. To create the knowledge base for each document, the parser PALAVRAS [7] was used, and the parsed sentences are rewritten using first-order logic. The system uses an ontology built on the basis of the first order logic expressions. For answer identification, the same processing is done to analyses both the question and the documents.

RAPOSA [8], [9] is an open domain QA system which adopts a shallow approach for answering biographical questions. Questions are divided into 3 groups: elementary, profile dependent and speculative questions. The first type refers to questions about common and general attributes (e.g. parenthood), the second depends on the particular person and requires world knowledge for a precise answer (e.g. *name one movie directed by Claude Lanzmann*), while the third may involve a false hypothesis (e.g. *when has X committed suicide?*). For a given question the system performs the following steps: (1) identification of named entities; (2) query generation from the question; (3) collecting of summaries where the answer may be found, (4) extraction of answer using a set of rules and a measure of the semantic similarity between the question and the possible answers, (5) merging of answers and (6) answer selection based on a confidence score of a summary given the question. During the analysis of the question the system identifies the type of question and reformulates it so that it only contains core words. The system then searches both a document collection and the Wikipedia and returns summaries of the texts. The extraction of the answer is done selecting the documents whose summaries have the same semantic type as that of the question, according to a set of rules [9]. In order to assess the effects of the semantic analysis of texts on a realistic application an evaluation of the system was performed using the CLEF 2007 dataset, and it resulted in around 20% incorrect answers. These errors may be partly explained as caused by the search in summaries and by the choice of summaries with a semantic permissible type.

The Priberam system uses TRUST (Text Retrieval Using Semantic Technologies) for information retrieval combined with deep linguistic analysis such as named entity recognition, part-of-speech (POS) tagging and context-free grammar for disambiguation, employing rich resources like ontologies and thesauri [10]. It follows 5 stages: (1) indexing, (2) question analysis, (3) document retrieval, (4) passage retrieval and (5) answer selection. The analysis of the question determines the type and topic of question, and these are used as basis for document retrieval. Passage retrieval is done on the basis of similarities between a passage and the question, prioritizing sentences with more complete information (e.g. *Fidel Castro* vs *Fidel*). To extract an answer, the system determines

the similarity between each candidate answer and the types identified during question analysis.

Esfinge [11,12] performs question reformulation and n-gram harvesting, filtering and composition. The question reformulation module identifies patterns based on the words in the question and rewrites the question to turn it into the expected form for the answer. The n-gram harvesting module uses the rewritten form obtained in the previous module to query the document repository for the highest scoring ngrams in the summaries. The n-gram filtering module uses POS information about n-grams for reducing the candidate set, and these are given to the n-gram composition module to try to answer the question, and it determines characteristics of the answer (e.g. whether it should be in the singular or plural).

2.1 Portuguese QA and CLEF

In 2004 2 systems took part in the CLEF evaluation: the University of Évora system and Esfinge (with and without access to the Web). The former had the best performance with 28,64% overall accuracy, where 29,17% was obtained for factoids and 25,81% over definitions, while Esfinge with Web access had the second best [13]. In 2005 Priberam also took part, and had the best performance (with 64,5% accuracy, where 67,41% is for factoids and 64,29% for definitions), while the University of Évora system obtained 26% accuracy, where 21,48% was for factoids and 35,71% for definitions [14]. There was an increase in the number of collections in 2006, which included the Portuguese newspapers Público and Folha de São Paulo. From the four participating systems (Esfinge, NILC, Priberam and Fox), Priberam had the best performance (with 67% accuracy) while Esfinge again came second (with 25%) [15]. In 2007 5 systems took part in CLEF: University of Évora, Esfinge, INESC, Priberam and Fox. Priberam had again the best overall performance (50%) but this time there was a smaller gap to the second best performance, University of Évora (42%) [16]. In 2008 the pattern was repeated, and among the 6 participants (the University of Évora, Esfinge, Fox, INESC, Priberam and the Universidade Aberta), Priberam obtained the best performance with the University of Évora system coming second [17]. Figure 1 summarize the performance of the Portuguese language in each of the years of evaluation by the QA@CLEF.

Given this scenario, although a common evaluation standard helps to compare these systems, their evolution through time, as well as the use of different datasets for CLEF do not allow straightforward conclusions to be drawn about the advantages of different system configurations and the use of particular resources and tools.

3 Materials and Methods

The goal of this investigation is to identify the impact of the processing stages, tools and resources employed in the performance of a QA system. In order to do that we developed a standard system with the most common modules used in

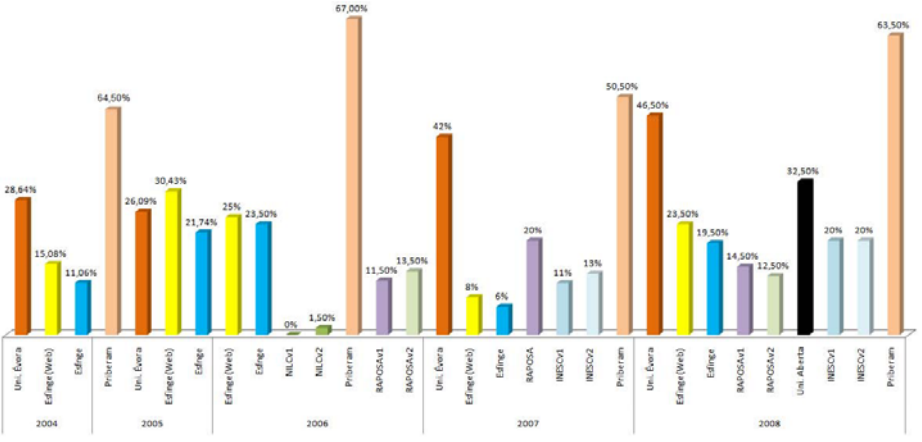


Fig. 1. Performance of QA systems for evaluations Portuguese QA@CLEF

different systems, and evaluated the performance of the different configurations over a single dataset, the OLinCom corpus.

3.1 The Corpus

The OLinCom corpus consists of a set of questions and documents prepared for the First Brazilian Olympiad on Computational Linguistics (OLinCom)[2]. This dataset contains 20 documents and 30 questions, which given its diversity allows the identification of classes of questions that are more difficult to answer for the different configurations, despite its small size in comparison to the corpora used in campaigns such as TREC and CLEF. The corpus contains 4936 tokens distributed in 267 sentences.

3.2 The QA Architecture

The standard QA architecture defined is shown in figure 2, and it has three modules: question analysis, document retrieval and passage retrieval. The system receives as input a question (used by the question analysis and document retrieval modules) and a corpus (used by the document retrieval module). For comparing the performance of shallow and deep methods for QA the system has several versions of each module, each version using a different technique.

For this module we look at the benefits of the adding deeper information to the rules. In order to do that we compare the use of two sets of rules:

– Shallow rules are based on surface information (words), whereby a mapping is defined between an interrogative pronoun and the expected answer type.

[2] http://www.nilc.icmc.usp.br/ arianidf/olincom/trilha1.html

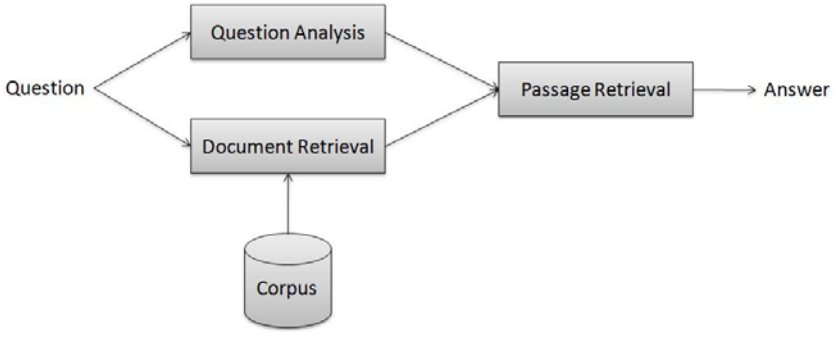


Fig. 2. System architecture developed

– Deep rules are constructed from a deeper analysis of the corpus, using a decision tree classifier (with the J48 algorithm [18]) for determining the expected answer type. The decision tree was trained on a set of 384 questions (from CLEF 2008) manually annotated with the expected answer type on the basis of both word forms and their grammatical classes.

The document retrieval module can be divided into two stages: generating the query and identifying relevant documents. For the first stage (a) all stopwords are removed, (b) terms are normalized to be used in a canonical form, (c) term expansion is performed and finally (d) the query is formatted. These steps are executed for helping the information retrieval (IR) tool to identify the most relevant documents for the query. For this paper, information retrieval is performed using Lucene[3], a standard IR tool, in its basic configuration of a vector space model and a stemmer for Brazilian Portuguese. The documents retrieved by the tool are ranked in order of relevance and passed to the passage retrieval module. In order to avoid a high processing costs in the next module, only the most relevant document is considered.

In order to maximize the data available, the corpus was pre-processed to reduce the number of variants and synonyms of each term, and these were annotated with morphosyntactic information (using the parser PALAVRAS).

The passage retrieval module takes the document retrieved by the IR module and splits it into sentences. Based on the expected type of answer determined by the question analysis module, these sentences are filtered to remove unrelated sentences.

The expected type of answer also determines the appropriate procedure that the module follows. If the required answer is a factoid, a named entity recognizer and a filter for numeric or date information are used. On the other hand, if the expected answer is not a factoid, the module uses (1) ngram comparison, (2) POS comparison of the words in the question and those in the candidate sentence and (3) comparison of subject, predicate and verb of the question and

[3] http://lucene.apache.org/java/docs/index.html

candidate answer (refered to as NP-PP-VP in the text). The candidate passages (both factoid and non-factoids) are ranked using bag of words.

4 Results

To identify the impact of each technique, we tested each with perfect input, by manually cleaning the output of the previous module. After we identify the best methods for each module, we applied the test system as a whole in order to identify the overall performance of the system.

The first module, which processes the question, was tested with the whole corpus. For these 30 questions it obtained an overall accuracy of 36% with shallow rules and 53% with the deep rules (patterns involving e.g. POS tags). Table 1 shows the results considering the types of questions. The main advantage of using a parser is the flexibility it allows of combining different levels of information in the patterns, like lemmas and POS tags. One example is the following rule that matches sentences where the pronoun "quem" followed by a verb (*pronoun="quem" & verb*) like *Quem conseguiu uma virada sobre Ksenia Pervak?* and *Quem é o técnico do Corinthians?*. These examples from the corpus cannot be not captured properly by the simple pattern. The document retrieval module was tested according to the amount of information provided about the input question:

- Shallow question uses all the words in the question directly for performing IR
- Deep question extract nouns terms from the parsed question

Table 2 show the results obtained. In both cases we evaluated the use of Lucene in its standard version (Lucene) and in the Portuguese version (Lucene-BR). In addition we test them using both the original and the pre-processed corpus. The best results were obtained with the (complete) shallow question, suggesting that using all the words in the sentence increases the chances of finding the answer, and further information may not imply in improvement of results. In addition, using Lucene customized for Brazilian Portuguese improved the results when the deep question was used. The last module, passage retrieval, was tested using as input the correct document for the question, as well as the correct identification of the type of expected answer. The results per type of answer are show in table 3.

Table 1. Percentage of correct answers per type of questions

Question Type	Shallow Rules	Deep Rules
Quote	**100%**	83%
Place	**33%**	**33%**
Entity	7%	**46%**
Numeric	33%	**50%**

Table 2. Performance of the document retrieval module

Method	Shallow Question	Deep Question
Lucene	**90%**	83%
Lucene-BR	**90%**	**90%**
Pre-processed corpus and Lucene	70%	86%
Pre-processed corpus and Lucene-BR	86%	86%

Table 3. Results of factoid and non-factoids question and their respective methods

Question Type	Approach	Results
factoids	entity recognizer	73%
non-factoids	bag of words	63%
non-factoids	ngrams	45%
non-factoids	POS tag	18%
non-factoids	NP-PP-VP	9%

Finally, we tested the overall performance of the system using two configurations. The first uses the configuration that provided best results at each stage (deep rules, shallow questions with simple Lucene, entity recognizer in factoids questions and bag of words in non-factoid questions), while the second uses the shallow version of each stage. In both cases the overall performance of the system is similar and not significantly different, reaching 56% accuracy.

5 Conclusions and Future Work

Much research effort has been devoted in recent years to the development of QA systems. Although for resource rich languages like English a variety of combinations of tools and resources have been employed for ensuring a good performance, for other languages like Portuguese, such resources and tools may not exist or be as widely available. Therefore it becomes crucial to determine the impact of different tools and resources over the performance of the system, as deeper processing may not necessarily imply in a huge impact for QA, and the best value for money may be obtained with a combination of shallow and deep methods. In this paper we investigated the benefits of using shallow vs deeper processing in different stages of a Brazilian Portuguese QA system.

In this paper we find that for the question answering task good results can be already obtained using shallow methods, which are on a similar level to the optimal results obtained with a combination of shallow and deep methods. The results obtained analyzing each phase separately suggest that for the stage of question analysis deep methods perform best, using learning algorithms based both in surface (words) and deep forms (parsing information). However, the use of a pos-tagger, which is more widely available for Brazilian Portuguese could supply the information needed. For document retrieval the best result is achieved with shallow information, using the whole question either with Lucene or with

the customized Portuguese version. The latter improves performance even when deep information is used, perhaps compensating for the size of the corpus. In passage retrieval the best results was with the shallow bag-of-words approach. However, due to the problem of data sparseness, further investigation is necessary to determine how much it influences results and/or to corroborate them.

For future work we plan to use a larger corpus for evaluation, such as the CLEF Corpus. In addition, an investigation of the robustness of the results in different domains is also planned.

References

1. Nyberg, E., Mitamura, T., Carbonell, J., Callan, J., Collins, K., Czuba, K., Duggan, M.: The javelin question-answering system at trec 2002. In: TREC (2002)
2. Kaisser, M.: Qualim at trec 2005: Web-question answering with framenet. In: TREC (2005)
3. Lin, J.: Evaluation of resources for question answering evaluation. Technical report, University of Maryland, College Park (2005)
4. Schlaefer, N., Gieselmann, P., Schaaf, T., Waibel, A.: A pattern learning approach to question answering within the ephyra framework. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2006. LNCS (LNAI), vol. 4188, pp. 687–694. Springer, Heidelberg (2006)
5. Zheng, Z.: Answerbus question answering system. In: HLT Human Language Technology Conference, HLT 2002 (2002)
6. Quaresma, P., Quintano, L., Rodrigues, I., Saias, J., Salgueiro, P.: The university of Évora's participation in QA@CLEF-2007. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 253–259. Springer, Heidelberg (2008)
7. Bick, E.: The Parsing System Palavras - Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. PhD thesis, Department of Linguistics, University of Arhus (2000)
8. Sarmento, L.: A first step to address biography generation as an iterative qa task. In: Peters, C., Clough, P., Gey, F.C., Karlgren, J., Magnini, B., Oard, D.W., de Rijke, M., Stempfhuber, M. (eds.) CLEF 2006. LNCS, vol. 4730, pp. 473–482. Springer, Heidelberg (2007)
9. Sarmento, L., Oliveira, E.: Making raposa (fox) smarter. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, Springer, Heidelberg (2008)
10. Amaral, C., Figueira, H., Martins, A., Mendes, A., Mendes, P., Pinto, C.: Priberam's question answering system for portuguese. In: Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., Giampiccolo, D. (eds.) CLEF 2005. LNCS, vol. 4022, pp. 410–419. Springer, Heidelberg (2006)
11. Costa, L.: First evaluation of esfinge – A question answering system for portuguese. In: Peters, C., Clough, P., Gonzalo, J., Jones, G.J.F., Kluck, M., Magnini, B. (eds.) CLEF 2004. LNCS, vol. 3491, pp. 522–533. Springer, Heidelberg (2005)
12. Costa, L.: Esfinge at clef 2008: Experimenting with answer retrieval patterns. can they help? In: Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G.J.F., Kurimo, M., Mandl, T., Peñas, A., Petras, V. (eds.) CLEF 2008. LNCS, vol. 5706. Springer, Heidelberg (2009)

13. Vallin, A., Magnini, B., Ayache, C., Erbach, G., Peñas, A., de Rijke, M., Rocha, P., Simov, K., Sutcliffe, R.: Overview of the CLEF 2004 multilingual question answering track. In: Peters, C., Clough, P., Gonzalo, J., Jones, G.J.F., Kluck, M., Magnini, B. (eds.) CLEF 2004. LNCS, vol. 3491, pp. 371–391. Springer, Heidelberg (2005)
14. Vallin, A., Magnini, B., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peñas, A., de Rijke, M., Sacaleanu, B., Santos, D., Sutcliffe, R.F.E.: Overview of the CLEF 2005 multilingual question answering track. In: Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., Giampiccolo, D. (eds.) CLEF 2005. LNCS, vol. 4022, pp. 307–331. Springer, Heidelberg (2006)
15. Magnini, B., Giampiccolo, D., Forner, P., Ayache, C., Jijkoun, V., Osenova, P., Peñas, A., Rocha, P., Sacaleanu, B., Sutcliffe, R.: Overview of the CLEF 2006 multilingual question answering track. In: Peters, C., Clough, P., Gey, F.C., Karlgren, J., Magnini, B., Oard, D.W., de Rijke, M., Stempfhuber, M. (eds.) CLEF 2006. LNCS, vol. 4730, pp. 223–256. Springer, Heidelberg (2007)
16. Giampiccolo, D., Forner, P., Peñas, A., Ayache, C.: Overview of the clef 2007 multilingual question answering track. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 200–236. Springer, Heidelberg (2008)
17. Forner, P., Peas, A., Alegria, I., Forascu, C., Moreau, N., Osenova, P., Prokopidis, P., Rocha, P., Sacaleanu, B., Sutcliffe, R., Sang, E.T.K.: Overview of the clef 2008 multilingual question answering track. In: Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G.J.F., Kurimo, M., Mandl, T., Peñas, A., Petras, V. (eds.) Evaluating Systems for Multilingual and Multimodal Information Access. LNCS, vol. 5706, pp. 262–295. Springer, Heidelberg (2009)
18. Quinlan, R.: C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco (1993)

Planning for Multi-robot Localization

Paulo Pinheiro and Jacques Wainer

Institute of Computing,
University of Campinas - UNICAMP
6176, Campinas -SP, Brazil
{pinheiro,wainer}@ic.unicamp.br
http://www.ic.unicamp.br/

Abstract. This paper will present a cooperative multi-robot localization model with planning support. Models of communication and transmission of pose estimates are constantly explored, however how the robots act on the environment is generally defined by random actions (from the localization task's point of view). Random actions generate observations that can be useless for improving the estimate. This work describes a proposal for multi-robot localization with planning of actions. The objective is to describe a model where policies define the best action to performed by robots. The proposed model, called Model of Planned Localization - MPL, uses POMDPs to model the problems of location and specific algorithms to generate policies. We compared the MPL to a model that does not make use of planning actions. The results showed that MPL is able to estimate the positions of robots with lower number of steps, being more efficient than model compared.

Keywords: Multi-robot localization, POMDP, Planning, Markov localization.

1 Introduction

The multi-robot localization problem consists in estimating the pose of each robot in a certain environment. The pose of a robot is represented by the position and heading in which the robot is found in relation to the environment.

It is fundamental to know the pose (or at least to have a good estimate) for tasks such as objects manipulation, exploration and development of navigation plans. Even to make use of a map, it is necessary to know the self-location of the robot. It can be stated, therefore, that the self-localization is a prerequisite for many of the tasks performed by mobile robots and this has a conditional relation to the final results. For these reasons, self-localization becomes one of the most important capabilities of a mobile robot [2].

Currently, the concept of cooperation in the localization problem is explored by communication between the robots. The communication contributes by allowing the beliefs about the pose to be shared, therefore, decreasing uncertainties about the poses of each robot. Many models of detection and transmission of information were devised in an attempt to improve the performance of the process of localization.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 183–192, 2010.

The proposal of this work is to present the Model of Planned Localization - MPL in which actions are defined by policies, allowing the robots to maximize the definition of poses. This way, planned actions produce more useful observations, improving the quality of the estimates and allowing the robots to locate each other faster in unknown environments.

2 Related Work and Objective

Regarding the matter of multi-robot localization, previous research have focused on the exploration of communication techniques among the agents. Several techniques for sharing of information were developed.

Many works studied local techniques to the localization problem [3,12,13]. These techniques use the localization process to compensate odometric errors occurred during the locomotion of the robot. Thus, the robots knew its initial pose and the techniques used to correct the errors during its movements in the environment.

To solve the problem of local localization many works use the Kalman filter approach. This approach use Gaussian distributions to define the robot's pose. In Kalman filter approach the robots know the map of the environment and use landmarks observed to estimate their positions. A landmark is any structure in the environment that is recognized by sensors of robots. Example of Kalman filter based approaches are [14] and [15], who used Kalman filter to solve simultaneously the problem of robot localization and the mapping of the unknown environment.

Markov Localization is a probabilistic framework that allows estimating the pose of robots even when the initial position is not known. The approach was used in works such as [8,1,4]. Markov Localization is the approach used in this work.

Usually, the communication among robots is dependent on detection [5]. Whenever a robot detects another one, their probabilities of poses are shared. In [5], when two robots meet, they exchange their estimates of poses, between themselves, and only between themselves. The disadvantage is that only these two robots benefit from the exchange of information.

In [10], the propagation detection model allows for the shared information between two robots to be transmitted to the rest of the group, and is denominated Propagation of Positive Detection. This way, such model enabled significant reduction of uncertainties about localization. This kind of transmission also occurs in [14], however, in this case Kalman filter is used instead of Markov localization.

To further improve the estimates in Markov localization, [6] described the model of Negative Information, in which the absence of observation of landmarks was used to assist in the definition of robot pose. Thus, if the expected landmark was not found then there is the certainty that the robot is not in that location [7].

[11] described how Negative Detection can improve localization. In this model of detection, the absence of detection information can be useful to obtain estimates. For example, a robot can be in a specified room and not locate another robot. The information of absence of detection is transmitted to the second robot, helping it locate itself faster.

So far, all previous research focused on exploring techniques of sharing information but very little has been done to improve the quality of information that is shared. With a focus on communication, these research have not dealt with means to improve the robot's actions, which are usually performed in a random way. The question to be answered is: *what action generate an observation, that when used in the calculation of estimates, it would be more useful to define the robot posture, enabling fewer steps need be taken.*

In this context, this paper proposes a localization model that is able to select the best action at each moment of decision. This is called *Model of Planned Localization* or *MPL*. The objective of MPL is solve the problem of global cooperative multi-robot localization using planning of actions. For each scenario, a policy is generated. Given the policy, the agent selects the best action in each decision period. The best action is one that maximizes the chances of obtaining a relevant observation able to improve the distribution of probabilities of the agent's pose.

3 General Aspects

Before describing the localization problem modeling and the policy generation, it is necessary to define the general aspects of the Model of Planned Localization to define how the localization is treated and what probabilistic technique and detection are used.

3.1 The Localization Technique

The work uses Markov Localization (ML). ML is a probability framework used to estimate the robot pose a according to its observations and the actions performed. This technique defines a probability distribution about all the cells of the environment, which allows for the robot to estimate its own location [3].

One of the reasons to use Markov Localization is its association to POMDPs (Partially Observable Markov Decision Processes), which were utilized in MPL. In a POMDP, the agent does not know the global state of the system and is able to remember the actions and observations that it performed over time. This information is used to make the next decision for which it is granted a reward [16].

3.2 The Detection Model

The detection models used are: positive detection model and negative detection model, the same that was presented by [9]. In this model, the absence of detection may be useful information in the process of localizations of the robots.

One example of this usefulness can be described in the following scenery: a robot r_1 is isolated in a room in which it tries to locate itself. A robot r_2 is outside of the room with a better pose distribution, and informs r_1 in which regions the isolated robot is not located; therefore helping it to decrease its uncertainty of self-location identification. Whenever one of the robots does not detect another in the environment, the robot updates its estimates. The work of [9] describes this model with more details.

3.3 The Propagation Model

MPL uses the transmission model by positive detection propagation. In this model, when there is a detection between two robots, the detection information is sent to all the other robots in the environment. Two phases are important, the first is the transmission of the new data set package to the other robots, and the second is, given this package, how to update the position beliefs.

Let us consider that a robot r_1 detects a robot r_2 and that r_3 is the robot that does not participate in the detection. At this point, r_1 sends its beliefs to r_2 and transmits to r_3 the information about detection.

In positive detection propagation, the information transmitted to r_3 is given by: the probability distribution of r_1, the probability distribution of r_2 and the probability distribution of any robot being between the two robots that participate in the detection.

The second phase is the updating of beliefs of the non participant robot r_3.The work of [11] describes the model of positive detection propagation and their contributions.

4 Localization Problem Modeling

Before producing a plan for the robots, it is necessary to model the problem. In this phase, several forms of modeling were tried to make it possible to produce efficient policies. For this, we applied the formalism of POMDP (Partially Observable Markov Decision Process).

POMDP model is a tuple $\langle S, A, T, R, W, O\rangle$, where S is a finite set of states, A is a finite set of actions, T is a markovian transition probability, R is a reward function, W is a finite set of observations and O is a table of observations probabilities [16].

Since it is a localization problem, we developed a mechanism that could represent such a problem. The mechanism allows for the robot to perform a special action, which we name *shout*, when it notices that it already knows its position in the environment. This action will reward it positively when the estimate of pose presented is correct and negatively when it is wrong. The weights for these rewards were tested exhaustively.

To demonstrate the modeling with the *Shout technique*, we will use an example of a localization problem.

Example 1 (map 2x3). A robot is in an environment of 2x3 size, made up of cells of 1x1 size; the environment presents obstacles. We suppose that there is a total of 4 free cells in the environment (1,2,3,4) as show Fig. 1; the robot is able to move with the following actions: walk towards the north, south, east,or west.

In this case, formalize the problem as:

- S is a finite set of states, where $S = s_1, s_2, s_3, s_4$;
- A is a finite set of actions available to agent, where in this scenario A = $\{move_{north}, move_{south}, move_{east}, move_{west}, shout_1, shout_2, shout_3, shout_4\}$;

Fig. 1. Environment whit four states

- T is a transition probability table. $[T : a : s : s^{'}]$ denotes the probability that taking joint action a in state s, results in a transition to state $s^{'}$. Thus, for the state e_3, for example:
  $T : move_{north} : e_3 : e_1$ $T : move_{south} : e_3 : e_3$ $T : move_{east} : e_3 : e_4$
  $T : move_{west} : e_3 : e_2$ $T : shout_1 : e_3 : e_3$ $T : shout_2 : e_3 : e_3$
  $T : shout_3 : e_3 : e_3$ $T : shout_4 : e_3 : e_3$

- R: For each move action, the reward is -1. For each inform action, where agent pose is correct, the reward is +100, otherwise it is penalized with -100; in such case, for example, if agent stay in state s_3 and choose some of those actions: $shout_1$, $shout_2$, $shout_4$, it looses 100; but if it choose the action $shout_3$ the reward is +100;
- W is a finite set of observations for agent i, where $W = \{obstacle_{west}$, $obstacle_{east}$, $obstacle_{none}$ $obstacle_{both}\}$
- O is a observation probabilities where O denotes the probability p that staying in state s, receive one of those observations o. Thus $[O : s : o : p]$ and in this case:
  $O : e_1 : obstacle_{both} : 0.9$, $O : e_2 : obstacle_{west} : 0.9$,
  $O : e_3 : obstacle_{none} : 0.9$, $O : e_4 : obstacle_{east} : 0.9$.

We suppose that the initial probability of robot pose is the same in all the states of the environment. It is easy to notice, that modeling using the *Shout technique*, allows the robot to look for the greatest reward and therefore make a correct localization of the environment. Once the problem has been modeled, it is possible to determine an optimum policy of actions by resolving the corresponding POMDP.

5 Experiments

In this section we present experiments that compare MPL to a model that does not include planning for localization. The comparative model and MPL uses i)positive and negative detection, ii)positive and negative information and iii)propagation of positive detection. The difference is that the MPL use planning actions.

The robot and environments characteristics are similar to the ones found in the research by [9], so as to enable the comparison of the impact of planning in relation to this model. The robots have a proximity sensor that can measure

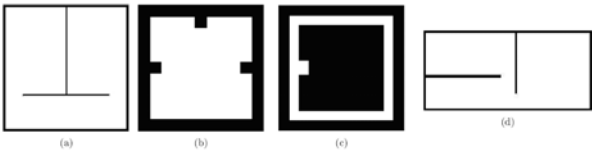


Fig. 2. Environments (a), (b), (c) and (d)

distances to an obstacle up to 15 cells ahead. The robots have a localization sensor that identifies other robots up to 17 cells of distance.

The environments used in this experiment are shown in Fig. 2. The lighter areas represent the free spaces and the dark, obstacles. The environments are divided into grades, and the sizes of data per unit cell.

The first environment (a) is 15x15, representing two rooms and a hall in common. The environment (b) measures 20x20 and has a larger area free of obstacles. The environment (c), also found in the [5] work, is 30x30 and the environment (d), the largest of all environments, is 40x18.

In the simulator, we select the environment and the number of robots that will be utilized. Each robot knows the environment map, however its initial pose is not known. Each robot has a probability distribution that identifies a different value of pose belief for each state of the environment. The robots are distributed randomly in the environment. Making use of the policy generated, the observations that are obtained with the new actions and the communication with the other robots, the believes are updated.

The algorithms tested were unable to produce policies for the environments with an amount of states larger than 1400. Aspects of the modeling, such as sensory abilities and the movements of the robots, interfere in the policy generation performance. However the amount of states is a more relevant factor, since it is associated with functions of transition and individual observations of each state. Table 1 describes the number of states of environments. Of all the environments tested, (d) is the biggest, with 2744 states.

Since the POMDP solution algorithms could not generate a policy for environment (d), no experiments were performed using it.

6 Results

In this section we present the comparative results between the performance of MPL and the comparative model (which does not support the planning of actions). We calculated the average number of steps and the average error in localization of the robots. The number of steps is the number of times a robot updates its beliefs, so any action, that generates a new update, will be recorded as a step. After each step the robot localization error is measured. This error is given by the euclidean distance (in meters) between the actual position of the robot and the estimated position, that is, one in which the value of belief is highest.

6.1 An Example

In environment (a), MPL obtained slightly better results than the comparative model. Fig. 3a describes these results. In this environment, MPL allowed for the robots to estimate their poses, in average, after five steps. As for the comparative model, the poses were found in six steps.

Thus, in environments consisted of rooms and halls, with size which approximates that of the vision reach of the agents, the application of negative detection becomes more important than the use of planning, for example. However, the negative detection, once applied together with planning, allows for a much faster definition of pose.

Fig. 3b shows the results for environment (b). In this environment, the robots that used the planning model, estimated their poses in 24 steps; and the robots that used the comparative model estimated their poses in 59 steps.

In the environment (c), the use of MPL considerably decreased the number of average steps in the localization process. On average only 39 steps were necessary, against 115 steps in the comparative model. Fig. 3c shows the results for environment (c).

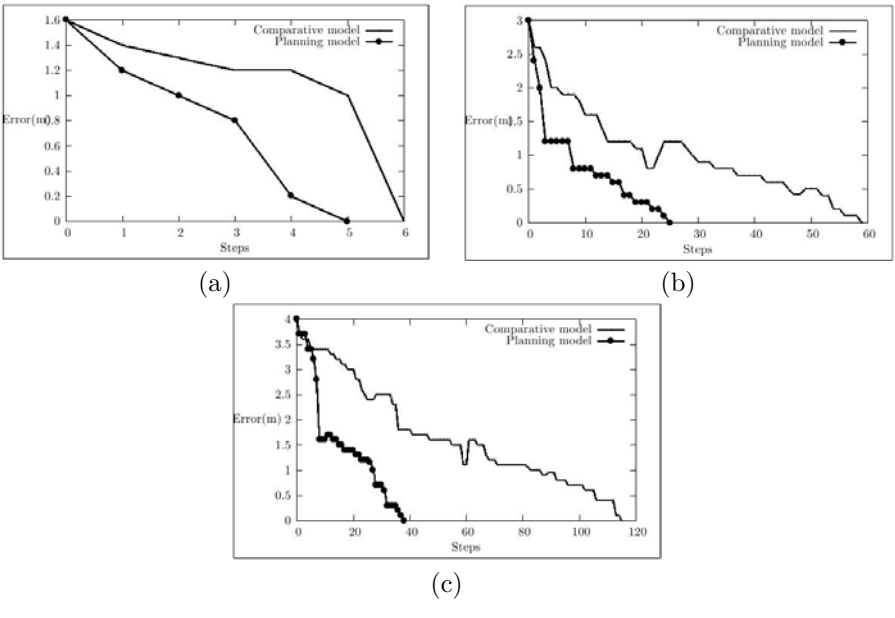


Fig. 3. Results for the environment (a), (b), (c)

6.2 Summary for All Cases

To compare the two models we compared both the MPL and the comparative model (*CModel*) regarding the number of steps until the localization was achieved. We also calculated the average gain of the MPL in relation to the

Table 1. Number of states and simulations

Environment	States	Simulations (2%)	p-value
(a)	824	17	0.008
(b)	1248	25	0.000
(c)	848	17	0.000

comparative model (*CModel*). The number of simulations executed corresponds to the value of 2% of environment states, as demonstrated by Table 1. This table also shows the *p-value* of *t-test* for each set of samples A pair of robots was used in each simulation.

Table 2a reports the number of steps for the simulations for environment (a). The pairwise t-test shows that the p-value is 0.008, that it, with more than 99% of confidence, the differences between the MPL and the comparative model are significant, and the average gain for the MPL was 22%, in other words, using MPL, the agents were located with a number of steps, 22% lower than the amount required in the comparative model.

For the environment (b), again the p-value was below 0.001, and thus with more than 99% confidence, the results are significantly different and the average gain for MPL was approximately 59%. Table 2b shows the number of steps until localization. In the environment (c) using the MPL, the agents executed a number of steps 72 % lower than the number of steps executed by other agents

Table 2. Number of steps until localization for the environments (a), (b), (c)

(a)

Environment (a)																	
sim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
CModel	9	8	7	10	7	5	12	11	8	12	9	18	20	5	12	10	8
MPL	11	6	7	5	5	6	8	4	6	6	8	8	6	6	8	7	10

(b)

Environment (b)																									
sim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
CModel	70	69	35	74	70	55	76	69	69	56	68	70	70	38	72	54	53	64	66	72	50	77	70	69	62
MPL	21	30	17	25	25	24	28	19	19	32	35	25	30	20	19	10	25	28	30	32	25	25	37	24	30

(c)

Environment (c)																	
sim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
CModel	150	145	143	156	130	148	160	150	160	140	110	106	150	120	110	131	145
MPL	40	42	33	42	28	45	40	34	32	45	30	41	42	34	29	35	41

in the comparative model. Table 2c describes the results. Again the differences are statistically significant.

7 Conclusion

In this research, we compared the model with planning, negative detection and transmission model by positive detection propagation (Model of Planned Localization - MPL) to a model of localization without planning support.

According to the experiments, the MPL was able to accelerate the process of localization, when compared to the traditional model. The average step count was smaller, which allowed for quicker estimates of the pose and with a smaller amount of exchanged messages.

The comparison between models using *t-test* showed with 99% level of confidence that the MPL has generated results significantly better than the comparative model, with a reduction in the number of steps taken by robots in the localization process.

It must be pointed out that our research is orthogonal and in some sense complements that of [5,10,11]. These works are interested in the communication among the agents, but they assume that localization is an opportunistic task, that is, the robot localization is done while it tries to achieve its other, more important goals. Thus, in these models, the localization process does not have control over the actions to be performed, which is modeled as random actions (from the localization task' point of view). We assume that localizations is a goal on itself, maybe a prerequisite for other goals, and our model plans the actions that will achieve it as quickly as possible.

This opens up a interesting line of future research. Probably, in real environments there are cases in which the uncertainties in localization are not as high that it precludes the robot from achieving its other goals. In these cases, the standard Markov localization models are appropriate. In other cases, the uncertainty on the pose maybe too high that it is futile to try to achieve these other goals, and thus MPL is the appropriate choice. How to decide when it is worth to follow the actions proposed the the MPL model instead of trying to achieve its other goals it still unclear to us, and deserve further research.

Acknowledgments. The authors wish to acknowledge the financial support of CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), grants 141061/2010-3 and FAPESP (Fundação de Amparo à Pesquisa de São Paulo), grants 2007/53606-2.

References

1. Cassandra, A., Kaelbling, L.P., Kurien, J.: Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 963–972 (1996)

2. Cox, I.J.: Blanche: an experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transaction on Robotics and Automation 7(2), 193–204 (1991)
3. Fox, D.: Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation. PhD thesis, University of Bonn, Germany, Germany (1998)
4. Fox, D., Burgard, W., Thrun, S.: Active markov localization for mobile robots. Robotics and Autonomous Systems 25, 195–207 (1998)
5. Fox, D., et al.: A probabilistic approach to collaborative multi-robot localization. Autonomous Robots 8(3), 325–344 (2000)
6. Hoffman, J., et al.: Making use of what you don't see: negative information in markov localization. In: International Conference on Intelligent Robots and Systems, vol. 1, pp. 2947–2952 (2005)
7. Hoffmann, J., Spranger, M., Gohring, M., Jungel, M.: Exploiting the unexpected: Negative evidence modeling and proprioceptive motion modeling for improved markov localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 24–35. Springer, Heidelberg (2006)
8. Nourbakhsh, I., Powers, R., Birchfield, S.: Dervish: An office-navigation robot. AI Magazine 16(2), 53–60 (1995)
9. Odakura, V.: Localização de Markov para multirrobôs cooperativos. PhD thesis, Escola Politécnica, USP, São Paulo (2006)
10. Odakura, V., Costa, A.H.R.: Cooperative multi-robot localization: using communication to reduce localization error. In: International Conference on Informatics in Control Automation and Robotics - ICINCO, vol. 3, pp. 88–93 (2005)
11. Odakura, V., Costa, A.H.R.: Negative information in cooperative multirobot localization. In: Sichman, J.S., Coelho, H., Rezende, S.O. (eds.) IBERAMIA 2006 and SBIA 2006. LNCS (LNAI), vol. 4140, pp. 552–561. Springer, Heidelberg (2006)
12. Romero, L., Arellano, J.J.: Robust local localization of a mobile robot using a 180 2-d laser range finder. In: Proceedings of the Sixth Mexican International Conference on Computer Science, vol. 1, pp. 248–255. IEEE Computer Society, Los Alamitos (2005)
13. Romero, L., Lara, C.: Robust local localization of a mobile robot in indoor environments using virtual corners. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 901–910. Springer, Heidelberg (2007)
14. Roumeliotis, S., Bekey, G.A.: Distributed multirobot localization. IEEE Transactions on Robotics and Automation 18, 781–795 (2002)
15. Tardos, J.D., et al.: Robust mapping and localization in indoor environments using sonar. The International Journal of Robotics Research 21(4), 311–330 (2002)
16. Theocharous, G., Murphy, K., Pack, L.: Representing hierarchical pomdps as dbns for multi-scale robot localization. In: International Conference on Robotics and Automation, vol. 1, pp. 1045–1051 (2004)

Symbolic Bounded Real-Time Dynamic Programming

Karina Valdivia Delgado[1], Cheng Fang[2], Scott Sanner[3], and Leliane Nunes de Barros[1]

[1] University of São Paulo, São Paulo, Brazil
[2] University of Sydney, Sydney, Australia
[3] National ICT Australia, Canberra, Australia

Abstract. Real-time dynamic programming (RTDP) solves Markov decision processes (MDPs) when the initial state is restricted. By visiting (and updating) only a fraction of the state space, this approach can be used to solve problems with intractably large state space. In order to improve the performance of RTDP, a variant based on symbolic representation was proposed, named sRTDP. Traditional RTDP approaches work best on problems with sparse transition matrices where they can often efficiently achieve ϵ-convergence without visiting all states; however, on problems with dense transition matrices where most states are reachable in one step, the sRTDP approach shows an advantage over traditional RTDP by up to three orders of magnitude, as we demonstrate in this paper. We also specify a new variant of sRTDP based on BRTDP, named sBRTDP, which converges quickly when compared to RTDP variants, since it does less updating by making a better choice of the next state to be visited.

1 Introduction

Markov Decision Processes (MDPs) [1] provide a convenient framework for modeling fully-observable stochastic planning problems. In an MDP, the agent computes a policy — a mapping from states to actions — in order to maximize a stream of rewards. A popular approach to policy computation is through a value function — a function that assigns a value to each world state. The computation of the value function can be either synchronous, where all states are updated during each iteration, or asynchronous, where the agent updates some states more than others.

Recent years have seen a resurgence of interest in asynchronous dynamic programming solutions to MDPs [2]. Of particular interest has been the trial-based real-time dynamic programming (RTDP) approach [3] as evidenced by a variety of recent work [4,5,6]. RTDP algorithms have a number of distinct advantages for practical MDP solutions, specifically: (a) *Anytime performance:* RTDP algorithms can be interrupted at any time, generally yielding a better solution the longer they are allowed to run; (b) *Optimality without exhaustive exploration:* By focusing trial-based search on states reachable from the set of initial states, RTDP algorithms may obtain an optimal policy while visiting only a fraction of the state space.

However, the advantages of RTDP may break down when the transition matrix is not sparse. Yet many MDPs that model interesting real-world problems do not have sparse transition matrices, e.g., elevator scheduling with random arrivals, traffic control with random car movements, and logistical problems with random failures. All of these

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 193–202, 2010.

problems have the property of *exogenous events* — events beyond the agent's control that can cause arbitrary state changes between time steps that lead to *dense and large transition matrices*. Although dense transition matrices lack structure in terms of sparsity, they often contain factored (symbolic) structure. Based on this last idea, Feng et al (2003) has proposed a factored variant of RTDP named Symbolic RTDP (sRTDP). However they have not shown how this sRTDP behaves when leading with dense transition matrices in large state spaces, as we demonstrate in this paper. Besides, we propose a new variant of sRTDP, a Symbolic Bounded RTDP, which maintains both upper and lower bounds on the optimal value function. We show that with this approach we make less value updates when compared to sRTDP [7].

2 Background

A **Markov decision process** (MDP) is a tuple $\langle S, A, T, R, \gamma \rangle$ [1]. $S = \{s_1, \ldots, s_n\}$ is a finite set of fully observable states. $A = \{a_1, \ldots, a_m\}$ is a finite set of actions. $T : S \times A \times S \rightarrow [0, 1]$ is a known stationary, Markovian transition function. $R : S \times A \rightarrow \mathbb{R}$ is a fixed known reward function associated with every state and action. γ is a discount factor s.t. $0 \leq \gamma \leq 1$ where rewards t time steps in the future are discounted by γ^t. There is a set of initial states $\mathcal{I} \subseteq S$, and a possibly empty set of absorbing goal states $\mathcal{G} \subset S$ where all actions lead to a zero-reward self-transition with probability 1.

A policy $\pi : S \rightarrow A$ specifies the action $a = \pi(s)$ to take in state s. Our goal is to find a policy that maximizes the value function, defined as the sum of expected discounted rewards $V_\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t \middle| s_0 = s \right]$, where r_t is the reward obtained at time t.

Value iteration (VI) is a *synchronous* dynamic programming (DP) solution to an MDP. Starting with an arbitrary $V^0(s)$, VI performs value updates for *all* states s, computing the next value function $V^{t+1}(s) := \textsc{Update}(V^t, s)$:

$$Q^{t+1}(s, a) := R(s, a) + \gamma \cdot \sum_{s' \in S} T(s, a, s') \cdot V^t(s') \tag{1}$$

$$V^{t+1}(s) := \max_{a \in A} \left\{ Q^{t+1}(s, a) \right\}. \tag{2}$$

This update is known as a *Bellman update*. For discounted problems and certain restrictions of undiscounted problems such as *stochastic shortest path* (SSP) problems [2], $V^t(s)$ converges to the unique optimal value function $V^*(s)$ in the infinite limit of updates, i.e., defining $\epsilon_t = \max_s |V^t(s) - V^*(s)|$ then $\lim_{t \rightarrow \infty} \epsilon_t = 0$. The greedy policy $\pi(s) = \textsc{GreedyAction}(V^t, s)$ w.r.t. V^t and state s is defined as follows:

$$\pi(s) := \arg\max_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \cdot V^t(s') \right\} \tag{3}$$

For discounted problems, the greedy policy π derived from V^t loses no more than $2\gamma\epsilon_t$ in value over the infinite horizon compared to the optimal policy [1].

Asynchronous DP & Real-time DP. *Asynchronous* DP methods [2] are a variant of dynamic programming that apply the Bellman update to states in an arbitrary order

Algorithm 1. RTDP

begin
  // Initialize $\hat{V}_u$ with admissible value function
  $\hat{V}_u := V_u$
  while *convergence not detected and not out of time* **do**
    $depth := 0$
    $visited$.CLEAR() *// Clear visited states stack*
    Draw s from $\mathcal{I}$ at random *// Pick initial state*
    while $(s \notin \mathcal{G}) \wedge (s \neq null) \wedge (depth < max\text{-}depth)$ **do**
      $depth := depth + 1$
      $visited$.PUSH(s)
      $\hat{V}_u(s) :=$ UPDATE($\hat{V}_u, s$) *// See (1) & (2)*
      $a :=$ GREEDYACTION($\hat{V}_u, s$) *// See (3)*
      $s :=$ CHOOSENEXTSTATERTDP(s, a) *// See (4)*
    // Optimization: end-of-trial update
    // not appearing in the original RTDP
    while $\neg$*visited*.EMPTY() **do**
      $s :=$ *visited*.POP()
      $\hat{V}_u(s) :=$ UPDATE($\hat{V}_u, s$)
  return $\hat{V}_u$
end

while still retaining convergence properties under certain conditions. The real-time dynamic programming (RTDP) [3] algorithm (Algorithm 1) is a popular asynchronous DP approach that updates states encountered during trial-based simulations of an MDP. This variant of DP explores the state space in depth-limited trials and performs Bellman updates at each state visited. RTDP selects states to visit by drawing next state samples s' from the transition distribution for the current greedy action a and current state s, i.e.,

$$\text{CHOOSENEXTSTATE}(s, a) := s' \sim T(s, a, \cdot). \tag{4}$$

We say that V_u is an *admissible* upper bound on the optimal value $V^*(s)$ if $V_u(s) \geq V^*(s); \forall s$. Given an admissible upper bound, RTDP converges to the optimal value function in the infinite limit of trials. The primary advantage of RTDP is that its solution is *targeted* to the initial state set. One weakness of RTDP is that unlikely paths tend to be ignored and as a consequence the convergence of RTDP is slow [4]. Thus, some extensions of RTDP were proposed in order to improve the convergence: Labeled RTDP (LRTDP) [4], Bounded RTDP (BRTDP) [5] and Bayesian RTDP [6].

Bounded RTDP (BRTDP) [5] maintains upper and lower bounds on the optimal value function, $V_u(s)$ and $V_l(s)$ respectively, and focus search in areas with high value uncertainty. It was proved that the subsequent updates of upper and lower bounds, defined as $V_u(s) \geq V^*(s)\ \forall s \in S$ and $V_l(s) \leq V^*(s)\ \forall s \in S$, preserve admissibility and monotonically converge to $V^*(s)$ [5]:

$$\lim_{t\to\infty} V_l^t(s) = V^*(s) = V_u^t(s).$$

The gap between upper and lower bounds provides a measure of the value uncertainty for state s. BRTDP (Algorithm 2) [5] first initializes both $\widehat{V}_u$ and $\widehat{V}_l$ with an admissible upper and lower bounds and then perform many trials. Each trial starts choosing an initial state and for each visited state, upper and lower values are updated and a greedy action is chosen. BRTDP prioritizes the choice of next state according to their *bound gap weighted distribution* $\frac{b(\cdot)}{B}$ (Algorithm 3), where $b(s')$ is given by:

$$b(s') = P(s'|s,a)(\widehat{V}_u(s') - \widehat{V}_l(s')). \tag{5}$$

BRTDP finishes a trial if encounters a goal state, a limited depth is achieved or there is low value uncertainty. BRTDP converges for the simple reason that it still updates all relevant states that RTDP would update, but with a different distribution which biases the updates for more uncertain states (large gap between upper and lower bound), thus reducing the value uncertainty more quickly and leading to faster convergence.

3 Symbolic MDPs and RTDP

Many MDPs often have a natural factored (symbolic) structure that can be exploited in the form of a *factored MDP* [8]. Here, states are represented by vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ of length n, where each $x_i \in X_i$ and for simplicity, we assume $X_i = \{0, 1\}$; hence the total number of states is $N = 2^n$. Two ways to exploit factored structure in an MDP first jointly introduced in the SPUDD value iteration algorithm [8] are the use of *dynamic Bayesian networks* (DBNs) to represent the transition function and *algebraic decision diagrams* (ADDs) [9] to represent the reward, value function, and *conditional probability tables* (CPTs) that comprise the DBN.

ADDs provide an efficient means for representing and performing arithmetic operations on functions from a factored boolean domain to a real-valued range (i.e., $\{0, 1\}^n \rightarrow \mathbb{R}$). They rely on two main principles to do this: (a) ADDs represent a function $\mathbb{B}^n \rightarrow \mathbb{R}$ as a directed acyclic graph (DAG) – essentially a decision tree with reconvergent branches and real-valued terminal nodes and (b) ADDs enforce a strict variable

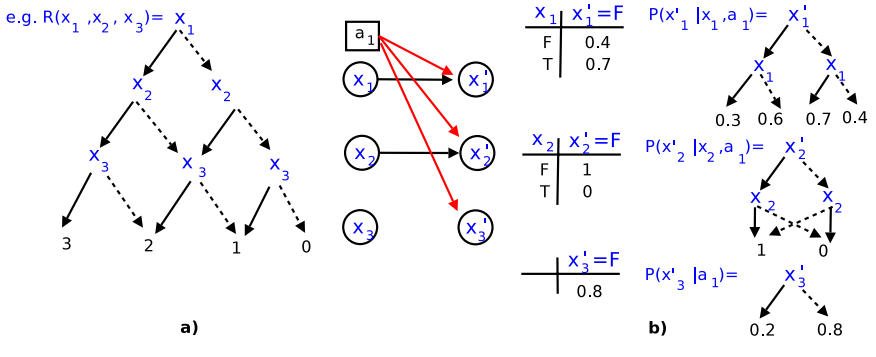


Fig. 1. a)Factored MDP reward and b) transition function example. In all ADDs, true branches are solid, false branches are dashed.

Algorithm 2. BRTDP

begin
  $\widehat{V}_u = V_u$;
  $\widehat{V}_l = V_l$;
  while *convergence not detected and not out of time* **do**
    depth=0;
    visited.Clear();
    Draw s from I at random;
    $s_0 = s$;
    while *($s \notin G$) and ($s \neq$ null) and (depth $<$ max_depth)* **do**
      depth=depth+1;
      visited.Push(s);
      $\widehat{V}_u(s)$ =Update $(\widehat{V}_u, s)$ *// See (1) & (2)*
      $\widehat{V}_l(s)$ =Update $(\widehat{V}_l, s)$ *// See (1) & (2)*
      a =GreedyAction $(\widehat{V}_u, s)$ *// See (3)*
      s =ChooseNextStateBRTDP (s_0, s, a, τ) *// See Algorithm 3*
    while $\neg$ *visited.Empty()* **do**
      s=visited.Pop();
      $\widehat{V}_u(s)$ =Update $(\widehat{V}_u, s)$;
      $\widehat{V}_l(s)$ =Update $(\widehat{V}_l, s)$;
  return $\widehat{V}_u$;
end

Algorithm 3. ChooseNextStateBRTDP(s_0,s,a, τ)

begin
  $\forall s', b(s') = P(s'|s,a)(\widehat{V}_u(s') - \widehat{V}_l(s'))$;
  $B = \sum_{s'} b(s')$;
  if $B < \frac{\widehat{V}_u(s_0) - \widehat{V}_l(s_0)}{\tau}$ **then**
    return *null*;
  return $s' \sim \frac{b(\cdot)}{B}$;
end

ordering on the decisions from the root to the terminal node, enabling a minimal, canonical diagram to be produced for a given function.

ADDs often provide an efficient representation of functions with context-specific independence [10] and redundant structure. For example, the function $\sum_{i=1}^{3} x_i$ ($x_i \in \{0, 1\}$) represented in Figure 1.a as an ADD exploits the redundant structure of sub-diagrams in a DAG to avoid an exponentially-sized tree representation. Unary operations such as scalar multiplication ($\cdot$) and marginalization ($\sum_{x_i \in X_i}$) and binary operations such as addition ($\oplus$), multiplication ($\oplus$), and $\max$ can be performed efficiently on ADDs [9], hence all operations required for DP can be performed directly with ADDs.

A DBN representation of a transition matrix factors the transition probabilities into CPTs $P(x_i'|\boldsymbol{x}_i, a)$ where each next state variable x_i' is only dependent upon the action

Algorithm 4. CHOOSENEXTSTATE_SRTDP($\boldsymbol{x}, a$) $\longrightarrow \boldsymbol{x}'$

begin
  // Sample each next state variable x'_i from DBN
  for *all* X'_i **do**
    $x'_i \sim CPT_a^{x'_i}(\boldsymbol{x})$
  return $\boldsymbol{x}' := (x'_1, \ldots, x'_n)$
end

a and its direct parents $\boldsymbol{x}_i$ in the DBN. Then the transition model can be *compactly* specified as $P(\boldsymbol{x}'|\boldsymbol{x}, a) = \prod_{i=1}^{n} P(x'_i|\boldsymbol{x}_i, a)$ *even when most of the probabilities are non-zero*. Rather than represent each CPT $P(x'_i|\boldsymbol{x}_i, a)$ in a tabular format, ADDs are often more compact as shown in Figure 1.b and facilitate efficient computation.

3.1 sRTDP

Feng et al (2003) proposed the symbolic RTDP (sRTDP) based on the ideas of Hoey et al (1999), i.e., representing the MDP using DBN and ADDs. In this section we explain in details the sRTDP algorithm. We begin by presenting factored forms of (1) and (2):

$$Q^{t+1}(\boldsymbol{x}, a) := R(\boldsymbol{x}, a) + \gamma \sum_{\boldsymbol{x}'} \prod_{i=1}^{n} P(x'_i|\boldsymbol{x}_i, a) V^t(\boldsymbol{x}') \tag{6}$$

$$V^{t+1}(\boldsymbol{x}) := \max_{a \in A} Q^{t+1}(\boldsymbol{x}, a). \tag{7}$$

From these observations, we can now proceed to define the Symbolic RTDP. To do this, we still use the basic RTDP procedure in Algorithm 1, but specify an initialization of $\hat{V}_u$ as an ADD (just a constant if an uninformed upper bound is used), as well as factored versions of the following:

- UPDATE computes the factored RTDP value function update for state $\boldsymbol{x}$. All operations in (6) and (7) can be computed using ADD operations when the reward and value functions and transition CPTs (denoted $CPT^a_{\boldsymbol{x}}(X'_i) = P(X'_i|\boldsymbol{x}, a)$ for each X'_i) are ADDs. In RTDP, computations of $V^{t+1}(\boldsymbol{x})$ yield a constant v for a known $\boldsymbol{x}$. Then sRTDP needs only insert this new constant v into the current value function V^t_{DD} for state $\boldsymbol{x}$ to obtain V^{t+1}_{DD}. This can also be done efficiently in ADDs.
- GREEDYACTION At the same time that UPDATE is performed, the greedy action $\pi(\boldsymbol{x})$ can be easily computed by keeping track of the $\arg\max_a$ when calculating v.
- CHOOSENEXTSTATE (Algorithm 4) samples a next state $\boldsymbol{x}'$ given a current state $\boldsymbol{x}$ and action a by using DBN structure to sample each next state variable x'_i.

3.2 sBRTDP

We can now define a new symbolic variant of BRTDP (sBRTDP). We modify Algorithm 2 first by initializing $\widehat{V}_u$ and $\widehat{V}_l$ as ADDs. Then we use the same Update and GreedyAction procedures from sRTDP. The major difference between sRTDP and sBRTDP is the way they choose the next state, which we explain here in details. We can factor the bound gap weighted distribution $\frac{b(\cdot)}{B}$, that is computed in the BRTDP algorithm, as:

$$P(x'_1, ..., x'_n|\boldsymbol{x}) = P(x'_n|x'_1, ..., x'_{n-1}, \boldsymbol{x})...P(x'_2|x'_1, \boldsymbol{x})P(x'_1|\boldsymbol{x}). \quad (8)$$

Note that we can sample x_1 independently, then condition on it, we can sample x_2, until we reach x_n, we will use this sequential sampling method that is exact. We begin by presenting a symbolic form of $b(s')$ (Equation (5)):

$$b(\boldsymbol{x}') = \prod_{i=1}^{n} P(x'_i|\boldsymbol{x}, a)V_{GAP}(\boldsymbol{x}')$$

where $V_{GAP} = \widehat{V}_u - \widehat{V}_l$. Then starting from x'_1 we do the following:

$$p(x'_1) \propto b(x'_1) = \sum_{x'_i, i\neq 1} \prod_{i=1}^{n} P(x'_i|\boldsymbol{x}, a)V_{GAP}(\boldsymbol{x}').$$

Exploiting the absence of synchronic arcs to decompose, we obtain:

$$b(x'_1) = P(x'_1|\boldsymbol{x}, a) \sum_{x'_2} P(x'_2|\boldsymbol{x}, a) \sum_{x'_3} P(x'_3|\boldsymbol{x}, a) \cdots \sum_{x'_n} P(x'_n|\boldsymbol{x}, a)V_{GAP}(\boldsymbol{x}').$$

Besides the symbolic calculation of $b(s)$, sBRTDP does also the sample of the state variables using an extended *cached ADD*, i.e., an ADD with labeled arcs. In order to do so, sBRTDP constructs a cached V_{GAP} ADD structure: first the transition probabilities of the current state are cached in the arcs of the V_{GAP} ADD and, in a bottom-up fashion, the sums from Equation (9) are also cached in the same structure, starting from inside to outside □s:

$$b(x'_1) = P(x'_1|\boldsymbol{x}, a) \sum_{x'_2} \boxed{P(x'_2|\boldsymbol{x}, a) \textstyle\sum_{x'_3} \boxed{P(x'_3|\boldsymbol{x}, a) \cdots \sum_{x'_n} \boxed{P(x'_n|\boldsymbol{x}, a)V_{GAP}(\boldsymbol{x}')}}} \quad (9)$$

Finally, with the resulting V_{GAP} ADD, sBRTDP samples each state variable in the ADD from top to down. This can be done since each box in Equation (9) is precisely one of $P(x'_j|x'_1, ..., x'_{j-1}, \boldsymbol{x})$ in Equation (8), once we have conditioned on the already sampled variables $(x'_1, \cdot, x'_{j-1})$. Then if the factorization in Equation (8) is done in the same order as the ADD variable order, the computation of $P(x'_j|x'_1, ..., x'_{j-1}, \boldsymbol{x})$ always refers to the subdiagram below $(x'_1, ..., x'_{j-1})$ in the ADD and indeed the recursive marginal computations required to compute the true/false probabilities for x_j are already stored locally on the branches of x_j in the subdiagram.

Figure 2 shows an example. Suppose we are in the state $x_1 = T$, $x_2 = T$ and $x_3 = T$; the greedy action is a_1 and we want to choose the next state. First we cache the probabilities from the CPTs (Figure 2.a) in V_{GAP} (Figure 2.b), and then we compute and cache all □s from Equation (9) in V_{GAP} from bottom to up (Figure 2.c). Based on these values we can now sample each state variable (top-down) in turn, conditional on the current values of the above variables that has been sampled. For example using the cached values in V_{GAP} (Figure 2.c), suppose that we have sampled $x'_1 = T$ and $x'_2 = F$. The state variable x'_3, which is not represented in the ADD in the case of $x'_1 = T$ and $x'_2 = F$, is sampled with 0.5 probabilities. Algorithm 5 has as input the initial and current state, action a, a constant τ, $V_{U_{DD}}$ and $V_{L_{DD}}$ and returns $\boldsymbol{x}'$ sampled according with the previous described procedure.

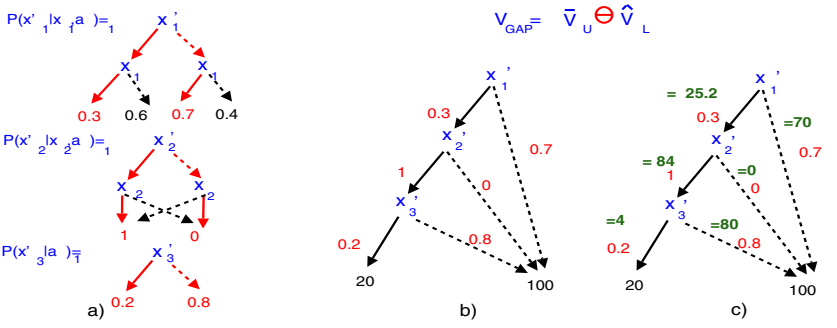


Fig. 2. Computing $b(\boldsymbol{x}') = \prod_{i=1}^{n} P(x_i'|\boldsymbol{x}, a) V_{GAP}(\boldsymbol{x}')$ caching the values in V_{GAP}

Algorithm 5. `ChooseNextState_sBRTDP`($\boldsymbol{x_0}$,$\boldsymbol{x}$,a, τ,$V_{U_{DD}}$,$V_{L_{DD}}$) $\longrightarrow \boldsymbol{x}'$

begin
  $V_{GAP} = V_{U_{DD}} - V_{L_{DD}}$;
  for *all* X_i' **do**
    put $CPT_a^{x_i'}(\boldsymbol{x})$ in V_{GAP}
  bottom-up in V_{GAP} compute $b(x_1')$ (Equation 9) and cache □s in V_{GAP};
  for *all* X_i' **do**
    // top-down in V_{GAP}
    $B = \sum_{x_i'} b(x_i')$;
    if $B < \frac{\hat{V}_u(\boldsymbol{x_0}) - \hat{V}_l(\boldsymbol{x_0})}{\tau}$ **then**
      return *null*;
    //sampling x_i'
    $x_i' \sim \frac{b(x_i')}{B}$;
    dismiss the other parts of V_{GAP};
  return $\boldsymbol{x}' := (x_1', \ldots, x_n')$
end

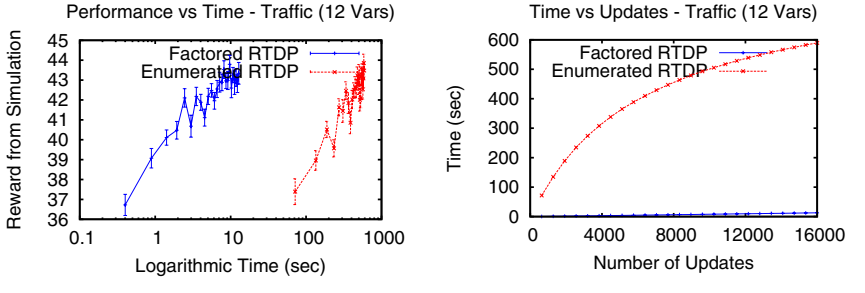


Fig. 3. Various analyses of Factored vs. Enumerated RTDP on the TRAFFIC problem

4 Empirical Results

First we focus on problems with dense transition matrices where traditional RTDP may be inappropriate (recall that dense matrices can occur in problems with a large exogenous event space). We evaluate Enumerated RTDP and sRTDP on two problems: the logistical problem SysAdmin in the uni-ring configuration with random exogenous equipment failures as defined in [11] and a Traffic MDP for signal control at a single intersection with random exogenous traffic movements [12].

We first analyse properties of the two algorithms on Traffic problem with 12 variables (2^{12} states) in Figure 3 where we show the averaged results of 10 training and policy simulation runs. In the left plot showing performance vs. a log-scale of RTDP training time, we note that sRTDP begins to reach high performance levels *long before* Enumerated RTDP has finished its initial training trials. To explain why, in the plot on the right, we see that the time per RTDP Update for Enumerated RTDP is much larger than sRTDP; Enumerated RTDP slightly speeds up on its later updates as its state cache begins to saturate and yield frequent hits, reducing overhead.

Since time-per-update is the most crucial detail indicating the relative speed of Enumerated and sRTDP, in Figure 4, we provide a time-per-update analysis over two domains varying problems with different number of state variables. Here we see that the overhead of sRTDP may make it slower than Enumerated RTDP on small problems, but as the problem size grows, sRTDP requires substantially less time. For the largest

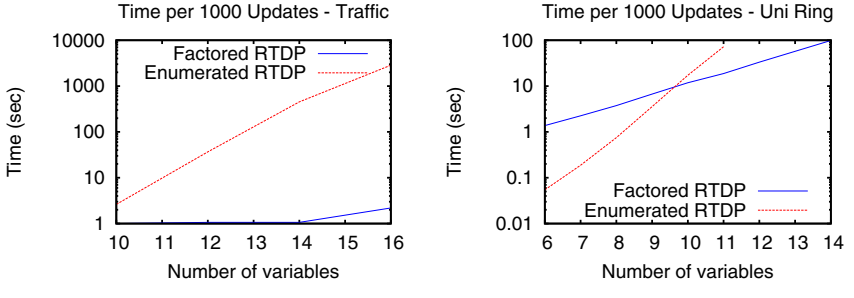


Fig. 4. Analysis of the time per RTDP update for Factored vs. Enumerated RTDP

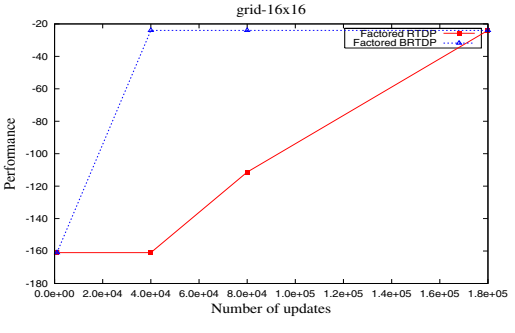


Fig. 5. Symbolic RTDP vs. Symbolic BRTDP

problems (with a large exogenous event space), Enumerated RTDP did not finish in the time limit given by the upper bound of the plots *and* given the log-scale time axis, we note an improvement up to *three orders of magnitude* on large TRAFFIC problems.

Finally we analyse the performance vs number of updates for both symbolic algorithms sRTDP and sBRTDP, for the grid problem 16x16. Figure 5 shows that the sBRTDP converges making $\sim$ 140000 less updates.

5 Concluding Remarks

While RTDP approaches have proved very popular in recent years for their ability to exploit initial state knowledge and sparse transition matrices, the advantages of traditional Enumerated RTDP are often lost on MDPs with dense transition matrices. As we showed in the experiments sRTDP can speedup learning over its enumerated state counterpart by up to *three orders of magnitude*. Motivated by the results of sRTDP [7], we have proposed a new variant of BRTDP [5] named sBRTDP which is able to converge to the optimal value function with much less updates than sRTDP. This is due to the additional information used for sampling next states. The original idea proposed in this paper is how to sample state variables in the symbolic representation using the ADD structure.

References

1. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
2. Bertsekas, D.P.: Distributed dynamic programming. IEEE Transactions on Automatic Control 27, 610–617 (1982)
3. Barto, A.G., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, U. Mass. Amherst (1993)
4. Bonet, B., Geffner, H.: Labeled RTDP: Improving the convergence of real-time dynamic programming. In: ICAPS-2003, Trento, Italy, pp. 12–21 (2003)
5. McMahan, H.B., Likhachev, M., Gordon, G.J.: Bounded real-time dynamic programming: RTDP with monotone upper bounds. In: ICML 2005, Bonn, Germany, pp. 569–576 (2005)
6. Sanner, S., Goetschalckx, R., Driessens, K., Shani, G.: Bayesian real-time dynamic programming. In: 21st IJCAI, San Francisco, CA, USA, pp. 1784–1789 (2009)
7. Feng, Z., Hansen, E.A., Zilberstein, S.: Symbolic generalization for on-line planning. In: 19th UAI, pp. 209–216 (2003)
8. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: Stochastic planning using decision diagrams. In: UAI 1999, Stockholm, pp. 279–288 (1999)
9. Bahar, R.I., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A., Somenzi, F.: Algebraic Decision Diagrams and their applications. In: IEEE /ACM International Conference on CAD, pp. 428–432 (1993)
10. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: UAI 1996, Portland, OR, pp. 115–123 (1996)
11. Guestrin, C., Koller, D., Parr, R., Venktaraman, S.: Efficient solution methods for factored MDPs. JAIR 19, 399–468 (2002)
12. Delgado, K.V., Sanner, S., de Barros, L.N., Cozman, F.G.: Efficient Solutions to Factored MDPs with Imprecise Transition Probabilities. In: 19th ICAPS, Thessaloniki, Greece (2009)

An Adaptive Genetic Algorithm to the Single Machine Scheduling Problem with Earliness and Tardiness Penalties

Fábio Fernandes Ribeiro[1], Marcone Jamilson Freitas Souza[2], and Sérgio Ricardo de Souza[1]

[1] Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG
Programa de Pós-Graduação em Modelagem Matemática e Computacional,
Avenida Amazonas 7675, Belo Horizonte, CEP: 30510-000, Minas Gerais, Brazil
fabiobh@gmail.com, sergio@dppg.cefetmg.br
http://www.mmc.cefetmg.br

[2] Universidade Federal de Ouro Preto - UFOP, Departamento de Computação, ICEB
Campus Universitário (UFOP), Ouro Preto, CEP: 35400-000, Minas Gerais, Brazil
marcone@iceb.ufop.br
http://www.iceb.ufop.br

Abstract. This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties, considering distinct due windows and sequence-dependent setup time. Due to its complexity, an adaptive genetic algorithm is proposed for solving it. Many search operators are used to explore the solution space where the choice probability for each operator depends on the success in a previous search. The initial population is generated by the combination between construct methods based on greedy, random and GRASP techniques. For each job sequence generated, a polynomial time algorithm is used for determining the processing initial optimal date to each job. During the evaluation process, the best individuals produced are added to a special group, called elite group. The individuals of this group are submitted to refinement, aiming to improve their quality. Three variations of this algorithm are submitted to computational test. The results show the effectiveness of the proposed algorithm.

1 Introduction

Scheduling problems are one of the most studied problems in combinatorial optimization [1]. It occurs mainly by two aspects: the first one concerns their practical importance, with various applications in several industrial fields. The second aspect is about the difficulty for solving the majority problems of this class. This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties (SMSPETP) with distinct due windows and sequence-dependent setup time. To our knowledge, this problem has not been still object of great attention of the scientific community, as it could seen in the recent survey [1].

The criteria to penalize the tardiness and earliness production goes to the Just-in-Time philosophy goal, that is, the production is done just when necessary.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 203–212, 2010.

The existence of a due window for each job, according to [2], is because of an uncertainly situation or tolerance related to due date. It is accepted that this time interval operations can be finalized without costs. On the other hand, in most industrial processes the machines should first be prepared for doing new jobs, including the time to obtain tools, positioning materials that will be used in the process, cleaning process, preparing process, tools adjustment and materials inspection. The necessary time to this preparation is known by setup time. Many production scheduling researches disregard this time or include it in the operation processing time. This act simplifies the analysis but affect the solution quality when the setup time has a relevant variability in function of the job sequence in machine. This work considers that the setup times are dependents from the production scheduling. Since it was showed in [3] that a simplified version of this problem is NP-Hard, the application of metaheuristics for solving this problem is justified.

In order to solve this scheduling problem with the presented characteristics, an Adaptive Genetic Algorithm, so-called AGA, is proposed here. To generate different individuals having good quality, the initial population is generated by a construction method based on GRASP [5], which uses five dispatch rules to form the individuals. During the evolution process, the population passes through mutation and crossover conventional process. However, the crossover uses criteria based on solution quality generated by each crossover operator to choose which operator will be used. By the way, according to how well an operator performs, the probabilities it be chosen is increased or decreased during the evolution. A local search is applied in the best offspring produced for each operator, to refine it. The survival population is composed by individuals chosen by elitism technique. Mutation process is then applied to a slice of the surviving population for diversifying it. Periodically, a Path Relinking module is applied taking the best one so far generated by the algorithm as base individual and each one of the five best individuals generated by each crossover operator as guide individual. The population improvement occurs until the stop criteria is reached.

The remaining of this work is organized as follows: section 2 details the studied problem; section 3 presents the adaptive algorithm for solving SMSETP; section 4 shows and discuss the results; finally, section 5 ends this work.

2 Problem Description

This work studies the single machine scheduling problem, with earliness and tardiness penalties, distinct due windows and sequence-dependent setup time. In this problem, one machine must process a set of n jobs. Each job i has processing time P_i, initial date E_i and final date T_i, desired for ending the processing. The machine executes one job per time and, if a job processing is started, it must be finished, since processing interruptions are not allowed. All jobs are available for processing in the instant 0. When a job j is sequenced immediately after a job i, for setting the machine is necessary a setup time S_{ij}. Setup times equal 0 mean products of the same family. The initial setup times are considered, i.e., the setup

time to the first job in the sequence is 0. The idle time between the execution with two consecutive jobs is allowed. The jobs must be finalized inside the time interval $[E_i, T_i]$, called due window. In case of job finalization before E_i, there is a cost to earliness. Case the job are finalized after T_i, a cost will be generated for tardiness. For jobs completed within due windows, none cost is incurred. The costs to earliness and tardiness of production depend on jobs. Each job i have a earliness cost α_i and a tardiness cost β_i. Finally, the objective of the problem is to minimize the summation of the earliness and tardiness penalties.

This scheduling problem is described by the mixed integer programming model (MIP) shown below, based on [6]:

$$\min \quad Z = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \tag{1}$$

$$\text{s.t:} \quad s_j - s_i - y_{ij}(M + S_{ij}) \geq P_i - M \quad \forall\, i = 0, \ldots, n; \quad \forall\, j = 0, \ldots, n; \quad i \neq j \tag{2}$$

$$\sum_{j=0,\ j\neq i}^{n} y_{ij} = 1 \qquad \forall\, i = 0, \ldots, n \tag{3}$$

$$\sum_{i=0,\ i\neq j}^{n} y_{ij} = 1 \qquad \forall\, j = 0, \ldots, n \tag{4}$$

$$s_i + P_i + e_i \geq E_i \qquad \forall\, i = 1, \ldots, n \tag{5}$$

$$s_i + P_i - t_i \leq T_i \qquad \forall\, i = 1, \ldots, n \tag{6}$$

$$s_i \geq 0 \qquad \forall\, i = 0, \ldots, n \tag{7}$$

$$e_i \geq 0 \qquad \forall\, i = 1, \ldots, n \tag{8}$$

$$t_i \geq 0 \qquad \forall\, i = 1, \ldots, n \tag{9}$$

$$y_{ij} \in \{0, 1\} \qquad \forall\, i, j = 0, \ldots, n \tag{10}$$

where:

- s_i: the starting time of job i;
- C_i: the completion time of job i;
- y_{ij}: binary variable that assumes value 1 if job j is processed immediately after job i and 0, otherwise;
- e_i: the earliness of job i, that is, $e_i = \max\{0, E_i - C_i\}$;
- t_i: the tardiness of job i, that is, $t_i = \max\{0, C_i - T_i\}$;
- M: a sufficiently large number;
- 0: a fictitious job, which precedes and succeeds all other jobs;

It also assumes that $P_0 = 0$, $S_{0i} = S_{i0} = 0 \ \ \forall i \in \{1, 2, \cdots, n\}$

The objective function (1) expresses the total earliness and tardiness cost. The constraints (2) establish that job j can be processed when job i is finished and the machine is prepared to processes it. The constraints (3), (4) and (10) guarantee that the variable y_{ij} assumes value 1 if and only if job j is processed immediately after job i. The constraints (5) and (6) define, respectively, the tardiness and earliness values according of the due window. The constraints (7) to (10) define the type of the variables.

3 Heuristic Framework

In this section, the adaptive genetic algorithm framework proposed for solving the problem is described.

Individual representation: An individual (i.e., a solution) is represented by a vector v of n genes (jobs). The production sequence of each job is given by the i position in the vector. For example, in the sequence $v = \{5, 2, 7, 4, 6, 3, 1\}$, the job 5 is the first to be processed and the job 1 is the last.

Evaluation of individuals: All of individuals are evaluated by the same objective function, given by expression (1) of MIP model (Mixed Integer Programming), where the individual which obtained the shortest value to the objective function is considered the most adapted.

Initial population construction: The initial population of the proposed adaptive genetic algorithm is generated by GRASP construction phase ([5]), having five dispatch rules (EDD, TDD, SPT, WSPT and LPT) as guide function. For each construction (GRASP + dispatch rule), 200 individuals are generated. In the sequel, the individuals are ordered according to evaluation function, from the best one to the worst one. The initial population is composed by the 100 best generated individuals.

GRASP construction procedure: In this construction procedure, an offspring is formed by genes that are inserted one by one. The offspring is constructed according with a partially greedy selection criteria. To estimate the insertion benefit of each gene, dispatch rules EDD, TDD, SPT, WSPT and LPT are used. Each variant gives a different construction. In Figure 1, the GRASP construction phase is showed. In this figure, $g_{\min}$ represents the best evaluation according to the selected dispatch rule and g_{max}, the worst one.

procedure *Construction*$(g(.), \gamma, v)$;
1 $v \leftarrow \emptyset$;
2 Initialize a set C of candidate genes;
3 while $(C \neq \emptyset)$ do
4 $g_{\min} = \min\{g(t) \mid t \in C\}$;
5 $g_{max} = \max\{g(t) \mid t \in C\}$;
6 $RCL = \{t \in C \mid g(t) \leq g_{min} + \gamma \times (g_{\max} - g_{\min})\}$;
7 Select, randomly, a gene $t \in RCL$;
8 $v \leftarrow v \cup \{t\}$;
9 Update C;
10 end-while;
11 Return v;
end *Construction*;

Fig. 1. Procedure to build an individual

```
Algorithm AGA(germax, nind, probcross, probmut, freq);
1  t ← 0;
2  Generate Initial Population P(t);
3  Evaluate P(t);
4  while (t < germax) do
5      t ← t + 1;
6      P(t) ← P(t − 1);
7      i ← 0; { number of new individuals }
8      while (i < nind) do
9          Select two individuals from P(t − 1);
10         cross ← Randomly number from 1 to 100;
11         if (cross ≤ probcross ) then
12             Choose a crossover operator O_k;
13             Apply the chosen crossover operator;
14             i ← i + 2;
15             Incorporate the new individuals to P(t);
16         end-if;
17     end-while;
18     Evaluate P(t);
19     Define nind survivors;
20     Apply mutation with probability probmut in all members of population P(t)
21     if (t mod freq = 0) then
22         Update the probability of selecting each crossover operator (p_(O_k));
23         Execute Local Search;
24         Apply Path Relinking;
25     end-if;
26 end-while;
end AGA;
```

Fig. 2. Pseudocode of the proposed Adaptive Genetic Algorithm

3.1 Adaptive Genetic Algorithm Applied to SMSETP

Figure 2 shows the pseudocode of the proposed Adaptive Genetic Algorithm (AGA). The algorithm phases are described in the following.

Individual selection method: The individuals are selected to reproduction (line 9 of Figure 2) by the tournament selection. In the implemented strategy, only one tournament involving all the population is realized. The winner of the tournament, that is, the one with the best fitness, is selected for crossover with each individual from the population. Therefore, all the individuals are selected to reproduction.

Crossover: The crossover process uses the following operators: (*i*) One Point Crossover (OX), (*ii*) Similar Job Order Crossover (SJOX), (*iii*) Relative Job Order Crossover (RRX), (*iv*) Based Order Uniform Crossover (BOUX) and (*v*) Partially Mapped Crossover (PMX). This choice was taken by the fact of this

operators being the most common operators to solve problems like this by genetic algorithm [4]. The choice probability of crossover operators modifies according to the quality of individuals produced by the operators in the past generations. More specifically, let O_k, with $k = 1, \cdots, 5$, be the five crossover operators. Initially, each crossover operator O_k has the same probability to be chosen, that means, $p(O_k) = 1/5$. Let $f(s^*)$ be the best individual so far and A_k the average value found for each operator O_k since the last update. Case the operator was not chosen in last five generations, make $A_k = 0$. Then, calculate $q_k = f(s^*)/A_k$ and $p(O_k) = q_k / \sum_{j=1}^{5} q_j$ for all $k = 1, \cdots, 5$. Observe that how much better the individual is, more high is the value of q_k and, consequently, the probability of choosing the O_k operator is increased. Therefore, during the algorithm evolution, the best operator have its chance of choice increased. This procedure is inspired in *Reactive GRASP* algorithm, proposed by [7].

Local search: Like said previously, at each five generations, a local search is applied to the best individual generated by each *crossover* operator. The local search used method is Random descent. This method uses two kinds of movement to explore the search space: the change of two jobs of the the sequence and the job relocation to another production sequence. The method works as follows: to an individual, two jobs are selected randomly and the positions are exchanged. If the new individual is better than the current one, according to the evaluation function, it is accepted and becomes the current solution; otherwise, another movement are randomly chosen. If during *MRDmax* iterations any solution better than the current one are generated, then relocate movements are used. If there is any improvement in this phase, the method returns to use exchange movements; otherwise, the local search is ended up after *MRDmax* iterations without improvement.

Path Relinking: During the evolutive process, a group with the best five individuals generated by each crossover operator are built. At each five generations, Path Relinking (PR) procedure is applied, taking as base solution the best individual generated by the method and as guide individual each one of the five best individuals generated by each crossover operator. The PR is interrupted when 75% of the guide individual is added to the base solution. Besides it, the base solution is better than the guide solution. So, it is so-called Truncated Backward Path Relinking. A job position of production sequence is considered as an attribute. For each job candidate to insertion, a local search method like described previously is applied, and a movement of a fixed job is not allowed.

Surviving individuals: The surviving individuals are chosen by the elitism technique. So, the individuals more adapted survive.

Stop criteria: The maximum number of generations is used as a stop criteria of the adaptive genetic algorithm.

3.2 Variants of Proposed Algorithm

In this work, three variants of AGA are shown. In Variant 1, called AGA1, the refresh of crossover operator selection rate (*freq* = 5 in Fig. 2) happens at

each five generations. After that, all elite group members are submitted to local search (see local search paragraph) and, in the sequel, they are submitted to path relinking procedure. In this variant, the elite group are composed by the best individuals produced by each crossover operator in last five generations.

Variant 2, called AGA2, differs to AGA1 variant by elite group composition. In variant 2, elite group are composed by the best individual produced by each crossover operators globally and not just at last five generations.

In Variant 3, called AGA3, the refresh of crossover operator selection rate, the elite group submission to local search and the submission to path relinking procedure happens at each ten generations ($freq = 10$ in Fig. 2). The elite group are composed by the best three individuals globally produced, by the best solution produced at past ten generation, if this individual have diversity index upper than 30% for another individuals in elite group. If the individual does not fit in this criteria, the second best solution are analyzed and so on until the one of them satisfied this criteria is found. The fifth element is chosen by selecting at random an individual of a set of the best ten individuals produced over past ten generations. To compute the diversity index of two individuals, the number of different genes in a same position is summed and this value is divided by the total number of genes of the individual.

4 Computational Experiments

The proposed algorithm was developed in C++ language, using Borland C++ Builder 5.0 compiler. The used parameters were obtained experimentally and they are presented in Table 1.

Table 1. Adaptive Genetic Algorithm parameters

Parameters	Values
Parameter γ GRASP construction phase	0.20
Maximum iterations of local search ($MRDmax$)	$7 \times n$
Maximum generations of AGA ($germax$)	100
Crossover probability ($probcross$)	80%
Mutation rate ($probmut$)	5%

Two instances were used to test the three variants of AGA. The first one is the same of [6]. These authors generated instances randomly with job number equal to 6, 7, 8, 9, 10, 11, 12, 15, 20, 25, 30, 35, 40, 50 and 75, using the same parameters of [2], [8] and [9]. The second one was generated by [11] with jobs numbers equal to 6, 7, 8, 9, 10, 11, 12, 15, 20, 25, 30, 35, 40, 50, 75 and 100. These instances are available for download at `http://www.iceb.ufop.br/decom/prof/marcone/projects/scheduling/instances.htm`.

All experiments were realized in a Pentium Core 2 Duo 2.1 GHz computer with 4 GB RAM and Windows Vista operational system. Two sets of experiments were realized. The description, details and results of each one of these experiments are described in the following.

Table 2. Results of the first set of experiments - BAT 1

#	Average result *deviation*			Best result *deviation*			Time (s)		
Jobs	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.94	0.93	0.70
9	0.15%	0.16%	0.24%	0.00%	0.00%	0.00%	1.26	1.25	0.78
10	0.24%	0.25%	0.41%	0.00%	0.00%	0.00%	1.6	1.59	0.99
11	0.03%	0.05%	0.10%	0.00%	0.00%	0.00%	2.21	2.20	1.64
12	0.07%	0.08%	0.21%	0.00%	0.00%	0.00%	2.81	2.80	2.44
15	0.76%	0.80%	1.16%	0.00%	0.00%	0.00%	6.02	5.98	6.09
20	0.73%	0.75%	0.85%	0.00%	0.00%	0.00%	20.6	20.47	17.87
25	1.02%	1.08%	1.42%	0.00%	0.00%	0.00%	45.72	45.44	39.65
30	1.60%	1.82%	2.64%	0.00%	0.00%	0.00%	112.06	111.38	41.65
40	2.33%	2.34%	3.56%	0.08%	-0.08%	-0.09%	335.88	333.81	41.61
50	4.06%	4.37%	6.32%	0.02%	-0.31%	-1.11%	896.1	890.60	222.04
75	6.52%	9.48%	11.86%	0.04%	-4.40%	-1.76%	2000.05	1992.73	1242.06
Avg	**1.46%**	**1.77%**	**2.40%**	**0.011%**	**-0.399%**	**-0.246%**	**285.85**	**284.10**	**134.79**

Table 3. Comparing AGA 1 × GTPRS, proposed by [12]

#	Average result *deviation*			Best result *deviation*			Time (s)	
Jobs	AGA 1	GTPRS	% Improv.	AGA 1	GTPRS	% Improv.	AGA 1	GTPRS
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.94	0.06
9	0.15%	0.00%	-15.00%	0.00%	0.00%	0.00%	1.26	0.09
10	0.24%	0.00%	-24.00%	0.00%	0.00%	0.00%	1.60	0.15
11	0.03%	0.00%	-3.00%	0.00%	0.00%	0.00%	2.21	0.25
12	0.07%	0.00%	-7.00%	0.00%	0.00%	0.00%	2.81	0.36
15	0.76%	1.25%	64.08%	0.00%	0.00%	0.00%	6.02	0.46
20	0.73%	1.11%	51.87%	0.00%	0.00%	0.00%	20.60	2.05
25	1.02%	1.60%	56.53%	0.00%	0.00%	0.00%	45.72	6.62
30	1.60%	2.57%	60.61%	0.00%	0.00%	0.00%	112.06	18.66
40	2.33%	3.77%	61.84%	0.08%	0.00%	8.00%	335.88	84.16
50	4.06%	5.58%	37.64%	0.02%	0.00%	2.00%	896.10	305.28
75	6.52%	7.41%	13.69%	0.04%	0.00%	4.00%	2005.05	3.472.26
Avg	**1.46%**	**2.01%**	**2.27%**	**0.011%**	**-0.40%**	**-0.25%**	**285.85**	**324.20**

4.1 The First Set of Experiments - BAT 1

The first set of experiments uses the first instance of problems. Each set of problems was tested 30 times for AGA1, AGA2 and AGA3, the variants of AGA method. Table 2 shows the results reached in this set of experiments. The first column shows the number of jobs; the second, third and fourth columns show how much the average of solutions of each variant are diverted from the best solution known. In the fifth, sixth and seventh columns are showed how much the best generated solutions are diverted from the best solution known. In the eighth, ninth and tenth columns the computational time average of applying the variants of AGA are showed.

Table 3 compares AGA1 with the algorithm GTSPR, proposed by [12]. In this table, "% Improv." indicates how much AGA1 improves the solutions produced by GTSPR with relation to the average deviation (or to the best deviation).

4.2 The Second Set of Experiments - BAT 2

The second set of experiments uses the second instance of problems. Each set of problems was tested 30 times for each variant of AGA method. Table 4 shows the

Table 4. Results of the second set of experiments - BAT 2

#	Average result *deviation*			Best result *Deviation*			Time (s)		
Jobs	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3
6	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.76	0.61	0.65
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.76	0.94	0.81
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.03	1.21	1.00
9	0.00%	0.13%	0.20%	0.00%	0.13%	0.20%	1.35	1.53	1.26
10	0.00%	0.03%	0.06%	0.00%	0.03%	0.06%	1.61	1.88	1.65
11	0.00%	0.11%	0.17%	5.97%	0.11%	6.16%	2.20	0.73	1.92
12	0.00%	0.01%	0.11%	0.00%	0.01%	0.11%	3.32	3.09	2.37
15	0.01%	0.14%	0.37%	0.01%	0.14%	0.37%	7.49	5.60	4.38
20	0.30%	0.64%	0.81%	0.30%	0.64%	0.81%	23.88	10.18	10.61
30	1.20%	1.19%	1.35%	1.36%	1.51%	1.50%	172.51	64.19	47.13
40	0.98%	1.21%	0.98%	1.03%	1.42%	2.34%	801.67	155.05	98.05
50	1.14%	1.46%	1.14%	1.26%	2.24%	2.58%	1575.11	529.11	416.78
75	0.00%	1.26%	2.36%	0.09%	1.67%	3.60%	4978.02	3381.58	1398.82
100	0.25%	2.50%	1.10%	0.25%	3.14%	2.41%	18107.72	23209.42	15853.97
Avg	**0.28%**	**0.62%**	**0.62%**	**0.73%**	**0.79%**	**1.44%**	**1834.10**	**1954.82**	**1274.24**

results reached in the first set of experiments. The first column shows the number of jobs. The second, third and fourth columns show how much the average of solutions of each variant are diverted from the best solution known. In the fifth, sixth and seventh columns are showed how much the best generated solutions are diverted from the best solution known. In the eighth, ninth and tenth columns the computational time average of applying the variants of AGA are showed.

5 Conclusions

This paper dealt with the single machine scheduling problem with earliness and tardiness penalties, considering distinct due windows and sequence-dependent setup time. To solve this problem, an adaptive genetic algorithm, named AGA, was proposed. The initial population was generated by a GRASP procedure using dispatch rules as guide functions. During the evolution process, population is submitted to selection, crossover and mutation processes. In crossover process, five operators are used, being that the best solutions produced by each operator are submitted to local search and path relinking. The path relinking procedure connects the best solution reached so far with the best solutions produced by each operator.

By the end, two set of instances were used to test the proposed algorithm, and three variants of AGA were developed. The results of each instance were compared with another algorithm from the literature. In these experiments, the proposed algorithm presented high quality solutions with lower gap, always reaching the best known value. The developed algorithm presented solutions better than the best solutions found in the literature, besides presenting a minor variability of final solutions.

Acknowledgment

The authors would like to thank CEFET/MG, CAPES, FAPEMIG and FAPERJ for the support to development of this work.

References

1. Allahverdi, A., Ng, C., Cheng, T.C.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research 187, 985–1032 (2008)
2. Wan, G., Yen, B.P.C.: Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. European Journal of Operational Research 142, 271–281 (2002)
3. Du, J., Leung, J.Y.T.: Minimizing total tardiness on one machine is np-hard. Mathematics of Operations Research 15, 483–495 (1990)
4. Lee, C.Y., Choi, J.Y.: A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. Computers and Operations Research 22, 857–869 (1995)
5. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
6. Gomes Jr., A.C., Carvalho, C.R.V., Munhoz, P.L.A., Souza, M.J.F.: Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. In: Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional - XXXIX SBPO, Fortaleza, Brazil, pp. 1649–1660 (2007)
7. Prais, M., Ribeiro, C.C.: An application to a matrix decomposition problem in tdma traffic assignmen. INFORMS - Journal on Computing 12, 164–176 (2000)
8. Liaw, C.F.: A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. Computers and Operations Research 26, 679–693 (1999)
9. Mazzini, R., Armentano, V.A.: A heuristic for single machine scheduling with early and tardy costs. European Journal of Operational Research 128, 129–146 (2001)
10. Rabadi, G., Mollaghasemi, M., Anagnostopoulos, G.C.: A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. Computers and Operations Research 31, 1727–1751 (2004)
11. Rosa, B.F., Souza, M.J.F., Souza, S.R.: Formulações de programação matemática para o problema de sequenciamento em uma máquina com janelas de entrega distintas e tempo de preparação dependentes da sequência de produção. In: Anais do XXXII Congresso Nacional de Matemática Aplicada e Computacional – CNMAC 2009, Cuiabá (2009)
12. Penna, P.H.V.: Um algoritmo heurístico híbrido para minimizar os custos com a antecipação e o atraso da produção em ambientes com janelas de entrega e tempos de preparação dependentes da sequência de produção. In: Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto (2009)

A Dijkstra Algorithm for Fixed-Wing UAV Motion Planning Based on Terrain Elevation

Felipe Leonardo Lôbo Medeiros and José Demisio Simões da Silva

Instituto de Estudos Avançados, Rod. dos Tamoios, km 5,5, Putim, CEP 12.228-001, São José dos Campos, São Paulo, Brasil
felipe@ieav.cta.br
Instituto Nacional de Pesquisas Espaciais, Av. dos Astronautas 1.758, Jardim da Granja, CEP 12227-010, São José dos Campos, São Paulo, Brasil
demisio@lac.inpe.br

Abstract. Abstract. The automatic motion or trajectory planning is essential for several tasks that lead to the autonomy increase of Unmanned Aerial Vehicles (UAVs). This work proposes a Dijkstra algorithm for fixed-wing UAVs trajectory planning. The navigation environments are represented by sets of visibility graphs constructed through the terrain elevations of these environments. Digital elevation models are used to represent the terrain elevations. A heuristics to verify if a trajectory is collision-free is also proposed in this work. This heuristics is a method of grid-based local search which presents linear computational time $O(n_p)$, where n_p is the number of verification steps. This heuristics is compared with another method for collision verification. Results are presented in this work.

Keywords: fixed-wing UAV; motion planning; Dijkstra algorithm; digital elevation model; grid-based local search.

1 Introduction

The automatic motion or trajectory planning is essential for several tasks that lead to the autonomy increase of Unmanned Aerial Vehicles (UAVs). This work approaches the problem of trajectory planning for fixed-wing UAVs navigating to constant altitudes. This problem is similar to the trajectory planning for nonholonomic wheeled mobile robots [1]. A trajectory is a sequence of waypoints connected by straight line segments and arcs that allow to the fixed-wing UAV navigates between an initial and a final waypoint of the navigation environment. A trajectory must be collision-free and dynamically feasible [1], i. e., a trajectory must allow the UAV navigate from a waypoint to the next waypoint without violating the UAV dynamics and kinematic constraints. A formal definition of dynamically feasible trajectory is presented in [1]. Fig. 1 presents an example of a dynamically feasible collision-free trajectory planned for a fixed-wing UAV.

A recent revision of UAV motion planning methods is presented in [2]. As mentioned in this revision, Dijkstra algorithms assure optimality when used to plan navigation paths, which are sequences of waypoints connected by straight line segments. However, the paths planned by these algorithms must be transformed in trajectories

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 213–222, 2010.

through the application of smoothing methods. A Dijkstra algorithm is proposed in [3] to solve the single-destination UAV shortest trajectory problem based on visibility graphs, without the necessity of application of smoothing methods. This algorithm is a modification of the original Dijkstra algorithm [4] used for the single-source shortest path problem. The Modified Dijkstra Algorithm (MDA) uses rectangular representations of the non-navigable regions of the navigation environment to create the collision-free nodes and edges of the visibility graphs. The rectangular representation is also used by a method to check a trajectory is collision-free. In the worst case, the computational time of this method is $O(n_o)$, where n_o is the number of obstacles or non-navigation regions of the navigation environment. Due to this computational time, the MDA is not efficient for navigation environments with a large number of obstacles as, for example, the navigation environments with surfaces represented by digital elevation models [5]. A digital elevation model is a computational representation of the terrain elevation of a navigation environment.

Therefore, this work proposes an Elevation-based Dijkstra Algorithm (EDA) that is an adaptation of the MDA [3] aiming at the planning of dynamically feasible collision-free trajectories through visibility graphs and digital elevation models.

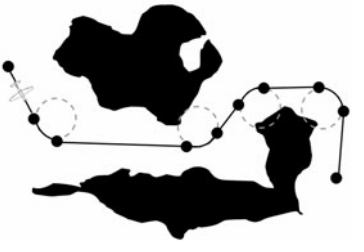


Fig. 1. Example of a planned trajectory. The planned trajectory is represented by *circles*, *straight line segments* and *arcs*. The *circles* indicate the waypoints of the trajectory. The *black polygons* represent the non-navigable regions of the navigation environment. The *stippled gray circles* indicate the circles defined by the UAV turning radius.

EDA presents three main distinctions regarding the MDA. The first distinction is the elaboration of the proposed algorithm to solve the single-pair shortest trajectory problem. As second distinction, EDA plans trajectories of the class k-trajectories proposed in [1]. A k-trajectory is a Dubins curve whose length and form can be adjusted by the parameter k. The last distinction is that this work also proposes a grid-based local search heuristics to verify if a trajectory is collision-free. This heuristics is invariant to the number of obstacles and presents a computational time of $O(n_p)$, in the worst case, where n_p is the number of verifications. Considering navigation environments with a large number of obstacles, in comparison with the method used in the modified Dijkstra algorithm, this heuristics allows a significant decrease of the computational time, becoming the main contribution of this work.

The remainder of this paper is organized as follows. Section 2 presents the construction of visibility graphs which represent the navigation environment used in this work. Section 3 presents the Dijkstra algorithm proposed in this work and presents the results of the application of this algorithm. Final conclusions based on the obtained results are presented in Section 4.

2 Visibility Graphs

A visibility graph is a set of nodes defined by the convex vertexes of polygons and connected by edges that do not intersect such polygons [6]. In path planning problems, visibility graphs are used to represent the navigable regions of navigation environments and allow the planning of the shortest paths between two waypoints. Thus, the nodes of a visibility graph are waypoints and the polygons are the obstacles or non-navigable regions of the navigation environment. Fig 2. presents an example of visibility graph.

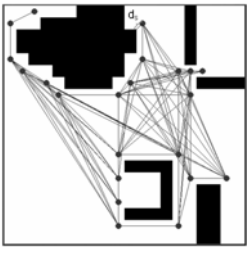


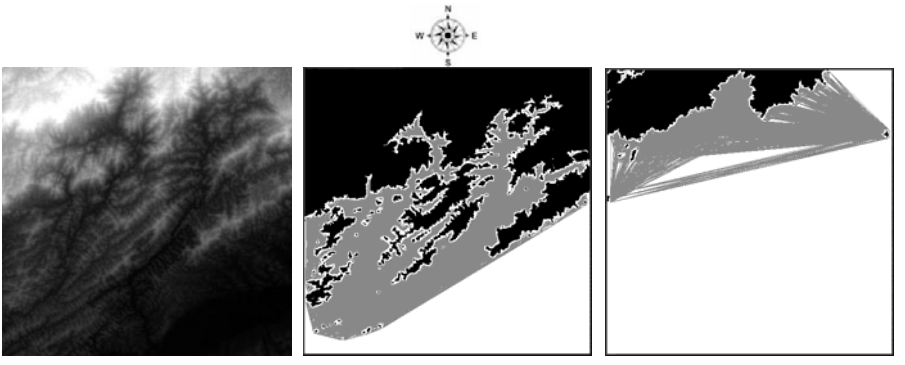
Fig. 2. Example of a visibility graph. The *black polygons* represent the non-navigable regions. The *gray circles* and the *gray straight line segments* represent respectively the nodes and edges of the visibility graph.

In this work, a navigation environment can be understood as a surface with a constant altitude a_n and divided in a regular grid B. Each region $b_{lc} \in B$ corresponds to a cell e_{lc} of the elevation grid E of the respective digital elevation model. The non-navigable regions b_{lc} are defined by the cells e_{lc} with elevation superior or equal to $a_n - a_s$., where: a_n is the navigation altitude; and a_s is a height of safety.

A single navigation environment is represented by a set of visibility graphs. The nodes and edges of the visibility graphs are determined by the application of two algorithms proposed in [5] which construct a visibility graph directly from a digital elevation model, considering: a constant altitude of navigation a_n; a constant height of safety a_s; and a constant distance of safety d_s.

Each node of a visibility graph constructed by the first of the algorithms proposed in [5] is a geographic coordinate which belongs to the axis that intersects the extremity and the central coordinate of a non-navigable region b_{lc}. The distance between a node and the extremity that defines this node is the distance of safety d_s. The second algorithm connects the nodes through edges that do not intersect the non-navigable regions and present a distance equal to d_s of these regions.

These algorithms were applied to the construction of visibility graphs based on the digital elevation model presented in Fig. 3a. The datum and projection of this digital elevation model are, respectively, the WGS-84 and the geographic projection. The resolution of this model is r_s = 30 meters (m), i. e., each cell e_{lc} represents a region with height and width equal to 30 m. The elevation grid E is a square matrix with order 1205. Two results of the application of these algorithms are presented in Fig. 3b



(a) (b) (c)

Fig. 3. (**a**) Digital elevation model used in this work. The *clear areas* represent the areas with larger elevation. Sets of visibility graphs constructed by the mentioned algorithms: (**b**) a_n = 1100 m; and (**c**) a_n = 1700 m. The *black polygons* indicate the non-navigable regions. The *gray circles* and the *gray straight line segments* represent, respectively, the nodes and edges of the sets of visibility graphs.

Table 1. Features of the obtained sets of visibility graphs

Sets of Visibility graphs	n_n	n_e	n_o
Fig. 3b	1489 nodes	65454 edges	724548 non-navigable cells
Fig. 3c	243 nodes	9540 edges	159478 non-navigable cells

and Fig. 3c, considering the height of safety a_s = 300 m and the distance of safety d_s = 100 m. Features of the obtained sets of visibility graphs are presented in Table 1. It is used the notation n_o for the total number of non-navigable cells b_{lc}.

3 Dijkstra Algorithm Based on Terrain Elevation

As mentioned previously, this work proposes an EDA to solve the single-pair shortest trajectory problem for fixed-wing UAVs. The algorithm is based on visibility graphs and digital elevation models of the navigation environments.

In [1] is proposed the class of *k*-trajectories based on the kinematic constraints of an UAV. An example of *k*-trajectory is presented in Fig. 4a. Given a path composed by the waypoints w_{i-1}, w_i and w_{i+1}, a *k*-trajectory is the unique, dynamically feasible, minimum-time trajectory which allows the navigation from the waypoint w_i to the waypoint w_{i+1}, passing directly through $p(k)$ [1].

As proven in [1], for k = 1, the trajectory is the shortest *k*-trajectory. Due to this fact, the proposed EDA plans *k*-trajectories, for k = 1 (1-trajectory), as presented in Fig. 4b. However, the EDA can be adapted simply for others k values. The proposed EDA is presented in Table 2.

This way, analyzing the Fig. 4b, a 1-trajectory planned by the EDA through a visibility graph is composed by: a straight line segment $\overline{w_1 w_{2e}}$; an arc from w_{2e} to w_{2s}; a sequence of a straight line segments $\overline{w_{(i-1)s} w_{ie}}$ connected by arcs from w_{ie} to w_{is}; and a straight line segment $\overline{w_{(n_t-1)s} w_{n_t}}$, for i varying from 3 to n_t - 1, where n_t is the number of elements of the trajectory. In Fig. 4, $\vec{q}$ is an unit vector defined by the unit vectors $\vec{q}_{i-1}$ and $\vec{q}_{i+1}$.

The waypoints w_1 and w_{n_t} are the initial waypoint w_s and the final waypoint w_d, respectively. These waypoints can be any coordinate of the navigable regions and they are inserted in the set of visibility graphs which represents the navigation environment. The waypoints w_l are nodes of the visibility graph, for l varying from 1 to n_n. The waypoints w_{ie} and w_{is} are, respectively, the input waypoint and the output waypoint of the curve defined by the UAV turning radius.

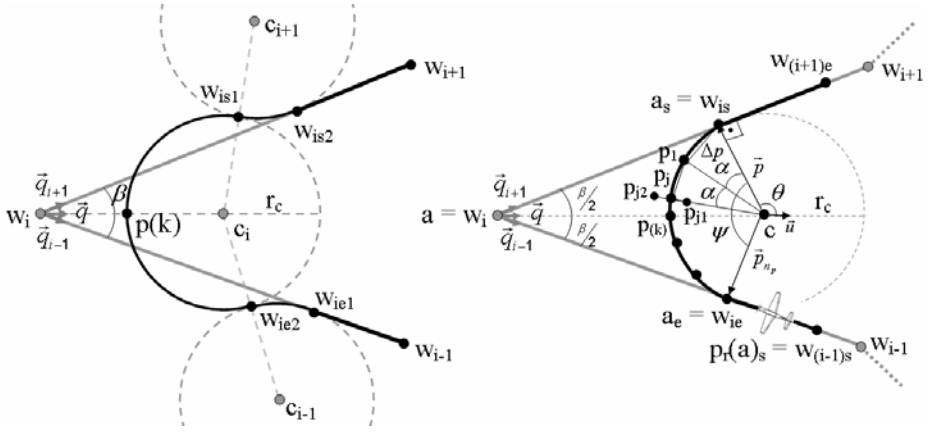


Fig. 4. (**a**) Example of a *k*-trajectory from w_{i-1} to w_{i+1}. (**b**) Discretization of the arc from w_{ie} to w_{is} of a 1-trajectory planned by EDA. The trajectories are represented by *black straight line segments*, *black arcs* and *black circles*.

Due to the this UAV turning radius, the waypoints w_{ie} and w_{is} must comply with the constraints given by Eq. 1 and Eq. 2. The planned 1-trajectory is dynamically feasible, if and only if the Eq. 1 and the Eq. 2 be not violated for i varying from 1 to n_t. These constraints are assured by the line 10 of the EDA, as presented in Table 2.

$$d(w_i, w_{ie}) \leq d(w_i, w_{(i-1)s}), \forall i . \tag{1}$$

$$d(w_i, w_{is}) \leq d(w_i, w_{(i+1)e}), \forall i . \tag{2}$$

Where: d is the distance between two waypoints.

Table 2. Elevation-based Dijkstra Algorithm. Where: *d'* is the distance between the initial node w_s and the analyzed node; p_r is the previous node of the analyzed node; r_c is the UAV turning radius; and h_{cs} is the heuristics to verify if an arc from a_e to a_s is collision-free.

Index	**Elevation-based Dijkstra Algorithm (EDA)**
1	For i varying from 1 to n_n do
2	store w_i in Q
3	$d'(w_i) \leftarrow \begin{cases} 0, para\ w_i = w_s \\ \infty, para\ w_i \neq w_s \end{cases}$
4	$a \leftarrow w_s$
5	while $a \neq w_d$ do
6	remove a from Q
7	for each neighbor v of a do
8	$f \leftarrow d'(a) + d(a,v)$
9	determine a_e, a_s, $p_r(a)_s$ through $p_r(p_r(a))$, $p_r(a)$, a and v
10	if $d(a,a_s) > d(a,v_e)$ or $d(a,a_e) > d(a,p_r(a)_s)$ do
11	$f \leftarrow \infty$
12	else
13	if $h_{cs}(a_e, a_s, E) = 1$ do
14	$f \leftarrow \infty$
15	if $f < d(v)$ do
16	$d'(v) \leftarrow f$
17	$p_r(v) \leftarrow a$
18	$a \leftarrow min\ (d(w_i))$, for $w_i \in Q$
19	while $a \neq w_s$ do
20	store a in the stack T
21	$a \leftarrow p_r(a)$

Each $\overline{w_{(i-1)s} w_{ie}}$ is a straight line segment of the edge that connect the nodes w_{i-1} and w_i of the visibility graph. This fact assures that the straight line segments of a trajectory are always collision-free, as explained in the previous section.

However, the visibility graph does not assure that the arcs are collision-free. Therefore, it is necessary the use of a method to verify if the arcs of a trajectory are collision-free. As mentioned previously, this work proposes a heuristics h_{cs} to verify collision situations in the arcs. The line 13 of the EDA is the application of this heuristics.

Section 3.1 presents the heuristics proposed in this work to verify collision situations. Section 3.2 presents results of the application of the proposed EDA.

3.1 Grid-Based Local Search Heuristics

The heuristics proposed in this work verifies if a fixed-wing UAV navigating through an arc intercepts some non-navigable cell $e_{lc} \geq (a_n - a_s)$ through a local search in E. As presented in Fig. 4b, the heuristics discretizes the arc from a_e to a_s in a sequence of points p_j, for j varying from 1 to $n_p = \psi / \alpha$. Analyzing the Fig. 5a, at each iteration j,

the heuristic verifies if the polygon that contains the possible trajectory of the UAV between p_{j-1} and p_j intercepts some non-navigable cell b_{lc}. The heuristics is presented in Table 3.

Table 3. Heuristics to verify if an arc from a_e to a_s is collision-free

Index	**Grid-based Local Search Heuristics (h_{cs})**
1	calculate n_p
2	*collision* ← 0
3	$j \leftarrow 1$
4	while $j < n_p$ and *collision* = 0 do
5	calculate p_j, p_{j1}, p_{j2} and p'_j
6	through the proposed theorem, if the polygon defined by $\overline{p_{(j-1)1}p_{(j-1)2}}$, $\overline{p_{(j-1)1}p_{j1}}$, $\overline{p_{(j-1)2}p'_j}$, $\overline{p_{j1}p_{j2}}$ and $\overline{p_{j2}p'_j}$ intercepts some non-navigable cell b_{lc} do
7	*collision* ← 1
8	else
9	$j \leftarrow j + 1$
10	return *collision*

Each point p_j is defined by Eq. 3.

$$p_j = \begin{cases} c + \begin{bmatrix} r_c \cos(\theta + j\alpha) \\ r_c \sin(\theta + j\alpha) \end{bmatrix}, \text{if } c, p \text{ and } w_i \text{ is counterclockwise} \\ c + \begin{bmatrix} r_c \cos(\theta - j\alpha) \\ r_c \sin(\theta - j\alpha) \end{bmatrix}, \text{if } c, p \text{ e } w_i \text{ is clockwise} \end{cases}. \quad (3)$$

The angle α is associated with the distance Δp between p_j and p_{j+1}. The distance Δp is based on the distance between two robot states or positions presented in [7] to verify collision situations. The problem of this approach is to assure that any collision is detected by the discretization. This work proposes a theorem that assures that the discretization of an arc in n_p points p_j spaced by Δp always allows the collision verification. The theorem is also valid for the straight line segments that define the pentagon of collision verification.

Theorem: given: a navigation surface *B* and the respective elevation grid *E*; the wingspan $\xi \leq \Delta p$ and the length $\zeta \leq \Delta p$ of the UAV; and a discretization of an arc in n_p points p_j, a straight line segment $\overline{p_{j-1}p_j}$ with length $\Delta p < r_s$ is collision-free if: the cells $b_{l_{j-1}c_{j-1}}$ and $b_{l_jc_j}$ that contain respectively p_{j-1} and p_j are navigable; and the cell $b_{l'c'}$ intercepted by $\overline{p_{j-1}p_j}$ is navigable, when $l_j \neq l_{j-1}$ and $c_j \neq c_{j-1}$, as presented in Fig. 5b.

Proof: if $\Delta p < r_s$ then the point p_j belongs to a cell $b_{l_jc_j}$ of the neighboring of the cell $b_{l_{j-1}c_{j-1}}$, where $|l_j - l_{j-1}| \leq 1$ and $|c_j - c_{j-1}| \leq 1$. Therefore, the segment $\overline{p_{j-1}p_j}$ only intercepts a cell $b_{l'c'}$ when $l_j \neq l_{j-1}$ and $c_j \neq c_{j-1}$. Due to this fact, $\overline{p_{j-1}p_j}$ does not intercept any other cell b_{lc}.

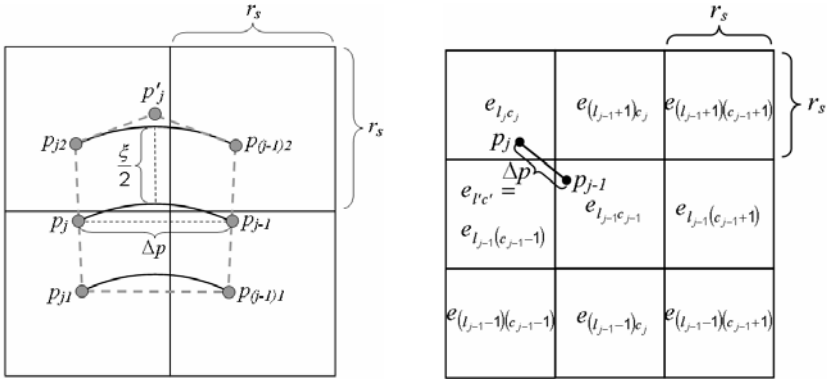


Fig. 5. (**a**) Verification of collision through a polygon that contains the trajectory of the UAV between the waypoints p_{j-1} and p_j. The *black arcs* represent the trajectory of the UAV. The polygon is represented by the *gray stippled pentagon*. The *black cells* represent a set of cells b_{lc}. (**b**) Local search in the neighboring of a cell $e_{l_{j-1}c_{j-1}}$ of the grid *E*.

This way, the angle α is defined by Eq. 4 to assure the distance Δp.

$$\alpha = 2\,arcsin\left(\frac{\Delta p}{2r_c}\right). \tag{4}$$

When an UAV navigates through an arc, the projection of the wingspan ξ of the UAV is equal to $\xi\,sin(90^o - \omega)$, where ω is the lift angle of the UAV. However, as presented in Fig. 5a, it was considered that this projection, distance between p_{j1} and p_{j2}, is equal to ξ, as a way to increase the safety of the planned trajectories.

3.2 Application of the Proposed EDA

EDA was applied to the trajectory planning considering the sets of visibility graphs obtained in Section 2.

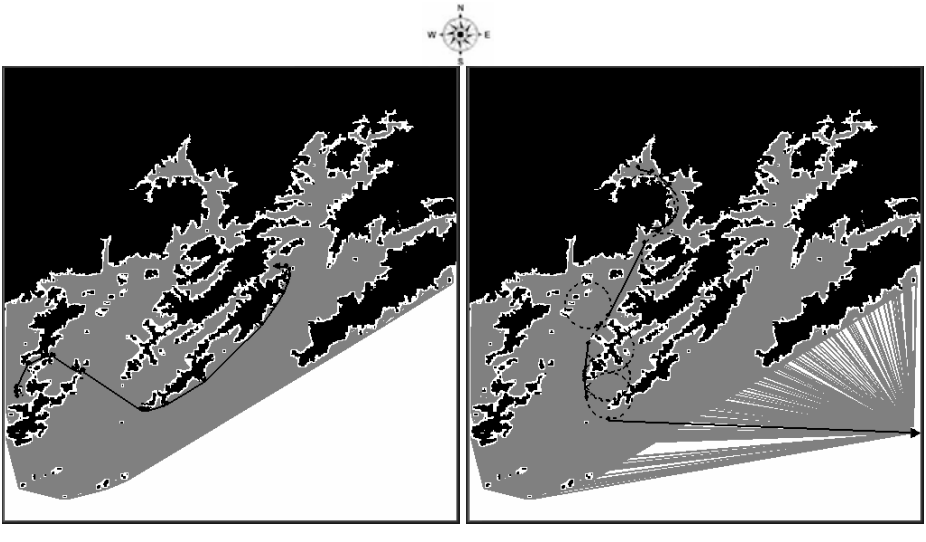
Initially, a set of specifications was considered to apply the EDA. Analyzing the Eq. 4, it can be observed that n_p decreases when the value of Δp increases. Due to this fact, the value of Δp is specified to be equal to $0.99r_s$ in the application of the algorithm. Analyzing the Fig. 4b, it can be observed that $0^o < \psi < 180^o$. Then, it was considered that the worst case for n_p occurs when: ψ is equal to 179.99^o; and to each iteration *j* of the heuristics, three cells are evaluated for each straight line segment of the pentagon. Therefore, in the worst case, the number of cells verified by the heuristic is $n_c = 12n_p = 12(179.99/2\,arcsin(0.99r_s/2r_c))$, considering that the straight line segment $\overline{p_{(j-1)1}p_{(j-1)2}}$ is evaluated in the previous iteration.

Table 4. Comparison between the methods to verify collision situations in an arc

Sets of Visibility graphs	proposed heuristics $O(n_p)$	method used in MDA $O(n_o)$
Fig. 3b	n_p = 32 iterations, n_c = 384 cells	n_o = 724548 cells
Fig. 3c	n_p = 32 iterations, n_c = 384 cells	n_o = 159478 cells

Table 4 presents a comparison between the proposed heuristics and the method used in MDA to verify collisions in an arc, analyzing the time complexity in the worst case. It was considered r_c = 300 m, $\Delta p = 0.99 r_s$ = 29.7 m and ψ = 179.99°.

Analyzing the obtained results, the proposed heuristics is more efficient than the method used in MDA, when the navigation environment is represented by a regular grid and $n_o > 12n_p$ cells, in the worst case.



(a) (b)

Fig. 6. 1-trajectories planned by EDA, considering: (**a**) r_c = 300 m, w_s = (latitude, longitude) = (-23.075, -46.072), and w_d = (-22.98, -45.87); and (**b**) r_c = 2000 m, w_s = (-22.91, -45.95), and w_d = (-23.0, -45.735). The planned 1-trajectories are represented by the *black straight line segments*, *black circles* and *black stippled arcs*. The arcs of the 1-trajectories are sections of the *black stippled circles* defined by r_c. The *black arrow* indicates the waypoint w_d. The *gray circles* and the *gray straight line segments* represent, respectively, the nodes and edges of the sets of visibility graphs.

Examples of 1-trajectories planned by EDA are presented in Fig. 6, considering d_s = 100 m. As demonstrated in the previous section, these shortest 1-trajectories are collision-free and dynamically feasible for fixed-wing UAVs with the following constraints: $\xi < r_s$ = 30 m; ζ < 30 m; and $r_{cmin} \leq r_c \leq r_{cmax}$, where r_{cmin} and r_{cmax} are the minimum and maximum turning radius of the vehicle, respectively. The edges that connect the waypoint w_d to the set of visibility graphs can be visualized in Fig. 6b.

In all the tests, the time to plan was inferior to 2.95 seconds (s). The tests were run in a computer with one processor of 1.7 GHz and with 1 GB of RAM.

4 Conclusions

The obtained results prove the efficiency of the Elevation-based Dijkstra algorithm (EDA) proposed in this work to plan trajectories for fixed-wing UAVs. These trajectories are dynamically feasible and collision-free. The algorithm presents a local search heuristics to verify collision situation that is based on the elevation grid of the navigation environment. This proposed heuristics allows the EDA an improvement in comparison with the MDA, decreasing, significantly, the computational time.

As future work, the proposed heuristics will be used in Rapidly–exploring Random Trees (RRTs) aiming for the fixed-wing UAV trajectory planning. The results will be compared with the results obtained by the EDA applied to visibility graphs.

References

1. Anderson, E.P., Beard, R.W., McLain, T.W.: Real-time dynamic trajectory smoothing for unmanned air vehicles. IEEE Transactions on Control Systems Technology 13(3), 471–477 (2005)
2. Goerzen, C., Kong, Z., Mettler, B.: A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. Journal of Intelligent and Robotic Systems, 65–100 (2010)
3. Kuwata, Y., How, J.P.: Stable trajectory design for highly constrained environments using receding horizon control. In: Proceedings of the 2004 American Control Conference, Boston, Massachusetts, pp. 902–907 (June 2004)
4. Dijkstra, E.W.: A Note on Two Problems in Connection with Graphs. Numerische Mathematik 1, 269–271 (1959)
5. Medeiros, F.L.L., Silva, J.D.S.: Grafos de Visibilidade Aplicados à Representação Computacional de Ambientes de Navegação Aérea. In: X– Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE), São José dos Campos – SP (2008)
6. Nilsson, N.J.: A Mobile Automaton: An Application of Artificial Intelligence Techniques. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 509–520. ACM, New York (1969)
7. Sanchez, G., Latombe, J.C.: A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. In: Jarvis, R.A., Zelinsky, A. (eds.) Published in Robotics Research: The Tenth Int. Symp. Springer Tracts in Advanced Robotics, pp. 403–417. Springer, Heidelberg (2003)

Feasible UAV Path Planning Using Genetic Algorithms and Bézier Curves

Douglas Guimarães Macharet, Armando Alves Neto,
and Mario Fernando Montenegro Campos

Computer Vision and Robotic Laboratory (VeRLab)
Department of Computer Science, Universidade Federal de Minas Gerais
Belo Horizonte - MG - Brazil
{doug,aaneto,mario}@dcc.ufmg.br

Abstract. With the growing in the use of UAVs (Unmanned Aerial Vehicles), it is necessary to develop techniques that allow the generation of feasible paths for these vehicles. These paths take into account the nonholonomic constraints intrinsic to UAVs, such as minimum curvature, minimum torsion and maximum climb (or dive) angle. Thus, this paper proposes the use of genetic algorithms to generate paths for these vehicles in the three-dimensional space, using Bézier curves with several advantages. We consider all these three constraints in order to generate a feasible path for a small fixed-wing aircraft with severe limitations. We show results for this vehicle.

Keywords: Path planning, UAVs, Bézier curves, Genetic algorithm.

1 Introduction

Nowadays, a great effort is being invested to increase the ability of mobile robots to make their own decisions. Especially on issues linked to its mobility, reducing or eliminating altogether the need for human intervention on robot's assigned tasks.

Among the various issues involved, one well known problem is posed by the mobile robotics basic questions related to navigation: "How does it get to a given goal?". The question is directly linked to the strategy being used by the robot to safely achieve a goal position.

Several strategies to obtain a path between two different positions can be found in the literature [1]. Among the simplest and most common are the visibility graph, cell decomposition and potential field path planners.

However, a fundamental premise when planning a path for a robot is that this path is executable, i.e., during navigation, we must ensure that the vehicle's movement restrictions will be considered (i.e. non-holonomic constraints).

Most planners basically consider ground vehicles moving only in the plane (two-dimensional space). However, now we have seen a increase in the diversity of the types of vehicles used for many different missions in the three-dimensional space.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 223–232, 2010.

The interest and research in Unmanned Aerial Vehicles (UAVs) has been increasing growing, specially due to the decrease in cost, weight, size and performance of sensors and processors. Clearly UAVs have their niche of applications, which cannot be occupied by other types of mobile robots, as such as they are capable of covering a broad set of relevant applications. They are able to navigate over large areas obviously faster than land vehicles, with a privileged view from above, which is one of their main uses in monitoring and surveillance.

UAVs can be divided into at least three classes: rotary-wing aircrafts (e.g. helicopters and quadrotors), aerostatic aircrafts (such as airships and hot air balloons) and fixed-wing aircrafts (airplanes). The technique described in this text will be instantiated for fixed-wing UAV, however, without loss of generality, it can be applied to other types of vehicles. Fixed-wing aircrafts present constraints on their mobility such as minimum curvature, maximum angle of climb or dive, and minimum speed.

Due to the issues raised, it is necessary to study techniques capable of generate trajectories achievable by these types of vehicles. Therefore, this paper proposes the use of genetic algorithms to the generation of trajectories for unmanned aerial vehicles.

2 Related Work

Path planning problem for autonomous vehicles is the subject of many investigations, and various works on this topic can be found in literature [1,2,3]. One possible taxonomy of the area can be based on the number of vehicles involved, and the presence or absence of obstacles in the environment. Even though the generation of feasible trajectories for nonholonomic vehicles is in itself a great challenge, ignoring the possible presence of obstacles restricts even further a broader use of such techniques.

Some approaches of single vehicle path planning in general environments can be found in literature [4,5]. Voronoi diagrams is a widely used technique to generate paths with such constraints [6,7].

Rapidly-exploring Random Trees (RRTs) can also be used in this case, especially for solving the case of nonholonomic vehicles. In [8] the authors present trajectory planning for both an automobiles and a spacecraft. In the later example, even though an obstacle free environment is considered, the focus remains on the motion constraints that need to be satisfied for a safe entry of the spacecraft in Earth's atmosphere.

The use of evolutionary algorithms has also been focus of study as can be seen in [9,10,11,12]. However, most of these works only consider the curvature constraint of the vehicle. There are several other restrictions that must be taken into account when dealing with aerial robots, mainly when using fixed-wing aerial vehicles as UAVs. Such constraints are the torsion and inclination of flight or climb angle.

In [9], for example, there are some problems with respect to the used fitness function. Some constants must be defined experimentally, which is not desirable.

Another problem is that this technique do not avoid collision with obstacles, nor it guarantees that the maximum curvature constraint is fulfilled. The same problems can be identified in [10].

3 Methodology

Our technique assumes an obstacle free environment, where the only restrictions for the navigation of the robot are imposed only by its own kinematic constraints. Two configurations, $\mathbf{P}_i$ and $\mathbf{P}_g$ mark the initial and final poses respectively, which define the position and (partially) the orientation of the robot at the extreme points of the path.

A path may be defined mathematically as a parametric curve $\boldsymbol{r}(t)$ in three-dimensional space, where t is a parameter that continuously varies in $\mathbb{R}$. In this manner, the path planning problem can formally be described as:

$$\begin{aligned} \mathbf{P}_i(x_i, y_i, z_i, \psi_i, \theta_i) &= \boldsymbol{r}(t_i), \\ \mathbf{P}_g(x_g, y_g, z_g, \psi_g, \theta_g) &= \boldsymbol{r}(t_g), \end{aligned} \tag{1}$$

where t_i and t_g are the initial and final values, respectively, for the curve parameter t.

Each waypoint is described by three position (x, y, z) and two orientation (ψ, θ) variables. The variable ψ is an angle that describes the way-point orientation parallel to the XY plane in relation to the $\mathbf{X}$ axis. We define θ as the way-point orientation measured in relation to the $\mathbf{X}$ axis and parallel to XZ the plane.

The poses $\mathbf{P}_i$ and $\mathbf{P}_g$ can represent any pair of way-points in a set, which in turn, is determined by the high-level mission planning modules.

3.1 Constraints

In order to be considered a feasible path for the robot, the curve $\boldsymbol{r}(t)$ must simultaneously fulfill kinematic and dynamic constraints and their maximum values. The three motion constraints mentioned before are the maximum curvature (κ_{max}), the maximum torsion (τ_{max}) and the maximum climb (or dive) angle (θ_{max}) realizable by the robot in 3D space. It is possible to completely define a curve in $\mathbb{R}^3$ as a function of curvature and torsion only [13].

As far as the underlying physics of the system is concerned, the curvature may be defined as a quantity that is directly proportional to the lateral acceleration of the robot in the space. The value of κ_{max} is inversely proportional to the minimum curvature radius (ρ_{min}) of the curve that the vehicle is capable executing, which is also proportional to the maximum lateral acceleration of the vehicle. The curvature function of a curve in the n-dimensional space is given by the following equation:

$$\kappa(t) = \frac{|\dot{\boldsymbol{r}}(t) \times \ddot{\boldsymbol{r}}(t)|}{|\dot{\boldsymbol{r}}(t)|^3}. \tag{2}$$

The torsion can be seen as being directly proportional to the angular moment (roll moment) of the robot, which is also physically limited. Thus, the value of τ_{max} is given as a function of the minimum torsion radius (σ_{min}) that the robot can describe in space. The torsion of a curve can be computed:

$$\tau(t) = \frac{[\dot{\boldsymbol{r}}(t) \times \ddot{\boldsymbol{r}}(t)] \cdot \dddot{\boldsymbol{r}}(t)}{|\dot{\boldsymbol{r}}(t) \times \ddot{\boldsymbol{r}}(t)|^2} \,. \tag{3}$$

Finally, the climb (or dive) angle is proportional to the ascent (or descent) rate of the robot in 3D space. In other words, it captures the variation of the altitude (z) throughout the path. For vehicles with limited values of inclination angle, such as fixed-wing aircrafts, this is a fundamental variable. The value of θ_{max} may depend on many factors, as the translation speed and the spatial orientation of the robot. The climb angle function of a parametric curve in three-dimensions is given by:

$$\theta(t) = \tan^{-1}\left(\frac{\dot{z}(t)}{\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}}\right) . \tag{4}$$

It is possible to show that this function is mathematically confined to the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ [14]. The same is true for the value of θ_{max}.

Finally, the path $\boldsymbol{r}(t)$ is valid for a vehicle if the modulus of curvature, torsion and climb angle functions, are smaller than the vehicles absolute maximum values, as described below:

$$\boldsymbol{r}(t) \rightarrow (|\kappa(t)| < \kappa_{max}) \text{ and } (|\tau(t)| < \tau_{max}) \text{ and } (|\theta(t)| < \theta_{max}) \,. \tag{5}$$

As far as dynamics is concerned, it is important to consider the form by which these constraints vary in time. Guaranteeing the continuity of the curvature, torsion, and climb angle functions is another fundamental characteristic in the path planning module of real vehicles. Discontinuities in the curvature function, for example, may induce infinite variations of lateral acceleration, which of course, are not realizable. The same is true for torsional values. In the case of the climb angle function, the lack of continuity implies in the tangential discontinuity of the curve itself.

Finally, the curve produced by the path planning algorithm should be continuously derivable, and it also should be third order differentiable, according to Equation 3.

3.2 Genetic Algorithm

This section presents the steps and decisions involved in the development of the genetic algorithm.

Cromossome

One of the main steps in genetic algorithms is the choice of a good representation of the individual. This must represent a candidate solution to the problem, and

in our specific case the solution consists of a path that must be followed by the vehicle.

Analyzing different proposed solutions to the problem encountered in literature, it was decided to use a Bézier curve to represent the path. The Bézier curve has some interesting features as continuity (first and second derivative), i.e. it is smooth, and it is a parametric curve, which facilitates its manipulation.

A curve (or path) is defined based on the location of certain control points. As stated earlier, the path is composed of a initial position ($\mathbf{P}_i$) and a desired final position ($\mathbf{P}_g$), each one representing the points $\mathbf{p}_0$ and $\mathbf{p}_n$ respectivelly of the curve.

Hence, the individual used in the algorithm is a vector containing the positions in the space (x, y, z) of the n control points of the curve. However, because of the properties of the Bézier curves, that define the tangent orientation of its extreme points in function of the four extreme control points, these four of the n control points are fixed and calculated from the following equations, which depend solely on the initial and final configurations of the vehicle:

$$\begin{aligned} \mathbf{p}_0 &= [x_i,\, y_i,\, z_i], \\ \mathbf{p}_1 &= \mathbf{p}_0 + k_1[\cos(\psi_i)\cos(\theta_i),\, \sin(\psi_i)\sin(\theta_i),\, \sin(\theta_i)], \\ \mathbf{p}_{n-1} &= \mathbf{p}_n - k_2[\cos(\psi_g)\cos(\theta_g),\, \sin(\psi_g)\cos(\theta_g),\, \sin(\theta_g)], \\ \mathbf{p}_n &= [x_g,\, y_g,\, z_g], \end{aligned} \tag{6}$$

where k_1 e k_2 represent gain factors representing the modulus of the directive vectors.

Besides the most extreme points ($\mathbf{p}_0$ and $\mathbf{p}_n$) that represent the initial and final position, the second most extreme points ($\mathbf{p}_1$ and $\mathbf{p}_{n-1}$) must be fixed as well. By that, not only the localization will be respected, but also the initial and final orientation set for the vehicle.

Thus, the position of the remaining control points are free and will vary during the execution of the algorithm within a certain limit, thus seeking to generate a feasible path. When starting the execution of the algorithm ensures that the initial population generated be composed of individuals valid. We also choose to use elitism during the reproduction stage.

Fitness Function

In order to avoid a very complex fitness function $\mathcal{F}$ for the evolutionary algorithm, with several constants to be empirically defined, we propose to use a unique function use to minimize all the constraints we consider in this paper. This function consider not only the maximum curvature, torsion and climb angle values, but also its profiles throughout the curve and the path length. The solutions which we are looking for are those that best fitts to all these constraints.

The function we used is known as the Bending Elastic Energy function, proposed in [15]. It provides an approach to the energy spend on the path considering some aspects of a continuous curve. In this case, these aspects are represented by the total curvature function $\omega(t)$, as shown in Equation 7.

$$\mathcal{E} = \int_0^1 \omega(t)^2 \, |\dot{\boldsymbol{r}}(t)| \, dt \tag{7}$$

In [14], the author also propose an energy function that considers only the curvature and torsion constraints:

$$\omega(t) = \sqrt{\kappa(t)^2 + \tau(t)^2}. \tag{8}$$

This solution, however, is not satisfactory for the problem considered here, since it does not take into account the climb angle function in the energy computation. The curves that minimize the cost function present before may present climb (or dive) angles unattainable for a given robot.

In order to solve this problem we use the following elastic bending energy function, proposed in [16]:

$$\omega(t) = \sqrt{\left(\frac{\kappa(t)}{\kappa_{max}}\right)^2 + \left(\frac{\tau(t)}{\tau_{max}}\right)^2 + \left(\frac{\theta(t)}{\theta_{max}}\right)^2}. \tag{9}$$

Besides minimizing the increase in the rate of climb of the vehicle in three-dimensional space, this new function still takes into account the three aforementioned constraints and their maximum values in a normalized form. This is a very important consideration since ρ_{min}, σ_{min} and θ_{max} may differ from each other by orders of magnitude for the same robot.

Finally, we use the inverse of this equation as the fitness function, presented in the Equation 10. It is not necessary to use constants to adjust the different constraints of this evolutionary algorithm.

$$\mathcal{F} = \mathcal{E}^{-1} = \int_0^1 \omega(t)^{-2} \, |\dot{\boldsymbol{r}}(t)|^{-1} \, dt \ . \tag{10}$$

Crossover

The crossover step is responsible for implementing a change in the context of the existing genotypes, thus allowing for a somehow guided evolution. This should be held exchanging information between two selected individuals from the population (parents).

Therefore, it is selected at random a point where the change of information should occur. After that, the genotype is traversed from this point until the end, replacing each information of the control points of a certain position by the value of the control point at the same position of another individual.

However, it is remarkable that this operation can generate invalid individuals, i.e. not meeting the imposed restrictions. If this occurs, the individual is discarded and a new one is generated. Recalling that the generation of new individuals occurs while the number of individuals in the population is not achieved.

Mutation
After the crossover step is performed, it is verified if the chromosome should or should not suffer some kind of mutation, accordly to a designated probability. Unlike the previous step, the mutation does not occur among individuals, but but internally to the individual. This step is responsible for bringing innovation to the obtained results, postponing the convergence.

The mutation occurs as follows, given the individual, for each control point that can be manipulated (internal), it is checked from the mutation probability if that gene must be changed, in affirmative case, a new random value is assigned to the point $(x,\ y,\ z)$. Equally to the crossover step, new invalid individuals are discarded. Also, if any of the control points suffer a mutation, the parent is also discarded, by that reducing the number of repeated individuals between generations, and forcibly introducing novelty.

4 Experiments

Our technique was used to plan paths for a simulated small unmanned aerial vehicle. The vehicle were modeled as fixed-wings aircraft based on real UAV. This is a small hand launched hybrid electric motor sail plane, equipped with GPS receptor, barometric altimeter, infrared inclinometer, airspeed sensor and CCD camera, and controlled by a set of PID stabilizators running on a Palm® PDA for autonomous navigation.

The UAV presents the following characteristics: ρ_{min} about 50 meters, σ_{min} about 300 meters, θ_{max} about $\pi/30$ radians. These values were determined using data from actual flights, considering a speed of approximately 50 km/h. The simulated vehicle has θ_{max} equals $\pi/6$, a more characteristic value for fixed-wings aircrafts.

The algorithm was executed a total of 30 times, and the following parameters were used (parameters determined from initial experiments):

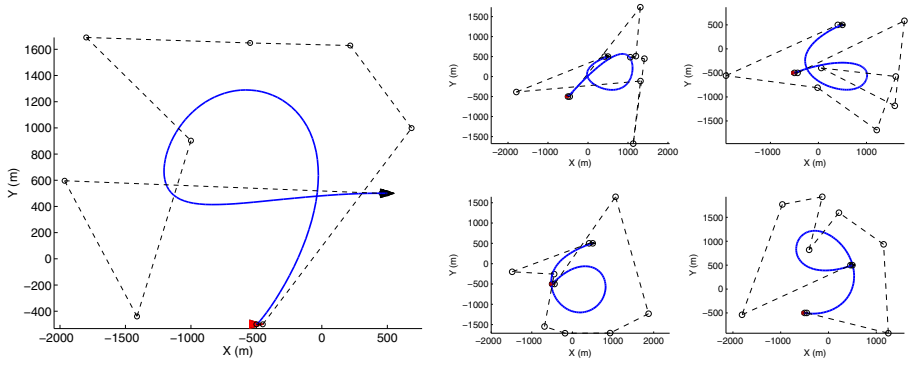
- Population size: 30
- Number of generations: 30
- Crossover probability: 0.95
- Mutation probability: 0.01
- Parents selection: Tournament of size 2
- Elitism: yes

To represent individuals was chosen a Bézier curve of tenth order, thus having seven variables control points.

4.1 Experiment 1

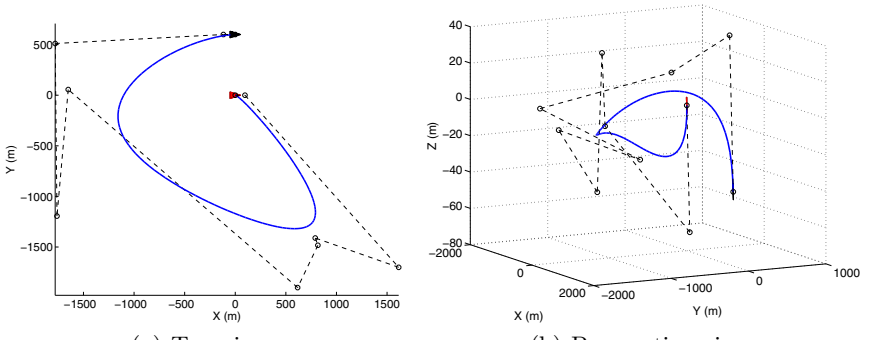
The initial and final configurations used in this experiment were:

- $\mathbf{P}_i\left(-500, -500, 0, -\frac{\pi}{5}, 0\right)$;
- $\mathbf{P}_g\left(500, 500, 50, \frac{\pi}{5}, 0\right)$,



(a) Best generated path. (b) Others generated paths.

Fig. 1. Some results obtained in experiment 1



(a) Top view. (b) Perspective view.

Fig. 2. Different views of the same result in experiment 2

representing a pose in the format (x, y, z, θ, ψ). These configurations were arbitrarily chosen.

Figure 1(a) presents the initial (red) and final (black) position and the generated path. The location of the control points are also showed.

Being a probabilistic method, it is interesting to observe that different executions will produce different results, and that is the main reason why the algorithm must be executed several times. Figure 1(b) exemplifies the variety of paths obtained.

It was observed that even with a number of 30 generations being used, the algorithm converged in half of this number for all executions of this experiment.

4.2 Experiment 2

In the next experiment, the initial and final configurations used were:

– $\mathbf{P}_i\left(0, 0, 0, -\frac{\pi}{5}, 0\right)$;
– $\mathbf{P}_g\left(0, 600, -50, -\frac{\pi}{5}, 0\right)$.

These configurations were chosen close to each other so as to present an extreme case the system. Figure 2 present the result path for the experiment. Table 1 presents, for both experiments, the maximum values obtained in the generated path for the curvature, torsion, climb angle and the maximum limit values, known as κ_{max}, τ_{max} and θ_{max}. Also the path length is presented. As can be seen, the curve fulfills all constraints, and by that being a feasible path for the vehicle.

Table 1. Path information of experiments 1 and 2

	$\kappa(m^{-1})$	$\tau(m^{-1})$	$\theta(rad)$	length(m)
Max values	$2.00\text{x}10^{-2}$	$3.30\text{x}10^{-3}$	$5.20\text{x}10^{-1}$	–
Experiment 1	$1.81\text{x}10^{-2}$	$3.11\text{x}10^{-4}$	$2.61\text{x}10^{-1}$	5035.77
Experiment 2	$1.43\text{x}10^{-2}$	$8.00\text{x}10^{-4}$	$4.13\text{x}10^{-1}$	5431.64

5 Conclusion and Future Work

This work presented the use of genetic algorithms for path planning of UAVs. The proposed methodology proved efficient, finding paths for all performed experiments. The paths fulfill all the constraints consider. The use of a energy function that incorporates all those constraints was a valuable characteristic of the genetic algorithm, mainly because it supress the problem of determine empirical constants. It is important to reforce that genetic algorithms are a stochastic technique, and thus the result may vary between executions, as was showed in Experiment 1.

Compared with other techniques in the literature, the method presented here incorpores several constraints that are very important when dealing with fixed-wing aerial vehicles (e.g. minimum curvature, minimum torsion and maximum climb angle). These are very limited and complex system, and need better techniques to allow autonomous navigation.

It was observed that the initial distribution of control points has direct influence on the final result. Thus a future work aims to define a better strategy for this distribution (a non-random strategy). The random strategy undermines the generation of the initial population, since in some cases the algorithm can not generate valid individuals (or takes a long period), and thus does not converge.

Related to the control points, the restriction of using a fixed number may also be removed, making it possible to remove existent points or insert new point during the mutation. By that, it would also be possible to optimize the number of necessary control points that describe a path.

Finally, a point to be further studied is wich is the best curve to be used to represent an individual. A widely used curve in other studies in the literature, and that should be better targeted in future study are the B-spline.

References

1. Siegwart, R., Nourbakhsh, I.R.: Introduction to Autonomous Mobile Robots. MIT Press, Cambridge (2004)
2. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
3. Dubins, L.E.: On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. American Journal of Mathematics 79, 497–516 (1957)
4. Kuwata, Y., Richards, A., Schouwenaars, T., How, J.P.: Robust Constrained Receding Horizon Control for Trajectory Planning. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2005)
5. Wzorek, M., Doherty, P.: Reconfigurable Path Planning for an Autonomous Unmanned Aerial Vehicle. In: International Conference on Hybrid Information Technology, ICHIT 2006, pp. 242–249 (November 2006)
6. Bortofi, S.A.: Path Planning for UAVs. In: Proceedings of the American Control Conference (2000)
7. Dogan, A.: Probabilistic Path Planning for UAVs. In: Proceedings of 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations (2003)
8. Cheng, P., Shen, Z., Lavalle, S.M.: RRT-Based Trajectory Design for Autonomous Automobiles and Spacecraft. Archives of Control Sciences 11(3-4), 167–194 (2001)
9. Nikolos, I., Valavanis, K., Tsourveloudis, N., Kostaras, A.: Evolutionary algorithm based offine/online path planner for UAV navigation. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 33(6), 898–912 (2003)
10. Hasircioglu, I., Topcuoglu, H.R., Ermis, M.: 3-d path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms. In: GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 1499–1506. ACM, New York (2008)
11. de la Cruz, J.M., Besada-Portas, E., Torre-Cubillo, L., Andres-Toro, B., Lopez-Orozco, J.A.: Evolutionary path planner for uavs in realistic environments. In: GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 14–77. ACM, New York (2008)
12. Pehlivanoglu, Y.V., Baysal, O., Hacioglu, A.: Path planning for autonomous UAV via vibrational genetic algorithm. Aircraft Engineering and Aerospace Technology: An International Journal 79(8), 352–359 (2007)
13. Kreyszig, E.: Differential Geometry, vol. 1. Dover Publications, New York (1991)
14. Farouki, R.T., Han, C.Y.: Algorithms for Spatial Pythagorean-Hodograph Curves. In: Geometric Properties for Incomplete Data, pp. 43–58 (2006)
15. Farouki, R.T.: The Elastic Bending Energy of Pythagorean Hodograph Curves. Comput. Aided Geom. Design 13, 227–241 (1996)
16. Alves Neto, A., Campos, M.F.M.: On the generation of feasible paths for aerial robots with limited climb angle. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2009), Kobe, Japan (2009)

High-Level Modeling of Component-Based CSPs

Raphaël Chenouard[1], Laurent Granvilliers[2], and Ricardo Soto[3]

[1] IRCCYN, Ecole Centrale de Nantes, France
[2] LINA, CNRS, Université de Nantes, France
[3] Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso, Chile
{ricardo.soto,laurent.granvilliers}@univ-nantes.fr,
{raphael.chenouard}@irccyn.ec-nantes.fr

Abstract. Most of modern constraint modeling languages combine rich constraint languages with mathematical notations to tackle combinatorial optimization problems. Our purpose is to introduce new component-oriented language constructs to manipulate hierarchical problems, for instance for modeling engineering system architectures with conditional sub-problems. To this end, an object-oriented modeling language is associated with a powerful constraint language. It offers the possibility of defining conditional components to be activated at solving time, declaring polymorphic components whose concrete types have to be determined, and overriding model elements. We illustrate the benefits of this new approach in the modeling process of a difficult embodiment design problem having several architectural alternatives.

1 Introduction

Constraint programming (CP) is a generic framework allowing users to state various kinds of constraint satisfaction problems (CSPs). Several specialized paradigms emerged to tackle specific concepts, for instance conditional CSPs [5] and composite CSPs [13].

A CSP is conditional when its sub-problems may not participate in the solutions. The activation of sub-problems is subjected to constraints to be verified during the solving process. A composite CSP can be seen as a hierarchy of sub-problems. It captures the inherent structure of a component-based system. Combining both features, composite and conditional CSPs make it possible to formulate models of complex systems and to study some variations in their architectures. This class of problems finds many applications in the configuration and design areas [12,5,8].

Modeling such problems is not trivial, as it is necessary to handle various concerns e.g., the component hierarchy, the activation of sub-problems, and even the definition of families of components sharing some properties. In this paper, we present a new approach to elegantly represent component-based CSPs. We extend the s-COMMA modeling language [15] by adding a new set of language constructs. We firstly introduce the concept of conditional (optional) component and we provide a simple construct to employ them in constraint models.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 233–242, 2010.

Depending on constraint-based conditions, it is possible to dynamically modify the problem architectures by activating new components. Second, components can be polymorphic. It is possible to define families of components, whose types can be refined at solving time. Third, as usual in object-oriented languages, model elements can be overridden in sub-types to facilitate the implementation of sub-classes. Finally, the language also makes it possible to handle components off the shelf or catalogs by means of compatibility constraints, which are sets of tuples to be assigned to a given set of variables.

Modeling complex real-world CSPs is a really hard task and s-COMMA is the result of several years of experience tackling that kind of problems. Now, let us illustrate such new modeling features by presenting a real-world CSP from Dassault Aviation about the design of an air conditioning system (ACS) for aircrafts.

1.1 Example

The ACS in an aircraft roughly mixes hot air from the turbojet (main air) and cold air from atmosphere (ram air) to inject air at right temperature and pressure in the cabin (see [3] for more details). Two topologies connecting several components are depicted in Fig. 1. On the left, only one heat exchanger is used to transfer the calorific energy between the main air and the ram air. On the right, a second heat exchanger is used to pre-cool the main air. Several other topological changes may also be considered in a more complete analysis of this complex system.

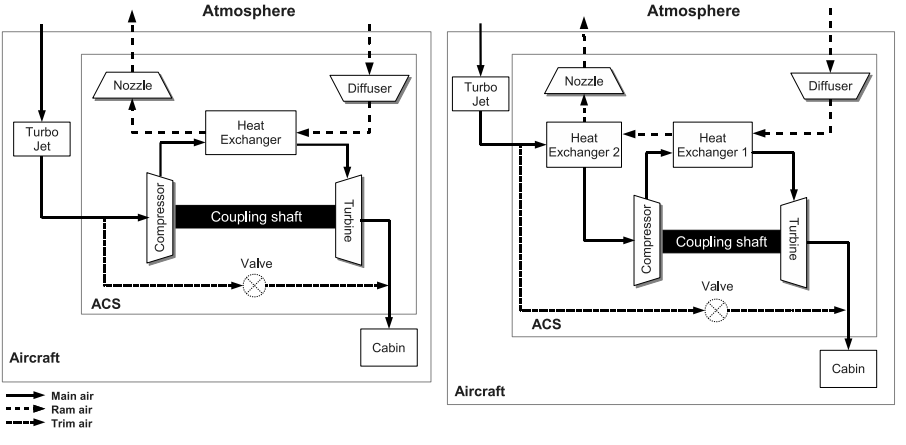


Fig. 1. Two possible topologies of an air conditioning system in an aircraft

These architectures are hierarchical, since every component can be refined as a complex system. Moreover, the choice of topology may depend on conditions. In this problem, the goals may be to simultaneously study:

- the embodiment design of the ACS, which in particular requires the solving of nonlinear constraints over continuous variables related to the geometry or the physical behavior of the system,
- the feasibility of each architectural design alternative.

Thus, we need a high-level modeling language being able to represent at least components subject to constraints, discrete and continuous variables, and conditional components.

1.2 Outline

The remaining of this paper is organized as follows. s-COMMA and the new modeling features are presented in Section 2. The architecture supporting the modeling language is shortly described in Sect. 3. The related work and conclusions follow.

2 Modeling

The core of the s-COMMA modeling language is briefly described now. A model is a collection of classes to be linked through usual composition and inheritance relations. The main class corresponds to the model entry, as follows.

```
1  // Model file
2  main class AirCraft {
3    ACS acs;                        // the ACS
4    TurboJet tj;                    // the turbojet
5
6    constraint Airflows {
7      tj.TIn = TRam;          // tj input = atmosphere
8      tj.TOut = acs.THotIn; // tj output = ACS input
9      acs.THotOut = TCab;    // ACS output = cabin
10     acs.TColdIn = TRam;    // ACS input = atmosphere
11     ...
12   }
13 }
14
15 class ACS {
16   HeatExchanger ex1; // the (first) heat exchanger
17   Turbine turbine;   // the turbine
18
19   real THotIn  in [200,1000]; // temperature in K of input hot air
20   real TColdIn in [200,1000]; //      "        in K of input cold air
21   real THotOut in [200,1000]; //      "        in K of output hot air
22   real TColdOut in [200,1000];//      "       in K of output cold air
23   real epsilon in [0,1];       // heat transfer efficiency
24   ...
25 }
26
27 class HeatExchanger {
28   real THotIn in [200,1000]; // temperature in K of input hot  air
29   real TColdIn in [200,1000]; //       "     in K of input cold air
30   real THotOut in [200,1000]; //       "     in K of output hot  air
31   real TColdOut in [200,1000];//       "     in K of output cold air
32   ...
```

In this model, the aircraft is composed of an air conditioning system `acs` and a turbojet `tj`, being instances of other classes. A constraint block `Airflows`

is defined to link the temperatures of air flows between the atmosphere, the turbojet, the ACS, and the cabin. A set of equality constraints is simply stated between air temperature attributes of `acs` and `tj`, and the flight conditions, namely the atmosphere air temperature `TRam` and the cabin air temperature `TCab`. These two parameters can be defined as constants in a data file for given flight conditions, as follows:

```
1  // Data file
2  real TRam := 220.055; // atmosphere air temperature in K at 10500m
3  real TCab := 280; // expected cabin air temperature in K
```

The `ex1` attribute corresponds to the heat exchanger in the first topology, and to the first one in the second topology. That means that every topology at least requires one exchanger. The presence of one or two exchangers will be more precisely discussed in Sect. 2.2.

For the sake of clarity, only a small extract of the full model is presented above. The other components are defined in the same way. In the rest of the section we focus on the new component-oriented language constructs.

2.1 Catalogs

In most configuration and design problems, it is necessary to reuse some components from the shelf, given as catalogs stating tuples of values for their main characteristics. For instance, a heat exchanger is made of exchange surfaces to be chosen from a set of known shapes. This can simply be modeled by means of a Boolean formula that restricts the possible values of several variables such as `rh`, `bh`, and `betah`.

```
1  class HeatExchanger {
2    ...
3    // Variables related to the exchange surfaces
4    real bh in [0,0.1];
5    real rh in [0,0.1];
6    real betah in [100,1.0e4];
7    ...
8  constraint SurfacesCatalog {
9    (rh=7.7089e-4 and bh=6.35e-3 and betah=1.204e+3) or
10   (rh=3.61315e-3 and bh=1.905e-3 and betah=2.496e+2) or
11   (rh=6.6167e-4 and bh=1.051e-2 and betah=1.368e+3) or
12   (rh=6.6992e-4 and bh=9.525e-3 and betah= 1.25e+3);
13 }
```

This is a natural representation of catalogs, but it is not suitable to deal with a higher number of constraints, as the model becomes confuse. To avoid this, it is possible to compact the model by using a compatibility constraint.

```
1    constraint SurfacesCatalog {
2      compatibility
3        (rh,          bh,         betah) {
4        (7.7089e-4,   6.35e-3,    1.204e+3);
5        (3.61315e-3,  1.905e-2,   2.496e+2);
6        (6.6167e-4,   1.051e-2,   1.368e+3);
7        (6.6992e-4,   9.525e-3,   1.25e+3);
8      }
9    }
```

2.2 Conditional Components

In various CP problems, it is necessary to model elements whose activation depends on the satisfaction of some conditions. For instance, consider the heat exchanger whose properties vary depending on the temperature of the input hot air (`THotIn`). This is commonly modeled by adding a set of implications that activate the involved constraints.

```
1  // Within the ACS class
2    constraint AirflowsOne {
3      (THotIn <= 600) -> ex1.TColdOut = nozzle.TIn;
4      (THotIn <= 600) -> THotIn = compressor.TIn;
5    }
6    constraint HeatEfficiencyOne {
7      (THotIn <= 600) -> epsilon = ex1.epsilon;
8    }
9  }
```

The model becomes more complicated if we need to model the activation of a new object depending on a condition. For instance, in the second ACS topology, an additional heat exchanger is required when the input hot temperature exceeds 600 K. One manner to state that is to include the second heat exchanger within the class and to activate the corresponding constraints.

```
1  // Within the ACS class
2    HeatExchanger ex2;  // the second exchanger
3
4    constraint AirflowsTwo {
5      (THotIn > 600) -> ex1.TColdOut = ex2.TColdIn;
6      (THotIn > 600) -> ex2.TColdOut = nozzle.TIn;
7      (THotIn > 600) -> THotIn = ex2.THotIn;
8      (THotIn > 600) -> ex2.THotOut = compressor.TIn;
9    }
10   constraint HeatEfficiencyTwo {
11     (THotIn > 600) -> epsilon = (ex1.epsilon + ex2.epsilon)/2;
12   }
13 }
```

Additionally, it is necessary to deactivate the object attributes (when the condition is not satisfied) in order to avoid useless splitting operations during the search. This can roughly be done by assigning to the variables its minimum domain value. Unfortunately, the resultant model is too verbose and hard to understand.

```
1    constraint deactivateAttributes {
2      not(THotIn > 600) -> (ex2.THotIn = minDom(ex2.THotIn));
3      not(THotIn > 600) -> (ex2.TColdIn = minDom(ex2.TColdIn));
4      not(THotIn > 600) -> (ex2.THotOut = minDom(ex2.THotOut));
5      not(THotIn > 600) ->  ...
6    }
```

To make the definition of such a conditional formulation more concise and understandable, we introduce a new conditional statement. This new construct considers a name, a condition, and a sequence of elements to activate if the condition is satisfied. For instance, the constraints defined within the `AirflowsOne` constraint block are only activated if the `OneExchanger` conditional block evaluates to true.

```
1  // Within the ACS class
2  cond OneExchanger (THotIn <= 600) {
```

```
3     constraint AirflowsOne {
4       ex1.TColdOut = nozzle.TIn;
5       THotIn = compressor.TIn;
6     }
7     constraint HeatEfficiencyOne {
8       epsilon = ex1.epsilon;
9     }
10  }
```

The integration of conditional objects is performed in the same way. This feature is more powerful since the set of variables and constraints embedded in the object can be activated at once, as for instance in the second ACS topology. The second heat exchanger is activated only if the `THotIn` attribute exceeds `600` K. It is also possible to include additional constraints acting over the conditional object, for example to establish the new links between air flows, and to compute the new heat efficiency.

```
1   // Within the ACS class
2   cond TwoExchangers (THotIn > 600) {
3     HeatExchanger ex2;  // the second exchanger
4
5     constraint AirflowsTwo {
6       ex1.TColdOut = ex2.TColdIn;
7       ex2.TColdOut = nozzle.TIn;
8       THotIn = ex2.THotIn;
9       ex2.THotOut = compressor.TIn;
10    }
11    constraint HeatEfficiencyTwo {
12      epsilon = (ex1.epsilon + ex2.epsilon)/2;
13    }
14  }
```

Note that the conditional block can also be located in a higher class (considering the composition's hierarchy), specifying that they apply on a given object. For instance, now the conditional block is stated within the `Aircraft` class, but it remains acting over the `acs` object.

```
1   class Aircraft {
2     ...
3     cond TwoExchangers (THotIn > 600) on acs {
4     ...
```

The main difference here is to modify a single object rather than the class itself. That allows one to manipulate instances of a same class with few modifications.

2.3 Polymorphic Components

Abstract or non final classes in object-oriented programming languages allow one to uniformly manipulate instances of sub-classes. From a modeling viewpoint, it is common to specify sub-types corresponding to families of systems differing in some aspects. Our goal is to give users a mean for declaring some objects of a given type and to let the solving engine dynamically refine this type, i.e., to allow non deterministic choices of component types.

We just define the new `polymorphic` keyword to be used in the declaration of a variable. As a consequence, every polymorphic object of type `T` occurring in a solution may have type `T` or a sub-type of `T`. In this case, we impose that the full type list from the inheritance hierarchy has to be explored.

This feature is illustrated by considering three main families of heat exchangers depending on directions of the cold and hot air flows: co-current flows, counter-current flows, and cross-current flows as depicted in Fig. 2. Each type from this family has its own way to compute the exchange of heat energy between the two air flows.

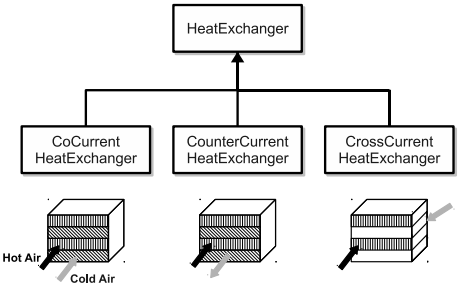


Fig. 2. A simple inheritance hierarchy for heat exchangers

The family of exchangers can be naturally represented by an inheritance graph as shown in Fig. 2. It directly follows the following classes.

```
1  class HeatExchanger { ... }
2  class COCHExchanger extends HeatExchanger { ... }
3  class CTCHExchanger extends HeatExchanger { ... }
4  class CRCHExchanger extends HeatExchanger { ... }
```

Now let the (first) heat exchanger from the ACS to be polymorphic. Furthermore, suppose that the second heat exchanger must be of type `CRCHExchanger` and that, in this case, the first exchanger must be of the same type. The following piece of model results.

```
1  // Within the ACS class
2  polymorphic HeatExchanger ex1; // polymorphic instance
3
4  constraint Airflows { // shared constraint between
5    ex1.THotOut = turbine.TIn; // the two topologies
6    ex1.TColdIn = diffuser.TOut;
7    ...
8  }
9
10 cond TwoExchangers (THotIn > 600) {
11   CRCHExchanger HeatExchanger ex2;  // the second exchanger
12
13   constraint TypeExchangerOne {// type restriction
14     typeOf(ex1) = typeOf(ex2); // for the first exchanger
15   }
16   ...
17 }
```

Introducing polymorphic instances leads to the need for manipulating instance types. In the example, the type of the first exchanger is restricted using the `typeOf` keyword (lines 13 to 15). That may be compared with runtime type checking mechanisms in computer programming languages, such as `instanceof` in `Java` or `type` in `Python`. The main difference is the declarative nature of the primitives in our language, which are just constraints on types.

2.4 Overriding Elements

The purpose of overriding mechanisms is to allow a sub-type from a hierarchy to provide a specific definition of an element — variable, constraint block, conditional block — defined in some ascending type. To this end, it suffices to declare an element using a naming correspondence.

In the ACS, the areas of the exchange surfaces are defined in the `HeatExchanger` base type. They are valid for all the given sub-types, except for the cold exchange surface of `CRCHExchanger` heat exchangers. In this case, it suffices to override the `ColdExchangeSurfaceArea` constraint, as follows.

```
1  class HeatExchanger {
2    // let Lx, Ly, and Lz be the dimensions
3    // let Afh and Afc be the related areas
4
5    constraint HotExchangeSurfaceArea {
6      Afh = Lx*Lz;
7    }
8    constraint ColdExchangeSurfaceArea {
9      Afc = Lx*Lz;
10   }
11 }
12
13 class CRCHExchanger extends HeatExchanger {
14   constraint ColdExchangeSurfaceArea {
15     Afc = Ly*Lz; // constraint overriding
16   }
17 }
```

3 Architecture

s-COMMA has been implemented on a model-driven solver-independent platform. Such an architecture allows the automatic reformulation of a s-COMMA model into different executable solver models. A model-driven engineering approach has been implemented to perform the necessary transformations among source, intermediate, and target models. Although, those transformations are not detailed here for space reasons, an extended presentation about them can be found in [15,1,2].

A general scheme of the reformulation process is depicted in Fig. 3. An s-COMMA model is the input of the system. This model is mapped to an intermediate model in order to reformulate the language constructs not supported at the solver level. Those transformations are standard and are based on the

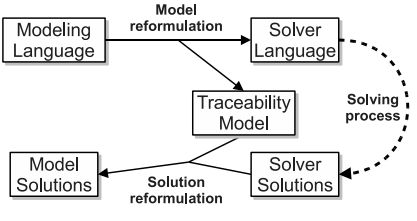


Fig. 3. Reformulation process

previous implementations of s-COMMA. For instance, objects are flattened i.e., every variable and constraint are just incorporated into the intermediate model. Compatibility constraints are mapped to the corresponding set of Boolean expressions. Conditional blocks are transformed into a set of implications, and polymorphic elements are stated as single conditional blocks. These conditional blocks are then treated as logical formulas. Once the reformulation is finished, a solver model is generated from the intermediate one, which is launched to obtain the results. Finally, the solution set is analyzed according to a traceability model to display the solutions in a correct object-oriented format.

4 Related Work

Zinc [9], MiniZinc [11] (successors of OPL [16]), and Essence [4] are the state-of-the-art constraint modeling languages. They provide a rich semantics for modeling different kinds of problems. In particular, Zinc offers the possibility of tackling specific applications domains by means of user-defined predicates. However, a main problem is that no object-oriented features are provided. Hence, it is difficult to naturally capture the structure of component-based problems.

Compatibility constraints can undoubtedly be represented by these languages as Boolean formulas, and conditional blocks can be simulated by using Boolean variables and/or implications as illustrated in Sect. 2. But, the result seems to be a hack of the model rather than a natural problem formulation. Finally, there is no mechanism to manage families of components through polymorphism.

COB [7] is an object-oriented constraint modeling language that can capture hierarchical problem structures. It provides a common language to post constraint models, which can be enriched with Prolog-like predicates. But, once again it lacks of compatibility constraints, conditional statements, and polymorphism. Additionally, less relevant to this paper, but not less important, the underlying architecture of COB is solver-dependent, being not possible to launch a model in different solving engines.

5 Conclusion

In this paper, we have presented new language constructs for modeling component-based CSPs. These constructs allow one to define more concise and understandable models, that naturally represents complex hierarchical problems. For instance, compatibility constraints avoids the definition of confuse Boolean expressions. Conditional blocks are a mean to express variables, constraints, and even objects whose relevance on the model depends on a given condition. Polymorphism is suitable for modeling families of components sharing some properties. Such constructs have been integrated on the s-COMMA language with the spirit of designing a full component-oriented language, an important CP challenge as stated in [8].

We believe that an important research direction is about solving strategies for component-based CSPs, interesting studies are pointed out in [6,5,14,10].

The main problem is that the set of required techniques may not be present in any existing solver, such as consistency techniques for mixed variables, interval methods for continuous problems, various search strategies, decomposition algorithms, and so on. Developing a new solver is a really hard task, and the solution may be to make several tools cooperate inside a CP platform.

References

1. Chenouard, R., Granvilliers, L., Soto, R.: Model-Driven Constraint Programming. In: Proceedings of ACM SIGPLAN PPDP, pp. 236–246. ACM Press, New York (2008)
2. Chenouard, R., Granvilliers, L., Soto, R.: Rewriting Constraint Models with Metamodels. In: Proceedings of SARA, pp. 42–49. AAAI Press, Menlo Park (2009)
3. Chenouard, R., Sébastian, P., Granvilliers, L.: Solving an Air Conditioning System Problem in an Embodiment Design Context Using Constraint Satisfaction Techniques. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 18–32. Springer, Heidelberg (2007)
4. Frisch, A.M., Harvey, W., Jefferson, C., Martínez Hernández, B., Miguel, I.: Essence: A constraint language for specifying combinatorial problems. Constraints 13(3), 268–306 (2008)
5. Gelle, E., Faltings, B.: Solving Mixed and Conditional Constraint Satisfaction Problems. Constraints 8, 107–141 (2003)
6. Geller, F., Veksler, M.: Assumption-Based Pruning in Conditional CSP. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 241–255. Springer, Heidelberg (2005)
7. Jayaraman, B., Tambay, P.: Modeling Engineering Structures with Constrained Objects. In: Krishnamurthi, S., Ramakrishnan, C.R. (eds.) PADL 2002. LNCS, vol. 2257, pp. 28–46. Springer, Heidelberg (2002)
8. Junker, U.: Configuration. In: Handbook of Constraint Programming, pp. 837–873. Elsevier, Amsterdam (2006)
9. Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P.J., Garcia de la Banda, M., Wallace, M.: The Design of the Zinc Modelling Language. Constraints 13(3), 229–267 (2008)
10. Mouhoub, M., Sukpan, A.: Managing Conditional and Composite CSPs. In: Kobti, Z., Wu, D. (eds.) Canadian AI 2007. LNCS (LNAI), vol. 4509, pp. 216–227. Springer, Heidelberg (2007)
11. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards A Standard CP Modelling Language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007)
12. O'Sullivan, B.A.: Constraint-Aided Conceptual Design. Professional Engineering Publishing (2001)
13. Sabin, D., Freuder, E.C.: Configuration as Composite Constraint Satisfaction. In: Proceedings of AI and MRP Workshop, pp. 153–161. AAAI Press, Menlo Park (1996)
14. Sabin, M., Freuder, E.C., Wallace, R.J.: Greater Efficiency for Conditional Constraint Satisfaction. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 649–663. Springer, Heidelberg (2003)
15. Soto, R., Granvilliers, L.: The Design of COMMA: An Extensible Framework for Mapping Constrained Objects to Native Solver Models. In: Proceedings of ICTAI, pp. 243–250. IEEE Computer Society, Los Alamitos (2007)
16. Van Hentenryck, P.: The OPL Optimization Programming Language. The MIT Press, Cambridge (1999)

Improving the Distributed Constraint Optimization Using Social Network Analysis

Allan Rodrigo Leite[1], André Pinz Borges[1],
Laércio Martins Carpes[1,2], and Fabrício Enembreck[1]

[1] Pontifical Catholic University of Paraná
{allan,andre.pinz,laercio,fabricio}@ppgia.pucpr.br
[2] Adventist University Center of São Paulo
laercio.carpes@unasp.edu.br

Abstract. Distributed Constraint Optimization Problem (DCOP) has emerged as one of most important formalisms for distributed reasoning in multiagent systems. Nevertheless, there are few real world applications based on methods for solving DCOP, due to their inefficiency in some scenarios. This paper introduces the use of Social Network Analysis (SNA) techniques to improve the performance in pseudo-tree-based DCOP algorithms. We investigate when the SNA is useful and which techniques can be applied in some DCOP instances. To evaluate our proposal, we use the two most popular complete and optimal DCOP algorithms, named ADOPT and DPOP, and compare the obtained results with others well-known pre-processing techniques. The experimental results show that SNA techniques can speed up ADOPT and DPOP algorithms.

Keywords: multiagent systems, distributed constraint optimization problem, pre-processing techniques.

1 Introduction

Multiagent systems represent a new paradigm of analyzing, projecting and developing complex systems. One of the main difficulties in modeling a multiagent system is to define the coordination model, because of the autonomous behavior of the agents [7]. Distributed Constraint Optimization Problem (DCOP) is a powerful formalism, capable to model a large class of real world problems naturally. This formalism provides a generic framework which allows agents to coordinate their actions under a certain domain, using only local communication and information. The coordination of these actions is performed by a global objective function modeled as a set of constraints, taking into account the interdependent partial solutions of each agent [12].

Although DCOP provides a generic approach able to model several multiagent problems naturally, there are few studies in the literature which introduce real world applications based on distributed constraint optimization, such as [5] [6] [8]. In most cases, it is connected with the inefficiency of current DCOP solvers. Since solving a DCOP optimally is known to be a NP-hard task, it is quite hard to find

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 243–252, 2010.

an optimal solution quickly or with a low computation cost. Although several researchers have developed algorithms to solve DCOPs optimally, unfortunately these algorithms still remain inefficient in some scenarios.

In this sense, pre-processing techniques have been investigated in recent decades to improving the efficiency of solving constraint satisfaction (CSP) and optimization (COP) problems. In CSP context, many techniques for pre-processing have been proposed, like arc, path, and k-consistency, consistency-enforcing, min-width, max-degree, and max-cardinality [3] [12]. In contrast, COP pre-processing techniques have focused in heuristics to guide the search quickly to the solution [1] [2].

In this paper, we propose a new pre-processing method to improve pseudo-tree-based DCOP algorithms. This particular kind of DCOP algorithms requires a pre-processing step to transform the constraints graph of the problem in a pseudo-tree structure. In other words, these algorithms need a pre-processing phase to order the agents into a hierarchical structure, before performing the search process. In contrast, it is well known that the ordering of agents may have a huge influence on the search space of the problem [4].

This work aims to demonstrate that Social Network Analysis (SNA) techniques can improve the performance in pseudo-tree-based DCOP algorithms. The key purpose of SNA is measuring the level of prominence of a node over the network. SNA techniques can be useful to detect and to privilege the most influential agents into the problem [14] reducing the search space by a strategic allocation of the prominent agents within the pseudo-tree.

To evaluate our proposal, we use the two most popular DCOP algorithms, namely ADOPT and DPOP. ADOPT performs a depth-first search method using distributed backtracking [9]. In contrast, DPOP implements an unusual strategy, based on dynamic programming [10]. Although these algorithms implement different search methods, both need a pseudo-tree priority order. We use two SNA measures in pre-processing phase to ordering the agents, named closeness and betweenness, and compare the obtained results with others well-known pre-processing techniques, called max-degree, min-width, and max-cardinality. We evaluate each measure in random DCOP instances with different densities and number of agents. Further, we discuss when SNA techniques are useful and which measure is the most suitable according to scenarios evaluated. The experimental results are encouraging. In most cases, we achieve a notable speed up in both algorithms.

2 Distributed Constraint Optimization Problems

Formally, a Distributed Constraint Optimization Problem (DCOP) is composed by n variables $V = \{v_1, v_2, \ldots, v_n\}$, and each variable is related to an agent x_i. Each variable has a domain both finite and discrete $D_1, D_2, \ldots, D_n$, respectively. Only agent x_i is able to set a value to v_i and also knows D_i.

In a DCOP, the constraints of a problem are called cost functions. By definition, a cost function for a pair of variables x_i and x_j is given by f_{ij}: D_i x $D_j \rightarrow N$. Two

agents x_i and x_j are neighbors if there is a constraint connecting them. Thus, a DCOP must find a set $A^* = \{d_1, \ldots, d_n \mid d_1 \in D_1, \ldots, d_n \in D_n\}$ of assignments, where the accumulated cost F is minimized, as showed in Equation 1 [13].

$$F(A) = \sum_{x_i, x_j \in V} f_{ij}(d_i, d_j), \text{where } x_i \leftarrow d_i, x_j \leftarrow d_j \in A \quad (1)$$

Several DCOP algorithms have been developed so far. ADOPT (Asynchronous Distributed Optimization) algorithm proposed by Modi et al. [9] was the first complete asynchronous method known to offer quality guarantee. In this algorithm, the agents must follow a tree-search priority ordering. By this ordering, the ADOPT tackles a depth-first search using distributed backtracking associated with an opportunistic strategy. Thus, each agent chooses a value to its variable based only on its local information. One of the disadvantages of this algorithm is the number of cycles and messages exchanged, which can be quite high in some situations.

On the other hand, the DPOP (Dynamic Programming Optimization Protocol) algorithm, proposed by Petcu and Faltings [10], implements an unusual strategy, based on dynamic programming. This strategy aims to explore the candidate states using cost matrixes, being able to offer better processing and communication costs most of the time. In this sense, DPOP is able to solve optimization problems with a linear number of cycles and messages. However, this feature is obtained with a high transfer rate due the messages size. Another disadvantage of DPOP algorithm is to be partially synchronous, forcing agents to wait for specific messages before continuing the search process, in some cases.

Both ADOPT and DPOP algorithms require a pseudo-tree priority order to perform their search process. These algorithms need a pre-processing phase to transform the constraints graph of the problem in a pseudo-tree, in order to obtain a ordering the agents into a hierarchical structure. A simple pre-processing method can be defined as depth-first search over the constraints graph, taking into account a priority order among agents by a lexicographical ordering.

3 Social Network Analysis

Social Networks and methods of Social Network Analysis (SNA) have attracted a considerable number of researchers in recent decades. SNA can be expressed as a set of techniques focused on the analysis of patterns or regularities in relationships among interacting units [14]. A Social Network is composed by a set of nodes and relationships, usually represented through a graph. A relationship is defined as a link which connects a pair of nodes.

The key purpose of SNA is measuring how prominent a node is. To achieve this purpose, SNA uses several graph theory concepts, such as path, degree of distribution, geodesic distance, and clusters. These concepts are usually useful to estimate the prominence of a node. There are many measures designed to highlight the most important node in a network [11]. In this paper, we focus in two SNA measures based on centrality concepts, namely closeness and betweenness.

Closeness estimates the prominence of a node by its distance of other nodes. In other words, closeness uses the geodesic distance $d(n_i, n_j)$ to measure the distance between two nodes n_i and n_j, where $n_j \neq n_i$. Geodesic distance represents the shortest path among a pair of nodes. In contrast, betweenness considers the interactions between two nonadjacent nodes might depend on the other intermediate nodes in a given network. Betweenness evaluates how between a given node n_i is according to the number of geodesic paths g_{jk} of all pair de nodes n_j and n_k, where $n_j \neq n_i$ and $n_k \neq n_i$.

4 Related Work

Pre-processing techniques have been investigated in recent decades to improving the efficiency of solving constraint satisfaction and optimization problems. These studies may be classified in two categories: consistency techniques (e.g. arc-consistency, path-consistency, k-consistency, or consistency-enforcing) and ordering variables methods (e.g. max-degree, minimum-width, or max-cardinality).

Consistency techniques are used to remove or reduce inconsistency among a pair or a set of connected nodes [4] [12]. In other words, consistency techniques transform a constraints graph into a more explicit representation before performing the search process. These techniques also involve estimating heuristics by solving a relaxed version of the problem [2]. In this case, the pre-processing technique is called consistency-enforcing and seeks to guide the search process quickly to the solution. In [1] was introduced an elegant pre-processing for DCOP using consistency-enforcing, composed by three heuristics based in dynamic programming. However, this pre-processing was designed to be used only on ADOPT algorithm, although the authors affirm that can be extended to other branch and bound DCOP algorithms.

In contrast, ordering variables methods aim to rearrange the variables in order to reduce the search space of the problem [4]. Since finding an optimal variable ordering is a NP-hard problem, many researchers have concentrated their studies in new heuristics to rearrange the variables. The three most popular ordering methods are max-degree, min-width, and max-cardinality.

procedure max-degree(a graph $G = (V,E)$,$V = \{v_1, \ldots, v_n\}$)
1: $d \leftarrow \{\}$
2: **for** $i = 1$ **to** number of nodes in G **do**
3: $d \leftarrow$ a variable with a maximum degree in V
4: $V \leftarrow V - r$
5: Put r at the end of d
6: **end for**
7: **return** d
end procedure

Fig. 1. The max-degree variable ordering procedure

procedure max-degree(a graph $G = (V,E)$,$V = \{v_1, \ldots, v_n\}$)
1: $d \leftarrow \{\}$
2: **for** $i =$ number of nodes in G **to** 1 **by** -1 **do**
3: $d \leftarrow$ a variable with a smallest degree in V
4: $V \leftarrow V - r$
5: Put r at the start of d
6: **end for**
7: **return** d
end procedure

Fig. 2. The min-width variable ordering procedure

procedure max-cardinality(a graph $G = (V,E)$,$V = \{v_1, \ldots, v_n\}$)
1: $d \leftarrow \{\}$
2: Put an arbitrary variable r at d
3: $V \leftarrow V - r$
4: **for** $i = 1$ **to** number of nodes in G **do**
5: $d \leftarrow$ a variable in V which is connected to a largest subset in d
6: $V \leftarrow V - r$
7: Put r at the end of d
8: **end for**
9: **return** d
end procedure

Fig. 3. The max-cardinality variable ordering procedure

The max-degree method orders the variables in a decreasing order by their degrees. In the graph theory, a degree $d(n_i)$ of a node represents the number of edges incident to n_i. The min-width method rearrange the variables from last to first by selecting of each node which has a minimal degree in the constraint graph. This heuristic is very similar to max-degree, but the min-width guarantees a variable ordering by a minimum node width. In this sense, a width of a node in an ordered graph represents its number of parents. Finally, the max-cardinality method attempts to arrange the graph by the most constrained variables. To reach this goal, the max-cardinality method selects the first variable randomly and chooses the next selected variables which are connected to the largest group among the variables already selected [3]. The max-degree, min-width, and max-cardinality variable ordering methods are presented in Figures 1, 2, and 3.

5 Improving DCOP Using SNA

As mentioned before, SNA techniques are able to detect who are the prominent agents in a given graph. Thus, we can take advantage of these techniques by applying them in the pre-processing phase on pseudo-tree-based DCOP algorithms. It is well known that the ordering of agents may have a huge influence on the search space of the problem [4]. In this way, it is feasible to reduce the search space by a

strategic allocation of the prominent agents within the pseudo-tree structure. In other words, we suggest using SNA measures to rearrange the agents ordering in pre-processing phase by their influence over the problem.

The key idea of our proposal is to make the most influential agents have higher priority in the pseudo-tree ordering. We use two SNA centrality measures, named closeness and betweenness, to detect which agents are prominent. Closeness focuses on how close an agent is of the other. In contrast, betweenness estimates how between an agent is according to the shortest paths to all pair of agents. Therefore, our strategy makes the prominent agents dominate the search process by assigning a high priority to them within the pseudo-tree. In addition, this proposal helps to balance the pseudo-tree, avoiding large branches.

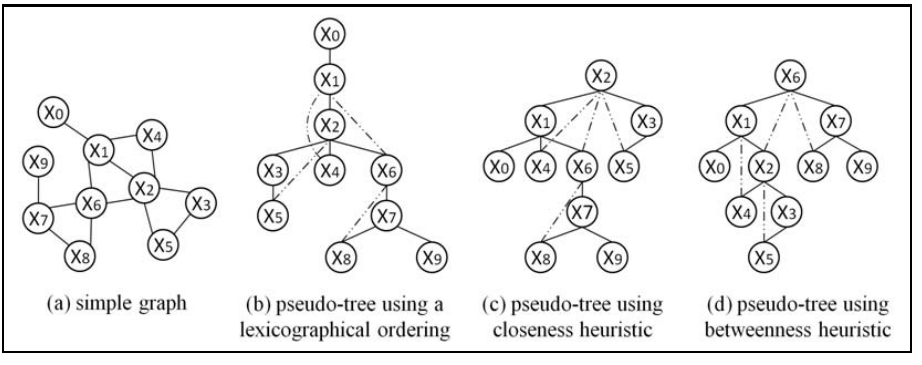


Fig. 4. Pseudo-tree arrangements produced with different heuristics

Figure 4 shows examples of three pseudo-tree arrangements from a simple graph (a) produced with different heuristics. The first pseudo-tree (b) was generated using a lexicographical ordering. The second pseudo-tree (c) was produced using closeness heuristic and we can observe that the x_2 is the most closeness agent, because it receives the higher priority in the pseudo-tree. Similarly, the third pseudo-tree (d) was produced using betweenness heuristic and we can observe that the x_{62} is the most intermediate agent, because it receives the higher priority in the pseudo-tree. Next sections give more details about closeness and betweenness heuristics.

5.1 Closeness Centrality

Closeness centrality measure uses geodesic distances to evaluate the proximity of an agent among all other agents [11]. Equation 2 shows how estimate the closeness $C_C(n_i)$ of an given agent n_i. For this measure, g represents the number of nodes in the graph and $d(n_i, n_j)$ denotes the geodesic distance between two nodes n_i and n_j

$$C_C(n_i) = \sum_{j=1}^{g} d(n_i, n_j), \text{where } n_j \neq n_i \tag{2}$$

5.2 Betweenness Centrality

Betweenness measure focuses in interactions between a pair of nonadjacent nodes. In fact, an intermediate agent potentially might have some control over the interactions between a pair of nonadjacent agents [14]. Equation 3 demonstrates a way to calculate the betweenness $C_B(n_i)$ of an agent n_i. For this measure, g_{jk} corresponds to the number of geodesic paths of all pair de nodes n_j and n_k, where $n_j \neq n_i$ and $n_k \neq n_i$, when n_i is in one or more paths between n_j and n_k.

$$C_B(n_i) = \sum_{j<k} [g_{jk}(n_i)/g_{jk}], \text{where } n_j \neq n_i \text{ and } n_k \neq n_i \tag{3}$$

6 Experiments and Results

To evaluate our proposal, we performed several experiments upon three distinct scenarios. These scenarios represent random constraints graph with different sizes (regarding the number of agents) and densities. For each experiment, we use five pre-processing based on variable ordering techniques, named max-degree, min-width, max-cardinality, closeness, and betweenness respectively. Each experiment was performed using different sizes and densities, involving 7, 10, and 15 agents varying link density in 3, 4, and 5.The experiments were made using two complete and optimal DCOP algorithms, named ADOPT and DPOP. Since the algorithms are asynchronous, we report averages over 25 problem instances for each experiment. However, we evaluate the performance in both algorithms using different metrics. Since the search strategies are distinct, we use the most suitable metric to assess the performance of each algorithm.

In this sense, for the ADOPT algorithm, we use the average of cycles count to evaluate the performance for each experiment. We do not use another metric, like message size for example, because for the ADOPT the message size is almost linear regarding the graph density [9]. In contrast, for the DPOP algorithm, we use message size to evaluate the performance for each experiment. Similarly, we do not use other metrics, like cycles or messages count, because DPOP can solve DCOP with linear number of cycles and messages regarding the agents [10].

Figure 5 shows the results obtained from ADOPT and DPOP algorithms to solve random problem instances with 7 agents, for each pre-processing method regarding the graph density. When analyzing the Figure 5, it is possible to notice that, when it is using pre-processing with SNA measures, less cycles and smaller messages are required in ADOPT and DPOP algorithm respectively to solve a problem. We also can notice that in higher density graphs we achieve better performance using betweenness heuristic.

Figure 6 shows the results obtained from ADOPT and DPOP algorithms to solve random problem instances involving 10 agents, for each pre-processing method regarding the graph density. When analyzing the Figure 6, it is possible to notice that, when it is using pre-processing with betweenness heuristic, less cycles and smaller messages are required in both algorithms to solve a problem, regarding other heuristics. Finally, Figure 7 shows the results obtained from ADOPT

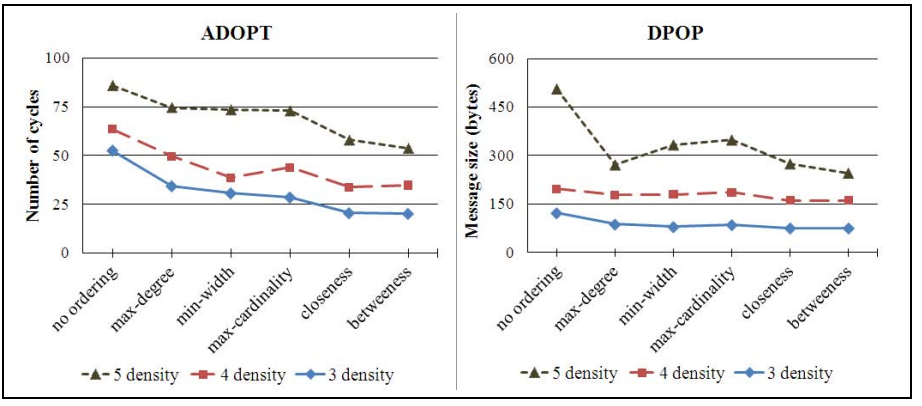


Fig. 5. Results from ADOPT and DPOP algorithms with 7 agents

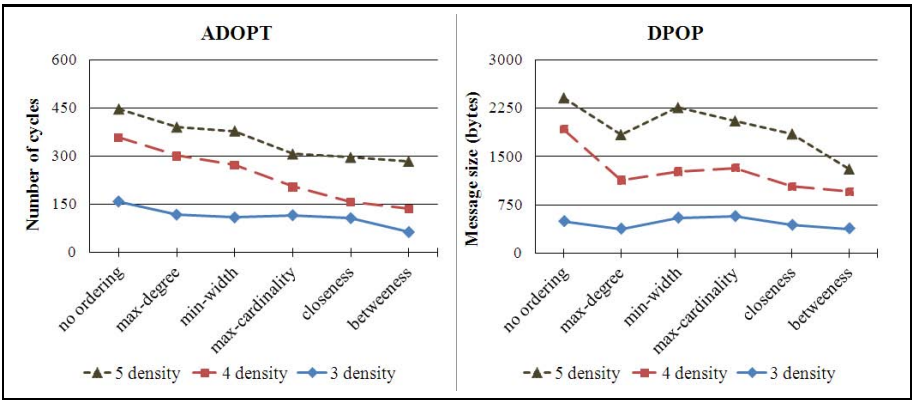


Fig. 6. Results from ADOPT and DPOP algorithms with 10 agents

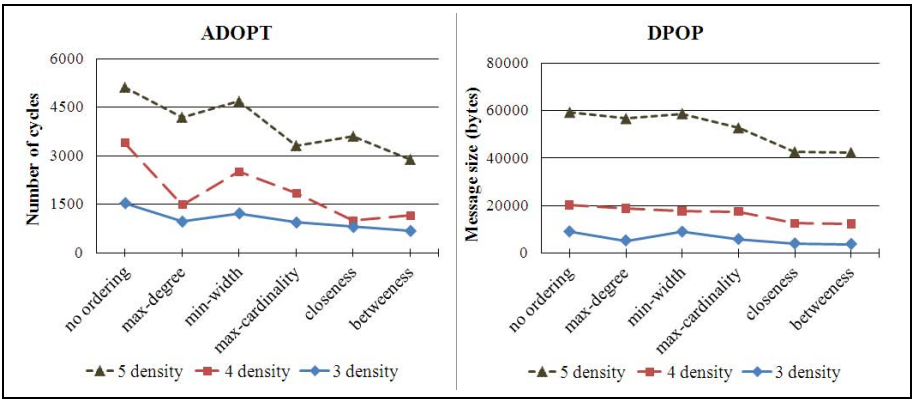


Fig. 7. Results from ADOPT and DPOP algorithms with 15 agents

and DPOP algorithms to solve random problem instances with 15 agents, for each pre-processing method regarding the graph density. Again, when it is using pre-processing with betweenness heuristic, less cycles and smaller messages are required to solve a problem. The main reason to speed up ADOPT algorithm when using SNA measures in pre-processing phase is due to the higher priority agent tends to change its value fewer times, regarding other agents. It happens because the higher priority agent only changes its value if some neighbor informs a cost greater than its local cost. Therefore, for the ADOPT algorithm, if the prominent agent does not have the higher priority, probably it will change its value many times. Consequently, these actions will affect the problem as a whole. Thus, the most influential agent causes a high impact (directly or not) over other agents involved in the problem. In contrast, we improve the DPOP performance when using SNA measures in pre-processing phase because it is propagate smaller cost matrixes to the higher priority agents. Thus, for the DPOP algorithm, we can obtain smaller messages through pseudo-trees with smaller branches. In general, we conclude that the pre-processing phase using betweenness heuristic is most appropriate on highly dense graphs for both algorithms and SNA theory provide us with concepts useful to improve DCOP algorithms.

7 Conclusions and Future Works

In this work we introduce the use of Social Network Analysis (SNA) techniques to improve the performance in Distributed Constraint Optimization Problem (DCOP) pseudo-tree-based algorithms. This kind of DCOP algorithms requires a pre-processing step to transform the constraints graph of the problem in a pseudo-tree structure. In this context, SNA techniques can be useful to detect and to privilege the most influential agents into the problem. Using this strategy, it is feasible reduce the search space by a strategic allocation of the prominent agents within the pseudo-tree.

To evaluate our proposal, we use the two most popular complete and optimal algorithm, namely ADOPT and DPOP. Although these algorithms implement different search methods, both need a pseudo-tree priority order. Our proposal uses two different SNA measures (closeness and betweenness) in pre-processing phase to ordering the variables of the problem. We evaluate each SNA measure in random DCOP instances and compare the results with three other well known ordering variable heuristics, called max-degree, min-width, and max-cardinality. We evaluate our proposal using different sizes for each problem, involving 7, 10, and 15 agents varying link density in 3, 4, and 5 respectively. From the obtained results, we demonstrate that it is possible to improve the ADOPT and DPOP performance using SNA techniques. We speed up both algorithms using SNA techniques in a pre-processing phase. By our strategy, we reach better performance than others well-known pre-processing methods. Finally, this work could be extended to other kind of DCOP algorithms. Currently, our strategy was designed to pseudo-tree-based DCOP algorithms. This strategy could be adapted to be used in dynamic

environments or in algorithms which not require a pseudo-tree priority ordering. We also suggest using other SNA measures in pre-processing, like clustering coefficient or information centrality. Such ideas should be studied in future researches.

References

1. Ali, S., Koenig, S., Tambe, M.: Preprocessing Techniques for Accelerating the DCOP Algorithm ADOPT. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1041–1048 (2005)
2. Bistarelli, S., Gennari, R., Rossi, F.: Constraint Propagation for Soft Constraints: Generalization and Termination Conditions. In: Dechter, R. (ed.) CP 2000. LNCS, vol. 1894, pp. 83–97. Springer, Heidelberg (2000)
3. Dechter, R.: Constraint Processing. Morgan Kaufmann, San Francisco (2003)
4. Dechter, R., Meiri, I.: Experimental Evaluation of Preprocessing Techniques in Constraint Satisfaction Problems. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 271–277 (1989)
5. Junges, R., Bazzan, A.L.: Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, Portugal Richland, pp. 599–606 (2008)
6. Leite, A.R., Giacomet, B., Enembreck, F.: Railroad Driving Model Based on Distributed Constraint Optimization. In: Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 474–482. IEEE Computer Society Press, Milan (2009)
7. Lesser, V., Ortiz, C.L., Tambe, M.: Distributed Sensor Networks: a Multiagent Perspective, vol. 9. Kluwer Academic Publishers, Massachusetts (2003)
8. Maheswaran, R.T., et al.: Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent, EUA, pp. 310–317. IEEE, Washington (2004)
9. Modi, P.J., et al.: ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, pp. 149–180. ACM, New York (2003)
10. Petcu, A., Faltings, B.: A Scalable Method for Multiagent Constraint Optimization. In: Proceedings of the International Joint Conference on Artificial Intelligence, Edinburgh, pp. 266–271. Morgan Kaufmann Publishers, San Francisco (2005)
11. Scoot, J.: Social network analysis: A handbook. Sage Publications, Thousand Oaks (2000)
12. Tsang, E.: Foundations of Constraint Satisfaction. Department of Computer Science, University of Essex, UK (1993)
13. Yokoo, M., et al.: The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. IEEE Transactions on Knowledge and Data Engineering 10, 673–685 (1998)
14. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press, Cambridge (1994)

A Survey and Classification of A* Based Best-First Heuristic Search Algorithms

Luis Henrique Oliveira Rios and Luiz Chaimowicz

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
{lhrios,chaimo}@dcc.ufmg.br

Abstract. A* (*a-star*) is a well known best-first search algorithm that has been applied to the solution of different problems. In recent years, several extensions have been proposed to adapt it and improve its performance in different application scenarios. In this paper, we present a survey and classification of the main extensions to the A* algorithm that have been proposed in the literature. We organize them into five classes according to their objectives and characteristics: *incremental*, *memory-concerned*, *parallel*, *anytime*, and *real-time*. For each class, we discuss its main characteristics and applications and present the most representative algorithms.

1 Introduction

Single-agent best-first search algorithms have been broadly applied in the resolution of several problems. One of the most important algorithms in this area is A* (*a-star*) [1]. A* has been used to solve problems of various areas such as the alignment of multiple DNA sequences in biology, path planning in robotics and digital games and classical artificial intelligence problems like the fifteen-puzzle. In spite of being extensively used, A* may present problems in some situations. Firstly, depending on the characteristics of the problem and heuristics used, its cost can be prohibitive. Also, some contexts such as dynamic environments or real-time searches may require adaptations in the original algorithm. Consequently, several extensions to the algorithm have been proposed in the last few years.

This paper presents a survey and classification of the main extensions to the A* algorithm that have been proposed in the literature. Since most of the extensions have specific objectives and try to improve similar points of the original algorithm, this type of classification is important to discuss and compare the new algorithms and to help users and researchers to keep track of the improvements. In this paper, we divide the algorithms into five classes: *incremental*, *memory-concerned*, *parallel*, *anytime*, and *real-time* according to their main characteristics. This paper is organized as follows: we start by presenting a quick review of the A* algorithm. Then, on Section 3, we describe each class discussing its main characteristics and presenting the most representative algorithms. Finally, Section 4 summarizes the classes and presents the conclusion and possibilities for future work.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 253–262, 2010.

2 Best-First Heuristic Search: A*

A* [1] is probably one of the most well known Artificial Intelligence algorithms. Its objective is to find the shortest path in a graph from a node x_{start} to a node x_{goal}. As a best-first heuristic search, it employs a function f that guides the selection of the next node that will be expanded. Using the notation from [2], function $f(x)$ is an estimate of $f^*(x)$ that is the cost of the shortest path that passes through a node x and achieves the goal. These two functions are computed as follows: $f(x) = g(x) + h(x)$ and $f^*(x) = g^*(x) + h^*(x)$. The term $g(x)$ is an estimate of $g^*(x)$, the cost of the shortest path from x_{start} to x, and $h(x)$ is an estimate of $h^*(x)$, the cost of the shortest path from x to x_{goal}. If $h(x) \leq h^*(x)$ for all x in the graph the heuristic is admissible and the algorithm can be proved optimal.

The implementation of the A* algorithm generally uses two lists named *open* and *closed*. The *open* list stores the nodes that are in the frontier of the search. The *closed* list stores the nodes that have already been expanded. In each iteration, the algorithm removes the node from the *open* list that has the smallest f-value, expands this node (inserts its successors that have not been expanded yet in the *open* list) and marks the node as expanded, inserting it in the *closed* list. It executes these steps until it removes x_{goal} from the *open* list or until there are no more nodes available in the *open* list. In the first case, A* has computed the shortest path between x_{start} and x_{goal}. If the second stop condition is true, there are not any solution available.

The main drawback of A* is that the number of nodes expanded can be exponential in the length of the optimal path. Therefore, the number of nodes stored in these two structures can be exponential in the length of the solution. The complexity analysis of A* and related algorithms depends primarily on the quality of the heuristic function and has been the subject of a large body of research, for example [3].

3 Classes

As mentioned, in recent years several extensions to the A* algorithm have been proposed. In this section, we organize them into five classes according to their objectives and characteristics: *incremental*, *memory-concerned*, *parallel*, *anytime*, and *real-time*. For each class, we discuss its main characteristics and applications and present the most representative algorithms.

3.1 The Incremental Class

The algorithms that belong to the incremental class assume that a series of similar search problems will be performed. The main idea is to reuse information from previous searches to solve new search problems faster than recomputing each problem from the beginning [4]. This is specially useful in dynamic

environments, in which the state space may change between searches. There are basically three different approaches to achieve this reuse of information [5][1]

- The first type restores the content of A* *open* and *closed* lists to the point in time when the current A* search can diverge from the previous one. That is, the state of an A* search, represented by the content of its lists, is recovered to allow the reuse of the beginning of the immediately preceding search tree. To be faster than a repetition of A* searches, this type of incremental search needs to efficiently reestablish the search state. Examples of algorithms that belong to this type of incremental search are: iA* [7] and Fringe-Saving A* (FSA*) [7]. The experiments executed in [7] showed that FSA* is faster than iA* and can be faster than LPA* (explained below) in some specific situations.
- The second type updates the heuristic (the nodes' h-values) to make it more accurate. Algorithms that belong to this class employ the last solution cost and the g-value of each node to change the h-values. They keep the heuristic consistent after the update phase. The fact that the heuristic is always consistent before starting another search guarantees that they will find an optimal solution if a solution exists. Generalized Adaptive A* (GAA*) [8] is an incremental heuristic search algorithm that adopts this technique to reuse information from previous searches.
- The last type of incremental heuristic search based on A* transforms the search tree of the previous search into the current search tree. Therefore, to be faster than an execution of A* from scratch, the overlap between the old and the new search trees needs to be large. Thus, these algorithms count on the fact that changes on the state space will affect only small branches of the search tree. If the changes occur near the search tree root, these algorithms will not perform well because there will be a small chance of large overlaps between the search trees. Lifelong Planning A* (LPA*) [9], D* [10] and D* Lite [2] are examples of this type of incremental heuristic search.

These algorithms, mainly the ones that belong to the last type, have been largely applied in robotics due to the dynamic fashion of most robotic applications. The D* Lite (a state-of-the-art incremental heuristic search algorithm [5]) is an example of algorithm that has been extensively applied in Robotics. In the same way as A*, D* Lite maintains, for each node x, an estimate of $g^*(x)$ denoted by $g(x)$. It also maintains a field named $rhs(x)$ that represents a one step lookahead of $g(x)$. Its value is computed based on x predecessors. According to the relation between these values, a node will be classified as local consistent ($g(x) = rhs(x)$) or local inconsistent (if $g(x) > rhs(x)$ it is overconsistent and if $g(x) < rhs(x)$ it is underconsistent).

D* Lite expands the nodes from x_{goal} to x_{start}. Its expansion loop expands the local inconsistent nodes following a precedence computed for each one using

[1] Although the authors have reported a problem with the experiments performed in [5] (see [6] for more information), the classification of incremental algorithms suggested on it still be valid.

its g-value, rhs-value and h-value. During this phase, its g-value and rhs-value are modified to eliminate the local inconsistency. D* Lite executes this loop until there are no more inconsistent nodes or until s_{start} becomes local consistent. The algorithm main loop calls this expansion loop to do the first search and starts executing the plan (moves the agent in direction to x_{goal}). If something in the graph changes, it updates the affected nodes and executes the expansion loop to recompute the shortest path.

Another incremental algorithm is called Field D* [11]. It is strongly based on D* Lite and has been used for path planning of real robotic systems in discretized (grid) environments. Traditional grid techniques may produce unnatural paths and require unnecessary robot movements because they restrict the robots' motion to a small set of directions (for example, 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, ...). However, this algorithm can generate smooth paths through non-uniform cost grids. That is, Field D* generates paths that can enter and exit cells at arbitrary positions. Basically, it changes the way the nodes are extracted from the grid. It also employs an interpolation to improve the cost estimation. The result are more natural paths in discretized environments.

3.2 The Memory-Concerned Class

The algorithms of this class tackle more specifically the problem related with the amount of memory necessary to store the *open* and *closed* lists. As mentioned, one of the main drawbacks of A* is the amount of space necessary to store the *open* and *closed* lists. This space restriction can be unacceptable for some applications. For example, the alignment of multiple DNA or protein sequences can be formalized as a shortest-path problem. A challenging feature of this particular search problem is its large branching factor, which is equal to $2^n - 1$ where n is the number of sequences to be aligned. The increased branching factor significantly worsens the algorithm memory requirements. Therefore, depending on the size of the problem, algorithms like A* can run out of memory before finding a solution.

There are three basic types of memory-concerned algorithms. The first two uses only the main (internal) memory [12], while the third one uses secondary memory as well.

The first type, named memory-efficient, does not keep the *open* and *closed* lists in memory. Typically, this kind of algorithm expands the same node multiple times and its space requirement is linear in the length of the solution. These algorithms clearly work on the traditional computer science trade-off: the compromise between memory usage and execution time. Two examples are Iterative-Deepening A* (IDA*) [13] and Recursive Best-First Search (RBFS) [14]. IDA* was derived from iterative deepening depth-first search. Differently from the original A*, it does not store the lists *open* and *closed*. It executes successive depth-first searches pruning paths that have a f-value greater than a threshold. The initial value of this boundary is the f-value of the start node. When it is not possible to proceed with the depth-first search, it updates the boundary to the smaller value that exceeded the previous threshold. The algorithm repeats these

steps until it finds the optimal solution. Its main weakness is the expansion of the same nodes multiple times.

One drawback of memory efficient algorithms is that they do not use all the internal memory available, so their running times tend to be longer. Named memory-bounded, the second type of memory concerned algorithms tries to overcome this problem retaining at least one of A* structures and limiting the maximum space occupied by them. For example, SMA* [15] and SMAG* [12] control the growth of the *open* list. When there is no more available space, they prune the least promising nodes to enable the insertion of more nodes. Another algorithm that belongs to the memory-bounded type is the Partial Expansion A* (PEA*) [16]. The idea behind this algorithm is to store only the necessary nodes for finding an optimal solution. PEA* has a predefined constant that helps in the decision of which nodes must be stored in the *open* list. To guarantee the optimal solution, the algorithm stores an additional value for each node that represents the lowest f-value among its unpromising child nodes. This algorithm was applied to the multiple sequence alignment problem and, on average, it could align seven sequences with only 4.7% of the amount of memory required by A*.

The third type of memory-concerned algorithms uses the secondary memory besides the main memory to store the algorithm structures. This kind of storage, disks for example, provides much more space with a small cost. However, to efficiently use this space, the algorithms must access it sequentially as the seek time is very high. These algorithms have mainly been applied to solve problems that demands a large amount of memory. The Frontier A*, for example, was implemented to store its *open* list using external memory [17]. Employing a technique called delayed duplicate detection (DDD), it does an external sort with multi-way merge and during this process it removes the duplications. One application of this algorithm, in the 4-peg Towers of Hanoi problem, searched a space with more than one trillion nodes. Frontier A* was also used to the optimal sequence alignment [18] outperforming the best existing competitors.

3.3 The Parallel Class

The algorithms of this class are suited for parallel execution, *i.e*, they explore the possibilities of concurrent execution to solve the search problem faster. Depending on the memory architecture, they are characterized as shared memory or distributed memory algorithms.

There are several reports on the adoption of parallel best-first heuristic search algorithms to solve search problems. Most of them were designed for distributed memory architectures. The Parallel Retracting A* (PRA*) [19], for example, has been used to solve the fifteen-puzzle with a significant speedup. PRA* can examine a large number of nodes simultaneously. It maintains an *open* list and a *closed* list for each processor in its local memory. To map the nodes to processors, a hash function is employed. As the first solution found is not necessarily the optimal one (because each processor expands the locally best node), PRA* maintains a global value with the best solution cost found so far. A similar algorithm, Parallel Local A* (PLA*) [20], has been successfully applied to solve the Traveling Salesman

Problem (TSP). Each processor has its local *open* and *closed* lists and the processors interact to inform the best solution found, to redistribute the work, to send or receive cost updates and to detect the algorithm termination.

On the other hand, the Parallel Best-NBlock-First (PBNF) algorithm [21] is an example of a shared memory parallel heuristic search algorithm. The PBNF implementation employs a technique called Parallel Structured Duplicate Detection (PSDD) to avoid the necessity of synchronization for each expanded node. The idea is to create an abstract function (specific for each kind of problem) that maps several nodes of the original graph to a unique node of the abstract graph (denoted as nblock). PBNF maintains an *open* list and a *closed* list for each nblock. When a node in a nblock b is expanded, its successors can only be in nblock b or in its successors. These set of nblocks are called the duplicate detection scope (DDS) of b. The abstract graph is used to help the selection of nblocks which DDS are disjoint. These nblocks can be explored in parallel without synchronization. As there are not any synchronization related with the f-values explored, the first solution found may not be the optimal. The algorithm will continue the search while there are nodes with f-values smaller than the cost of the current solution. The authors evaluated the algorithm in three different domains: STRIPS planning, grid path-finding and fifteen puzzle. They reported a significant speedup for all the evaluated domains.

3.4 The Anytime Class

The fourth class of single-agent best-first heuristic search considered in this work is known as anytime. An algorithm belongs to this class if it is able to give a feasible solution, not necessarily the best one, whenever the algorithm is stopped (except during the initial period when the first solution is being calculated) [22]. These algorithms are also expected to return better solutions when more time is provided for execution. After a sufficient execution time, the solution will converge and its quality will not increase anymore. That is, the main characteristic present in anytime algorithms is the trade-off between solution quality and computation time. This feature is very important for some applications as will be discussed next.

There are several reports of the adoption of anytime algorithms to solve search problems. A common feature of some anytime algorithms is the ability to prune the *open* list after the computation of the first solution, providing the necessary upper bound. The Anytime Weighted A* (AWA*) [23] is an example of such algorithm: it reduces the size of the *open* list when it prunes some paths. Similarly to A*, AWA* maintains two lists of nodes: *open* and *closed*. But it uses two evaluation functions, namely $f(x)$ (which is computed in the same way as A*) and $f'(x)$. The main difference between them is that the last is composed of the sum of $g(x)$ and $h'(x)$ (an inadmissible heuristic). This heuristic is computed by the following equation: $h'(x) = w \times h(x)$. That is, an inadmissible heuristic is acquired by multiplying the admissible heuristic by w, where $w \geq 1$ is a weight factor provided by the user. This parameter adjusts the trade off between time and solution quality.

AWA* employs the $f'(x)$ function to select the best node from the *open* list. Before expanding a node, the algorithm checks if it can prune it. To prune a node its f-value needs to be greater than the f-value of the best solution found so far. The algorithm can expand the same node more than once, like an implementation of A* that uses an admissible (but not consistent) heuristic. The algorithm termination condition is based on f-value and that is why it can find the optimal solution. It also provides an error upper bound that is the f-value of the best solution already found less the minimum f-value in the *open* list.

Another anytime algorithm is the Anytime Repairing A* (ARA*) [24]. It is very similar in operation to AWA*. The authors have successfully applied it for real-time robot path planning. Specifically, the algorithm performance has been evaluated on two applications: planning of robotic arm movements and path planning for mobile robots in outdoor environments. In the first application, they used other anytime algorithms (Anytime A* [25] and a succession of A* searches) to show that ARA* is more efficient. The robot arm application has a huge number of states and demands an anytime algorithm to prune some paths and reduce the computational resources necessary to compute the optimal solution.

Anytime Dynamic A* (AD*) [26] is an algorithm that combines the benefits of anytime and incremental searches. The authors have evaluated the algorithm using a robotic arm in a dynamic environment to show that it can generate better results than D* Lite [2] and ARA* (both algorithms have inspired the creation of AD*). They have also reported the adoption of AD* to plan paths in large partially known environments. In this context, the robot needs to be able to quickly generate solutions when information about the world changes and improve the solution quality without exceeding the time constraints. The problem was modeled as a search over a 4D state space involving position (x, y), orientation and speed. AD* was able to search this space and provide trajectory planning in a real-time fashion.

3.5 The Real-Time Class

The real-time class, also known as local search or agent-centered search [27], groups the algorithms that can search in the presence of time constraints. As they search with a limited lookahead, they can not guarantee the optimal solution. They can be considered a special case of anytime search but, as they have special characteristics that differ them from the approaches treated in subsection 3.4, they are presented separately. Typically, real-time search algorithms interleave planning and execution and are used in situations in which executing the plan within the time limits is more important than minimizing the path cost.

The Learning Real-Time A* (LRTA*) [28] is one of the first real-time heuristic search algorithms and inspired the creation of other approaches. It chooses the next state computing the f-value of all successors of the current state. In this case, $f(x)$ is the sum of the cost associated with the edge that connects the two nodes plus $h(x)$, where x is a successor of s (the current state). The next state will be the x with the smallest f-value. Before changing the current state,

LRTA* updates its heuristic, setting its value to the f-value of the next state. The name "learning" comes from the fact that after solving the problem multiple times, the h-values will converge to their exact values (that is, the h-values will converge to h^*-values). It is a real-time algorithm because one can configure the number of nodes considered (that is, the lookahead) in the step that selects the next state according to the time limits of the application.

Real-Time Adaptive A* (RTAA*) [29] is a real-time algorithm that has been applied to goal directed navigation of characters in real-time strategy (RTS) computer games. RTAA* employs the A* algorithm as a subroutine and limits its execution using a lookahead. After A* execution, the algorithm updates the h-values using the g-value of the best node in the *open* list. It is worth to mention that, due to its interactive fashion, computer games are very good candidates for the use of real-time planning algorithms.

Another suitable area for real-time search algorithms is Robotics. The Moving Target Search (MTS) [30] algorithm, which is a generalization of LRTA*, has been proposed to deal with robotic problems, for example, of a robot pursuing another one. When the target moves, the algorithm updates the h-values. The authors have also investigated bidirectional search and proposed an algorithm called Real-Time Bidirectional Search (RTBS). It can be viewed as a cooperative problem solving for uncertain and dynamic situations. Two kinds of RTBS have been proposed: the centralized RTBS (in which the best action is selected considering the two searches) and the decoupled RTBS (in which the two problem solvers make their decisions independently).

4 Summary and Conclusion

The A* algorithm is a powerful tool that can be used to solve a large number of different problems. However, its computational cost may be prohibitive for some applications. To alleviate this problem and to adapt it to different application contexts, various extensions of A* have been proposed. In this paper, we presented a survey and classification of the main extensions to the A* algorithm that have been proposed in the literature. We divided the algorithms in five classes: *incremental*, *memory-concerned*, *parallel*, *anytime*, and *real-time*.

The algorithms that belong to the incremental class reuse information from previous searches to solve similar search problems faster than recomputing each problem from the beginning. This type of algorithm has been largely used in Robotics since there is a lot of dynamism caused by sensor uncertainty and/or by the characteristics of the environment. These algorithms can also be useful in digital games where the environment is dynamic.

The memory-concerned class groups the algorithms that tackles the problems related with the amount of memory necessary to store the *open* and *closed* lists. As mentioned, these structures can grown exponentially depending on the application. So, memory-concerned algorithms try to limit the amount of memory used during the execution to be able to find a solution without exhausting the available memory.

An algorithm belongs to the parallel class if it tries to explore the advantages of parallel execution. These algorithms can be interesting for any application that has a hardware that provides the necessary conditions. It is currently being explored mainly in problems with large search spaces.

The anytime algorithms can generate a fast, non-optimal solution and then refine it to the optimal if more time is provided. Some applications have applied these algorithms to find a fast solution that will help the algorithm to prune some paths during the subsequent computation that will find the optimal solution. They can also be employed in situations that need quick answers such as digital games and robotics.

The Real-time class groups algorithms that can operate within time constrains. Typically, real-time search algorithms interleave planning and execution. Some digital games demand this kind of algorithm because they need to generate responses fast as they interact with the user in a real-time fashion.

This survey and classification is a first step in comparing these algorithms. We intend to extend the survey, including other algorithms and features that could not be described here due to space limitations. We also intend to experimentally evaluate these algorithms in order to compare the pros and cons of each class under several metrics in different applications. For this, we are planning to use toy problems as well as real applications in the areas of robotics and digital games.

References

1. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
2. Koenig, S., Likhachev, M.: Improved fast replanning for robot navigation in unknown terrain. In: Proc. of the Int. Conf. on Robotics and Automation, pp. 968–975 (2002)
3. Korf, R.E., Reid, M.: Complexity analysis of admissible heuristic search. In: Proceedings of the National Conference on Artificial Intelligence - AAAI (1998)
4. Koenig, S., Likhachev, M., Liu, Y., Furcy, D.: Incremental heuristic search in ai. AI Mag. 25(2), 99–112 (2004)
5. Sun, X., Yeoh, W., Koenig, S.: Dynamic fringe-saving a*. In: AAMAS 2009, pp. 891–898 (2009)
6. Koenig, S.: Dynamic fringe-saving a* (June 2010), http://idm-lab.org/bib/abstracts/Koen09e.html (Retrieved July 2010)
7. Sun, X., Koenig, S.: The fringe-saving a* search algorithm - a feasibility study. In: IJCAI, pp. 2391–2397 (2007)
8. Sun, X., Koenig, S., Yeoh, W.: Generalized adaptive a*. In: Int. Foundation for Autonomous Agents and Multiagent Systems AAMAS 2008, pp. 469–476 (2008)
9. Koenig, S., Likhachev, M., Furcy, D.: Lifelong planning a*. Artif. Intell. 155(1-2), 93–146 (2004)
10. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 3310–3317 (1994)

11. Ferguson, D., Stentz, A.: Field d*: An interpolation-based path planner and replanner. In: Proc. of the Int. Symposium on Robotics Research, pp. 1926–1931 (2005)
12. Zhou, R., Hansen, E.A.: Memory-bounded a* graph search. In: Proc. of the Fifteenth Int. Florida Artif. Intell. Research Society Conf., pp. 203–209. AAAI Press, Menlo Park (2002)
13. Korf, R.E.: Depth-first iterative-deepening: An optimal admissible tree search. Artif. Intell. 27, 97–109 (1985)
14. Korf, R.E.: Linear-space best-first search. Artif. Intell. 62(1), 41–78 (1993)
15. Russell, S.: Efficient memory-bounded search methods. In: ECAI 1992, pp. 1–5. Wiley, Chichester (1992)
16. Yoshizumi, T., Miura, T., Ishida, T.: A* with partial expansion for large branching factor problems. In: Proc. of the Seventeenth National Conf. on Artif. Intell. and Twelfth Conf. on Innovative Applications of Artif. Intell., pp. 923–929 (2000)
17. Korf, R.E.: Best-first frontier search with delayed duplicate detection. In: AAAI, pp. 650–657. AAAI Press, The MIT Press (2004)
18. Korf, R.E., Zhang, W., Thayer, I., Hohwald, H.: Frontier search. J. ACM 52(5), 715–748 (2005)
19. Evett, M., Hendler, J., Mahanti, A., Nau, D.: Pra*: massively parallel heuristic search. Technical report (1991)
20. Dutt, S., Mahapatra, N.R.: Parallel A* algorithms and their performance on hypercube multiprocessors. In: Proc. of the 7th Int. Parallel Processing Symposium, pp. 797–803. IEEE Computer Society Press, Los Alamitos (1993)
21. Burns, E., Lemons, S., Zhou, R., Ruml, W.: Best-first heuristic search for multi-core machines. In: IJCAI, pp. 449–455 (2009)
22. Teije, A., Harmelen, F.: Describing problem solving methods using anytime performance profiles. In: Proc. of the 14th European Conf. on Artif. Intell., pp. 181–185. IOS Press, Amsterdam (2000)
23. Hansen, E.A., Zhou, R.: Anytime heuristic search. J. Artif. Intell. Res (JAIR) 28, 267–297 (2007)
24. Likhachev, M., Gordon, G., Thrun, S.: Ara*: Anytime a* with provable bounds on sub-optimality. In: NIPS 2003. MIT Press, Cambridge (2004)
25. Zhou, R., Hansen, E.A.: Multiple sequence alignment using anytime a*. In: Eighteenth National Conf. on Artif. Intell., pp. 975–976. AAAI, Menlo Park (2002)
26. Likhachev, M., Ferguson, D.I., Gordon, G.J., Stentz, A., Thrun, S.: Anytime dynamic a*: An anytime, replanning algorithm. In: ICAPS 2005, pp. 262–271. AAAI, Menlo Park (2005)
27. Koenig, S.: Agent-centered search. AI Mag. 22(4), 109–131 (2001)
28. Korf, R.E.: Real-time heuristic search. Artif. Intell. 42(2-3), 189–211 (1990)
29. Koenig, S., Likhachev, M.: Real-time adaptive a*. In: AAMAS 2006, pp. 281–288. ACM Press, New York (2006)
30. Ishida, T.: Real-time search for autonomous agents and multiagent systems. Autonomous Agents and Multi-Agent Systems 1(2), 139–167 (1998)

A Sequent Calculus for 3-Dimensional Space

Norihiro Kamide

Waseda Institute for Advanced Study, Waseda University,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, Japan
logician-kamide@aoni.waseda.jp

Abstract. A central issue for spatial reasoning in practice is to formalize reasoning about 3-dimensional space. In this paper, a spatial logic called 3-dimensional spatial logic (3SL), which can appropriately represent the 3-Cartesian product ω^3 of the set ω of natural numbers, is introduced as a Gentzen-type sequent calculus. 3SL is an extension and generalization of the linear-time temporal logic LTL in which the time domain is ω. The completeness and cut-elimination theorems for 3SL are proved.

1 Introduction

A central issue for spatial reasoning in practice is to formalize reasoning about 3-dimensional space. It is known that time can be appropriately modeled by the set ω of natural numbers. Similarly, in a very simple case, space may be naturally modeled by the 3-Cartesian product ω^3 of ω. Although temporal logics based on ω have successfully been studied, spatial logics based on ω^3 have not yet been studied. An appropriate spatial logic with both semantically and proof-theoretically good properties has been required for specifying and verifying spatial properties in ω^3. The aim of this paper is thus to obtain a sound, complete and cut-free sequent calculus for reasoning about ω^3. For this aim, a new spatial logic called *3-dimensional spatial logic* (3SL) is introduced as a sequent calculus by extending and generalizing a sequent calculus for the well-known *linear-time temporal logic* LTL [1,4].

The proposed semantics for 3SL has 3-dimensional satisfaction relations $\models_{i_x;i_y;i_z}$ where i_x, i_y and i_z represent points on the x-axis, y-axis and z-axis, respectively. $\models_{i_x;i_y;i_z} \alpha$ intuitively means "α is true at the point $(i_x, i_y, i_z) \in \omega^3$." The spatial operators P_x (position in x-axis), P_y (position in y-axis), P_z (position in z-axis), A (anywhere) and A^- (converse anywhere) are then defined semantically by

1. $\models_{i_x;i_y;i_z} \mathrm{P}_x\alpha$ iff $\models_{i_x+1;i_y;i_z} \alpha$,
2. $\models_{i_x;i_y;i_z} \mathrm{P}_y\alpha$ iff $\models_{i_x;i_y+1;i_z} \alpha$,
3. $\models_{i_x;i_y;i_z} \mathrm{P}_z\alpha$ iff $\models_{i_x;i_y;i_z+1} \alpha$,
4. $\models_{i_x;i_y;i_z} \mathrm{A}\alpha$ iff $\models_{j_x;j_y;j_z} \alpha$ for any $j_x, j_y, j_z \in \omega$ with $j_x \geq i_x$, $j_y \geq i_y$ and $j_z \geq i_z$,
5. $\models_{i_x;i_y;i_z} \mathrm{A}^-\alpha$ iff $\models_{j_x;j_y;j_z} \alpha$ for any $j_x, j_y, j_z \in \omega$ with $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ and $0 \leq j_z \leq i_z$.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 263–272, 2010.

We illustrate some example formulas of 3SL. For the sake of simplicity, we restrict ourselves on the 2-dimensional space ω^2 in the following explanation. An expression $\mathrm{P}^i_m\alpha$ with $m \in \{x, y\}$ for any $i \in \omega$ is defined inductively by $\mathrm{P}^0_m\alpha \equiv \alpha$ and $\mathrm{P}^{i+1}_m\alpha \equiv \mathrm{P}_m\mathrm{P}^i_m\alpha$. We then illustrate the following formulas.

1. $\mathrm{P}^{i_x}_x\mathrm{P}^{i_y}_y\alpha$ means "α is true at the point $(i_x, i_y) \in \omega^2$."
2. $\mathrm{P}^2_x\mathrm{P}^2_y\mathrm{A}\alpha$ means "α is true at the area $\{(i_x, i_y) \in \omega^2 \mid i_x \geq 2, i_y \geq 2\}$."
3. $\mathrm{P}^3_x\mathrm{P}^3_y\mathrm{A}^-\alpha$ means "α is true at the area $\{(i_x, i_y) \in \omega^2 \mid 0 \leq i_x \leq 3, 0 \leq i_y \leq 3\}$."
4. $(\mathrm{P}^2_x\mathrm{P}^2_y\mathrm{A}\alpha) \wedge (\mathrm{P}^3_x\mathrm{P}^3_y\mathrm{A}^-\alpha)$ means "α is true at the area $\{(i_x, i_y) \in \omega^2 \mid i_x \geq 2, i_y \geq 2\} \cap \{(i_x, i_y) \in \omega^2 \mid 0 \leq i_x \leq 3, 0 \leq i_y \leq 3\}$, i.e., $\{(2, 2), (2, 3), (3, 2), (3.3)\}$."

Note that A^- is a useful new operator for representing areas in ω^3.

Remark that 3SL includes LTL as a sublogic. Assume the reduced language without $\{\mathrm{P}_y, \mathrm{P}_z, \mathrm{A}^-\}$. Suppose that P_x and A are read as X (next time) and G (anytime). Then, the corresponding one-dimensional satisfaction relation $\models_{i_x}$ is just the satisfaction relation of LTL, i.e., the one-dimensional case becomes the case of LTL.

Some theorems for embedding 3SL into infinitary logic are effectively used for proving the cut-elimination and completeness theorems for 3SL. These embedding theorems are inspired from the embedding theorem [2] of LTL into infinitary logic. In [2], the cut-elimination theorems for some extensions of Kawai's sequent calculus LT_ω [3] for LTL were proved using the corresponding embedding theorems. However, the completeness theorems for these extended logics have not yet been proved. The present paper establishes the way of proving the completeness theorem for 3SL.

The contents of this paper are then summarized as follows. In Section 2, 3SL is introduced as a Gentzen-type sequent calculus. The cut-elimination theorem for 3SL is proved using a theorem for "syntactically" embedding 3SL into a sequent calculus LK_ω for infinitary logic. In Section 3, a *space-indexed semantics* is introduced for 3SL, and a theorem for "semantically" embedding this semantics into a semantics of LK_ω. The completeness theorem with respect to the space-indexed semantics is proved combining the syntactical and semantical embedding theorems. In Section 4, this paper is concluded, and as an additional result, a fragment of 3SL is shown to be CoNP-complete.

2 Sequent Calculus and Cut-Elimination

The following list is adopted for the language of the underlying logic: (countable) propositional variables, $\rightarrow$ (implication), $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), P_x (position in x-axis), P_y (position in y-axis), P_z (position in z-axis), A (anywhere), S (somewhere), A^- (converse anywhere) and S^- (converse somewhere). Small letters $p, q, ...$ are used to denote propositional variables, Greek lower-case letters $\alpha, \beta, ...$ are used to denote formulas, and Greek capital letters $\Gamma, \Delta, ...$ are used to represent finite (possibly empty) sets of formulas.

An expression $\circ\Gamma$ where $\circ \in \{\mathrm{P}_x, \mathrm{P}_y, \mathrm{P}_z, \mathrm{A}, \mathrm{S}, \mathrm{A}^-, \mathrm{S}^-\}$ is used to denote the set $\{\circ\gamma \mid \gamma \in \Gamma\}$. An expression $A \equiv B$ denotes the syntactical identity between A and B. The symbol ω is used to represent the set of natural numbers. The symbol P is used to represent $\{\mathrm{P}_x, \mathrm{P}_y, \mathrm{P}_z\}$, and the symbol P^* is used to represent the set of all words of finite length of the alphabet P. Note that P^* includes $\emptyset$. Greek lower-case letter ι is used to denote any member of P^*. ι' $(\in P^*)$ is called a *permutation* of ι $(\in P^*)$ if ι' is obtained from ι by a permutation of the symbols in ι. For example, $\iota' \equiv \mathrm{P}_x\mathrm{P}_y\mathrm{P}_x$ is a permutation of $\iota \equiv \mathrm{P}_x\mathrm{P}_x\mathrm{P}_y$, but $\iota'' \equiv \mathrm{P}_x\mathrm{P}_y\mathrm{P}_y$ is not a permutation of ι. Remark that ι is itself a permutation of ι. Lower-case letters $i, j, k, i_x, i_y, i_z, ...$ are sometimes used to denote any natural numbers. An expression $\mathrm{P}^i_m\alpha$ with $m \in \{x, y, z\}$ for any $i \in \omega$ is defined inductively by $\mathrm{P}^0_m\alpha \equiv \alpha$ and $\mathrm{P}^{i+1}_m\alpha \equiv \mathrm{P}_m\mathrm{P}^i_m\alpha$. An expression of the form $\Gamma \Rightarrow \Delta$ is called a *sequent*. An expression $L \vdash S$ or $\vdash S$ is used to denote the fact that a sequent S is provable in a sequent calculus L. A rule R of inference is said to be *admissible* in a sequent calculus L if the following condition is satisfied: for any instance

$$\frac{S_1 \cdots S_n}{S}$$

of R, if $L \vdash S_i$ for all i, then $L \vdash S$.

The logic 3SL is then introduced below.

Definition 1 (3SL). *The initial sequents of* 3SL *are of the form: for any propositional variable p, $\iota p \Rightarrow \iota p$.*

The structural inference rules of 3SL *are of the form:*

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\mathrm{cut}) \qquad \frac{\Gamma \Rightarrow \Delta}{\Sigma, \Gamma \Rightarrow \Delta, \Pi}\ (\mathrm{we}).$$

The logical inference rules of 3SL *are of the form: for any $m, n \in \{x, y, z\}$,*

$$\frac{\Gamma \Rightarrow \Delta, \iota\alpha \quad \iota\beta, \Sigma \Rightarrow \Pi}{\iota(\alpha{\to}\beta), \Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\to\mathrm{left}) \qquad \frac{\iota\alpha, \Gamma \Rightarrow \Delta, \iota\beta}{\Gamma \Rightarrow \Delta, \iota(\alpha{\to}\beta)}\ (\to\mathrm{right})$$

$$\frac{\iota\alpha, \iota\beta, \Gamma \Rightarrow \Delta}{\iota(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\wedge\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \Delta, \iota\alpha \quad \Gamma \Rightarrow \Delta, \iota\beta}{\Gamma \Rightarrow \Delta, \iota(\alpha \wedge \beta)}\ (\wedge\mathrm{right})$$

$$\frac{\iota\alpha, \Gamma \Rightarrow \Delta \quad \iota\beta, \Gamma \Rightarrow \Delta}{\iota(\alpha \vee \beta), \Gamma \Rightarrow \Delta}\ (\vee\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \Delta, \iota\alpha, \iota\beta}{\Gamma \Rightarrow \Delta, \iota(\alpha \vee \beta)}\ (\vee\mathrm{right})$$

$$\frac{\Gamma \Rightarrow \Delta, \iota\alpha}{\iota\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\mathrm{left}) \qquad \frac{\iota\alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \iota\neg\alpha}\ (\neg\mathrm{right})$$

$$\frac{\iota\mathrm{P}_m\mathrm{P}_n\alpha, \Gamma \Rightarrow \Delta}{\iota\mathrm{P}_n\mathrm{P}_m\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Pleft}) \qquad \frac{\Gamma \Rightarrow \Delta, \iota\mathrm{P}_m\mathrm{P}_n\alpha}{\Gamma \Rightarrow \Delta, \iota\mathrm{P}_n\mathrm{P}_m\alpha}\ (\mathrm{Pright})$$

$$\frac{\iota\mathrm{P}^{k_x}_x\mathrm{P}^{k_y}_y\mathrm{P}^{k_z}_z\alpha, \Gamma \Rightarrow \Delta}{\iota\mathrm{A}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Aleft}) \qquad \frac{\{\ \Gamma \Rightarrow \Delta, \iota\mathrm{P}^{j_x}_x\mathrm{P}^{j_y}_y\mathrm{P}^{j_z}_z\alpha \mid j_x, j_y, j_z \in \omega\ \}}{\Gamma \Rightarrow \Delta, \iota\mathrm{A}\alpha}\ (\mathrm{Aright})$$

$$\frac{\{\ \iota\mathrm{P}^{j_x}_x\mathrm{P}^{j_y}_y\mathrm{P}^{j_z}_z\alpha, \Gamma \Rightarrow \Delta \mid j_x, j_y, j_z \in \omega\ \}}{\iota\mathrm{S}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Sleft})$$

$$\frac{\Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha}{\Gamma \Rightarrow \Delta, \iota \mathrm{S} \alpha} \ (\mathrm{Sright}) \quad \frac{\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha, \Gamma \Rightarrow \Delta}{\iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{A}^- \alpha, \Gamma \Rightarrow \Delta} \ (\mathrm{A}^-\mathrm{left})$$

with the conditions: $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ *and* $0 \leq j_z \leq i_z$,

$$\frac{\{\ \Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\ \}}{\Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{A}^- \alpha} \ (\mathrm{A}^-\mathrm{right})$$

$$\frac{\{\ \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha, \Gamma \Rightarrow \Delta \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\ \}}{\iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{S}^- \alpha, \Gamma \Rightarrow \Delta} \ (\mathrm{S}^-\mathrm{left})$$

$$\frac{\Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha}{\Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{S}^- \alpha} \ (\mathrm{S}^-\mathrm{right})$$

with the conditions: $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ *and* $0 \leq j_z \leq i_z$.

Note that (Aright) and (Sleft) have infinite premises. Note also that Gentzen's sequent calculus LK for classical logic is a subsystem of 3SL.

Proposition 2. *The rules of the form: for any* $m \in \{x, y, z\}$,

$$\frac{\Gamma \Rightarrow \Delta}{\mathrm{P}_m \Gamma \Rightarrow \mathrm{P}_m \Delta} \ (\mathrm{P}_m\mathrm{regu})$$

are admissible in cut-free 3SL.

Proof. By induction on the proofs P of $\Gamma \Rightarrow \Delta$ in cut-free 3SL. **Q.E.D.**

Proposition 3. *The sequents of the form:* $\iota\alpha \Rightarrow \iota\alpha$ *for any formula* α *are provable in cut-free* 3SL.

Proof. By induction on the complexity of α. We use Proposition 2. **Q.E.D.**

Next, we consider a sequent calculus LK_ω for infinitary logic in order to show the syntactical embedding theorem of 3SL into LK_ω. For more information on infinitary logic, see e.g. [2] and the references therein. A language of LK_ω is obtained from the language of 3SL by deleting $\{\mathrm{P}_x, \mathrm{P}_y, \mathrm{P}_z, \mathrm{A}, \mathrm{S}, \mathrm{A}^-, \mathrm{S}^-\}$ and adding $\bigwedge$ (infinitary conjunction) and $\bigvee$ (infinitary disjunction). For $\bigwedge$ and $\bigvee$, if Θ is a countable nonempty set of formulas, then $\bigwedge\Theta$ and $\bigvee\Theta$ are also formulas. Expressions $\bigwedge\{\alpha\}$ and $\bigvee\{\alpha\}$ are equivalent to α. The standard binary connectives $\wedge$ and $\vee$ are regarded as special cases of $\bigwedge$ and $\bigvee$, respectively.

A sequent calculus LK_ω for infinitary logic is then presented below.

Definition 4 ($\mathbf{LK}_\omega$). *The initial sequents of* LK_ω *are of the form: for any propositional variable* p, $p \Rightarrow p$.

The structural rules of LK_ω *are* (cut) *and* (we) *presented in Definition 1.*

The logical inference rules of LK_ω *are of the form:*

$$\frac{\Gamma \Rightarrow \Sigma, \alpha \quad \beta, \Delta \Rightarrow \Pi}{\alpha \to \beta, \Gamma, \Delta \Rightarrow \Sigma, \Pi} \ (\to\mathrm{left}^{\emptyset}) \quad \frac{\alpha, \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \to \beta} \ (\to\mathrm{right}^{\emptyset})$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\text{left}^{\emptyset}) \quad \frac{\alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\alpha}\ (\neg\text{right}^{\emptyset})$$

$$\frac{\alpha, \Gamma \Rightarrow \Delta \quad (\alpha \in \Theta)}{\bigwedge\Theta, \Gamma \Rightarrow \Delta}\ (\bigwedge\text{left}) \quad \frac{\{\ \Gamma \Rightarrow \Delta, \alpha \mid \alpha \in \Theta\ \}}{\Gamma \Rightarrow \Delta, \bigwedge\Theta}\ (\bigwedge\text{right})$$

$$\frac{\{\ \alpha, \Gamma \Rightarrow \Delta \mid \alpha \in \Theta\ \}}{\bigvee\Theta, \Gamma \Rightarrow \Delta}\ (\bigvee\text{left}) \quad \frac{\Gamma \Rightarrow \Delta, \alpha \quad (\alpha \in \Theta)}{\Gamma \Rightarrow \Delta, \bigvee\Theta}\ (\bigvee\text{right})$$

where Θ is a countable nonempty set of formulas.

The superscript "$\emptyset$" in the rule names in LK_ω means that these rules are the special cases of the corresponding rules of 3SL, i.e., the case that ι is $\emptyset$. The sequents of the form $\alpha \Rightarrow \alpha$ for any formula α are provable in cut-free LK_ω. As well-known, LK_ω enjoys cut-elimination.

Definition 5. *Fix a countable non-empty set Φ of propositional variables, and define the sets $\Phi_\iota := \{p_\iota \mid p \in \Phi\}$ $(\iota \in P^*)$ of propositional variables where $p_\emptyset = p$ (i.e., $\Phi_\emptyset := \Phi$). The language $\mathcal{L}^s$ (or the set of formulas) of* 3SL *is defined using Φ, $\to, \neg, \wedge, \vee$, P_x, P_y, P_z, A, S, A^- and S^-. The language $\mathcal{L}^i$ of LK_ω is defined using $\bigcup_{\iota \in P^*} \Phi_\iota$, $\to, \neg$, $\bigwedge$ and $\bigvee$. The binary versions of $\bigwedge$ and $\bigvee$ are also denoted as $\wedge$ and $\vee$, respectively, and these binary symbols are assumed to be included in $\mathcal{L}^i$. For any permutations ι_1 and ι_2 of ι $(\in P^*)$ and any $p \in \Phi$, we assume $p_{\iota_1} = p_{\iota_2}$, i.e., $\Phi_{\iota_1} = \Phi_{\iota_2}$.*

A mapping f from $\mathcal{L}^s$ to $\mathcal{L}^i$ is defined as follows.

1. *for any $p \in \Phi$, $f(\iota p) := p_\iota \in \Phi_\iota$ $(\iota \in P^*)$, esp., $f(p) := p \in \Phi_\emptyset$,*
2. *$f(\iota(\alpha \circ \beta)) := f(\iota\alpha) \circ f(\iota\beta)$ where $\circ \in \{\to, \wedge, \vee\}$,*
3. *$f(\iota\neg\alpha) := \neg f(\iota\alpha)$,*
4. *for any $l, m \in \{x, y, z\}$, $f(\iota \mathrm{P}_l \mathrm{P}_m \alpha) := f(\iota \mathrm{P}_m \mathrm{P}_l \alpha)$,*
5. *$f(\iota\mathrm{A}\alpha) := \bigwedge\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid j_x, j_y, j_z \in \omega\}$,*
6. *$f(\iota\mathrm{S}\alpha) := \bigvee\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid j_x, j_y, j_z \in \omega\}$,*
7. *for any $i_x, i_y, i_z \in \omega$, $f(\iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{A}^- \alpha) :=$*
$$\bigwedge\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid 0 \le j_x \le i_x, 0 \le j_y \le i_y, 0 \le j_z \le i_z\},$$
8. *for any $i_x, i_y, i_z \in \omega$, $f(\iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{S}^- \alpha) :=$*
$$\bigvee\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid 0 \le j_x \le i_x, 0 \le j_y \le i_y, 0 \le j_z \le i_z\}.$$

An expression $f(\Gamma)$ denotes the result of replacing every occurrence of a formula α in Γ by an occurrence of $f(\alpha)$.

Theorem 6 (Syntactical embedding). *Let Γ and Δ be sets of formulas in $\mathcal{L}^s$, and f be the mapping defined in Definition 5. Then:*

1. *if* 3SL $\vdash \Gamma \Rightarrow \Delta$, *then* $\mathrm{LK}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta)$.
2. *if* $\mathrm{LK}_\omega - (\mathrm{cut}) \vdash f(\Gamma) \Rightarrow f(\Delta)$, *then* 3SL $- (\mathrm{cut}) \vdash \Gamma \Rightarrow \Delta$.

Proof. (1) : By induction on the proofs P of $\Gamma \Rightarrow \Delta$ in 3SL. We distinguish the cases according to the last inference of P, and show some critical cases.

Case ($\iota p \Rightarrow \iota p$): The last inference of P is of the form: $\iota p \Rightarrow \iota p$. In this case, we obtain $\mathrm{LK}_\omega \vdash f(\iota p) \Rightarrow f(\iota p)$, i.e., $\mathrm{LK}_\omega \vdash p_\iota \Rightarrow p_\iota$ ($p_\iota \in \Phi_\iota$) by the definition of f.

Case (Aleft): The last inference of P is of the form:

$$\frac{\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha, \Gamma \Rightarrow \Delta}{\iota \mathrm{A} \alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Aleft}).$$

By induction hypothesis, we have $\mathrm{LK}_\omega \vdash f(\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha), f(\Gamma) \Rightarrow f(\Delta)$. Let Φ be $\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid j_x, j_y, j_z \in \omega\}$. We then obtain the required fact:

$$\frac{\begin{array}{c}\vdots \\ f(\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha), f(\Gamma) \Rightarrow f(\Delta) \quad (f(\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha) \in \Phi)\end{array}}{\bigwedge \Phi, f(\Gamma) \Rightarrow f(\Delta)}\ (\textstyle\bigwedge \mathrm{left})$$

where $\bigwedge \Phi$ coincides with $f(\iota \mathrm{A} \alpha)$ by the definition of f.

Case (Aright): The last inference of P is of the form:

$$\frac{\{\ \Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha \mid j_x, j_y, j_z \in \omega\ \}}{\Gamma \Rightarrow \Delta, \iota \mathrm{A} \alpha}\ (\mathrm{Aright}).$$

By induction hypothesis, we have $\mathrm{LK}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta), f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha)$ for all $j_x, j_y, j_z \in \omega$. Let Φ be $\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid j_x, j_y, j_z \in \omega\}$. We obtain the required fact:

$$\frac{\begin{array}{c}\vdots \\ \{\ f(\Gamma) \Rightarrow f(\Delta), f(\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha) \mid f(\iota \mathrm{P}_x^{k_x} \mathrm{P}_y^{k_y} \mathrm{P}_z^{k_z} \alpha) \in \Phi\ \}\end{array}}{f(\Gamma) \Rightarrow f(\Delta), \bigwedge \Phi}\ (\textstyle\bigwedge \mathrm{right})$$

where $\bigwedge \Phi$ coincides with $f(\iota \mathrm{A} \alpha)$ by the definition of f.

Case (A^-right): The last inference of P is of the form:

$$\frac{\{\ \Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\ \}}{\Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{A}^- \alpha}\ (\mathrm{A}^-\mathrm{right}).$$

By induction hypothesis, we have $\mathrm{LK}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta), f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha)$ for all $j_x, j_y, j_z \in \omega$ with the conditions: $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ and $0 \leq j_z \leq i_z$. Let Φ be $\{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\}$. We obtain the required fact:

$$\frac{\begin{array}{c}\vdots \\ \{\ f(\Gamma) \Rightarrow f(\Delta), f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \in \Phi\ \}\end{array}}{f(\Gamma) \Rightarrow f(\Delta), \bigwedge \Phi}\ (\textstyle\bigwedge \mathrm{right})$$

where $\bigwedge \Phi$ coincides with $f(\iota \mathrm{P}_x^{i_x} \mathrm{P}_y^{i_y} \mathrm{P}_z^{i_z} \mathrm{A}^- \alpha)$ by the definition of f.

(2) : By induction on the proofs Q of $f(\Gamma) \Rightarrow f(\Delta)$ in LK_ω. We distinguish the cases according to the last inference of Q. We show only the following case.

Case ($\bigwedge$right): The last inference of Q is of the form:

$$\frac{\{\ f(\Gamma) \Rightarrow f(\Delta), f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \in \Phi\ \}}{f(\Gamma) \Rightarrow f(\Delta), \bigwedge \Phi}\ (\bigwedge \text{right})$$

where $\Phi = \{f(\iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha) \mid j_x, j_y, j_z \in \omega\}$, and $\bigwedge \Phi$ coincides with $f(\iota \mathrm{A} \alpha)$ by the definition of f. By induction hypothesis, we have $\mathrm{3SL} \vdash \Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha$ for all $j_x, j_y, j_z \in \omega$. We thus obtain the required fact:

$$\frac{\begin{array}{c}\vdots \\ \{\ \Gamma \Rightarrow \Delta, \iota \mathrm{P}_x^{j_x} \mathrm{P}_y^{j_y} \mathrm{P}_z^{j_z} \alpha \mid j_x, j_y, j_z \in \omega\ \}\end{array}}{\Gamma \Rightarrow \Delta, \iota \mathrm{A} \alpha}\ (\text{Aright}).$$

Q.E.D.

Theorem 7 (Cut-elimination). *The rule* (cut) *is admissible in cut-free* 3SL.

Proof. Suppose $\mathrm{3SL} \vdash \Gamma \Rightarrow \Delta$. Then, we have $\mathrm{LK}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta)$ by Theorem 6 (1), and hence $\mathrm{LK}_\omega - (\mathrm{cut}) \vdash f(\Gamma) \Rightarrow f(\Delta)$ by the cut-elimination theorem for LK_ω. By Theorem 6 (2), we obtain $\mathrm{3SL} - (\mathrm{cut}) \vdash \Gamma \Rightarrow \Delta$. **Q.E.D.**

Remark that by Theorem 7, we can strengthen the statements of Theorem 6 by replacing "if then" with "iff". This fact will be used to prove the completeness theorem for 3SL. Remark also that the "iff" condition cannot directly be proved before showing the cut-elimination theorem, since the case of (cut) cannot be shown.[1]

3 Semantics and Completeness

Let Γ be a set $\{\alpha_1, ..., \alpha_m\}$ $(m \geq 0)$ of formulas. Then, Γ^* represents $\alpha_1 \vee \cdots \vee \alpha_m$ if $m \geq 1$, and otherwise $\neg(p \rightarrow p)$ where p is a fixed propositional variable. Also Γ_* represents $\alpha_1 \wedge \cdots \wedge \alpha_m$ if $m \geq 1$, and otherwise $p \rightarrow p$ where p is a fixed propositional variable. The symbol $\geq$ or $\leq$ is used to represent a linear order on ω.

A semantics for 3SL is defined below.

Definition 8. *Space-indexed valuations* $I^{i_x;i_y;i_z}$ $(i_x, i_y, i_z \in \omega)$ *are mappings from the set of all propositional variables to the set* $\{t, f\}$ *of truth values. Then, space-indexed satisfaction relations* $\models_{i_x;i_y;i_z} \alpha$ $(i_x, i_y, i_z \in \omega)$ *for any formula* α *are defined inductively by*

1. *for any propositional variable* p,
   $\models_{i_x;i_y;i_z} p$ *iff* $I^{i_x;i_y;i_z}(p) = t$,

[1] The proof of the syntactical embedding theorem in [2] has such an error.

2. $\models_{i_x;i_y;i_z} \alpha \wedge \beta$ *iff* $\models_{i_x;i_y;i_z} \alpha$ *and* $\models_{i_x;i_y;i_z} \beta$,
3. $\models_{i_x;i_y;i_z} \alpha \vee \beta$ *iff* $\models_{i_x;i_y;i_z} \alpha$ *or* $\models_{i_x;i_y;i_z} \beta$,
4. $\models_{i_x;i_y;i_z} \alpha \rightarrow \beta$ *iff not-*$(\models_{i_x;i_y;i_z} \alpha)$ *or* $\models_{i_x;i_y;i_z} \beta$,
5. $\models_{i_x;i_y;i_z} \neg\alpha$ *iff not-*$(\models_{i_x;i_y;i_z} \alpha)$,
6. $\models_{i_x;i_y;i_z} \mathrm{P}_x\alpha$ *iff* $\models_{i_x+1;i_y;i_z} \alpha$,
7. $\models_{i_x;i_y;i_z} \mathrm{P}_y\alpha$ *iff* $\models_{i_x;i_y+1;i_z} \alpha$,
8. $\models_{i_x;i_y;i_z} \mathrm{P}_z\alpha$ *iff* $\models_{i_x;i_y;i_z+1} \alpha$,
9. $\models_{i_x;i_y;i_z} \mathrm{A}\alpha$ *iff* $\models_{i_x+j_x;i_y+j_y;i_z+j_z} \alpha$ *for any* $j_x, j_y, j_z \in \omega$,
10. $\models_{i_x;i_y;i_z} \mathrm{S}\alpha$ *iff* $\models_{i_x+j_x;i_y+j_y;i_z+j_z} \alpha$ *for some* $j_x, j_y, j_z \in \omega$,
11. $\models_{i_x;i_y;i_z} \mathrm{A}^-\alpha$ *iff* $\models_{j_x;j_y;j_z} \alpha$ *for any* $j_x, j_y, j_z \in \omega$ *with* $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ *and* $0 \leq j_z \leq i_z$,
12. $\models_{i_x;i_y;i_z} \mathrm{S}^-\alpha$ *iff* $\models_{j_x;j_y;j_z} \alpha$ *for some* $j_x, j_y, j_z \in \omega$ *with* $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ *and* $0 \leq j_z \leq i_z$.

A formula α *is called* 3SL-valid *if* $\models_{0;0;0} \alpha$ *holds for any space-indexed satisfaction relations* $\models_{i_x;i_y;i_z}$ $(i_x, i_y, i_z \in \omega)$. *A sequent* $\Gamma \Rightarrow \Delta$ *is called 3SL-valid if so is the formula* $\Gamma_* \rightarrow \Delta^*$.

A semantics for LK_ω is defined below.

Definition 9. *Let* Θ *be a countable nonempty set of formulas. A valuation* I *is a mapping from the set of all propositional variables to the set* $\{t, f\}$ *of truth values. A satisfaction relation* $\models \alpha$ *for any formula* α *is defined inductively by*

1. $\models p$ *iff* $I(p) = t$ *for any propositional variable* p,
2. $\models \bigwedge \Theta$ *iff* $\models \alpha$ *for any* $\alpha \in \Theta$,
3. $\models \bigvee \Theta$ *iff* $\models \alpha$ *for some* $\alpha \in \Theta$,
4. $\models \alpha \rightarrow \beta$ *iff not-*$(\models \alpha)$ *or* $\models \beta$,
5. $\models \neg\alpha$ *iff not-*$(\models \alpha)$.

A formula α *is called* LK_ω-valid *if* $\models \alpha$ *holds for any satisfaction relation* $\models$. *A sequent* $\Gamma \Rightarrow \Delta$ *is called* LK_ω*-valid if so is the formula* $\Gamma_* \rightarrow \Delta^*$.

As well known, the following completeness theorem holds for LK_ω: For any sequent S, $\mathrm{LK}_\omega \vdash S$ iff S is LK_ω-valid.

Lemma 10. *Let* f *be the mapping defined in Definition 5. For any space-indexed satisfaction relation* $\models_{i_x;i_y;i_z}$ $(i_x, i_y, i_z \in \omega)$, *we can construct a satisfaction relation* $\models$ *such that for any formula* α *in* $\mathcal{L}^s$, $\models_{i_x;i_y;i_z} \alpha$ *iff* $\models f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\alpha)$.

Proof. Let Φ be a set of propositional variables and Φ_ι be the set $\{p_\iota \mid p \in \Phi\}$ of propositional variables. Suppose that $I^{i_x;i_y;i_z}$ $(i_x, i_y, i_z \in \omega)$ are mappings from Φ to $\{t, f\}$. Suppose that I is a mapping from $\bigcup_{\iota \in P^*} \Phi_\iota$ to $\{t, f\}$. Suppose moreover that $I^{i_x;i_y;i_z}(p) = t$ iff $I(p_\iota) = t$ where $\iota \equiv \mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}$. The lemma is then proved by induction on the complexity of α. Let $\iota \equiv \mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}$ and $\hat{\iota} \equiv i_x; i_y; i_z$.

• Base step:

Case ($\alpha \equiv p \in \Phi$): $\models_{\hat{\iota}} p$ iff $I^{\hat{\iota}}(p) = t$ iff $I(p_\iota) = t$ iff $\models p_\iota$ iff $\models f(\iota p)$ (by the definition of f).

• Induction step: We show some cases.

Case ($\alpha \equiv \alpha_1 \rightarrow \alpha_2$): $\models_{\hat{\iota}} \alpha_1 \rightarrow \alpha_2$ iff not-($\models_{\hat{\iota}} \alpha_1$) or $\models_{\hat{\iota}} \alpha_2$ iff not-($\models f(\iota\alpha_1)$) or $\models f(\iota\alpha_2)$ (by induction hypothesis) iff $\models f(\iota\alpha_1) \rightarrow f(\iota\alpha_2)$ iff $\models f(\iota(\alpha_1 \rightarrow \alpha_2))$ (by the definition of f).

Case ($\alpha \equiv \mathrm{P}_x\beta$): $\models_{i_x;i_y;i_z} \mathrm{P}_x\beta$ iff $\models_{i_x+1;i_y;i_z} \beta$ iff $\models f(\mathrm{P}_x^{i_x+1}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\beta)$ (by induction hypothesis) iff $\models f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\mathrm{P}_x\beta)$ (by the definition of f).

Case ($\alpha \equiv \mathrm{A}\beta$):

$\models_{i_x;i_y;i_z} \mathrm{A}\beta$
iff $\models_{i_x+j_x;i_y+j_y;i_z+j_z} \beta$ for any $j_x, j_y, j_z \in \omega$
iff $\models f(\mathrm{P}_1^{i_x+j_x}\mathrm{P}^{i_y+j_y}\mathrm{P}_z^{i_z+j_z}\beta)$ for any $j_x, j_y, j_z \in \omega$ (by induction hypothesis)
iff $\forall j_x, j_y, j_z \in \omega\ [\models f(\mathrm{P}_1^{i_x+j_x}\mathrm{P}_y^{i_y+j_y}\mathrm{P}_z^{i_z+j_z}\beta)]$
iff $\models \gamma$ for any $\gamma \in \{f(\mathrm{P}_x^{i_x+j_x}\mathrm{P}_y^{i_y+j_y}\mathrm{P}_z^{i_z+j_z}\beta) \mid j_x, j_y, j_z \in \omega\}$
iff $\models \bigwedge\{f(\mathrm{P}_x^{i_x+j_x}\mathrm{P}_y^{i_y+j_y}\mathrm{P}_z^{i_z+j_z}\beta) \mid j_x, j_y, j_z \in \omega\}$
iff $\models \bigwedge\{f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\mathrm{P}_x^{j_x}\mathrm{P}_y^{j_y}\mathrm{P}_z^{j_z}\beta) \mid j_x, j_y, j_z \in \omega\}$ (by the definition of f)
iff $\models f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\mathrm{A}\beta)$ (by the definition of f).

Case ($\alpha \equiv \mathrm{A}^-\beta$):

$\models_{i_x;i_y;i_z} \mathrm{A}^-\beta$
iff $\models_{j_x;j_y;j_z} \beta$ for any $j_x, j_y, j_z \in \omega$ with $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ and $0 \leq j_z \leq i_z$
iff $\models f(\mathrm{P}_1^{j_x}\mathrm{P}^{j_y}\mathrm{P}_z^{j_z}\beta)$ for any $j_x, j_y, j_z \in \omega$ with $0 \leq j_x \leq i_x$, $0 \leq j_y \leq i_y$ and $0 \leq j_z \leq i_z$ (by induction hypothesis)
iff $\models \gamma$ for any $\gamma \in \{f(\mathrm{P}_x^{j_x}\mathrm{P}_y^{j_y}\mathrm{P}_z^{j_z}\beta) \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\}$
iff $\models \bigwedge\{f(\mathrm{P}_x^{j_x}\mathrm{P}_y^{j_y}\mathrm{P}_z^{j_z}\beta) \mid 0 \leq j_x \leq i_x, 0 \leq j_y \leq i_y, 0 \leq j_z \leq i_z\}$
iff $\models f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\mathrm{A}^-\beta)$ (by the definition of f).

Q.E.D.

Lemma 11. *Let f be the mapping defined in Definition 5. For any satisfaction relation $\models$, we can construct a space-indexed satisfaction relation $\models_{i_x;i_y;i_z}$ such that for any formula α in $\mathcal{L}^s$, $\models f(\mathrm{P}_x^{i_x}\mathrm{P}_y^{i_y}\mathrm{P}_z^{i_z}\alpha)$ iff $\models_{i_x;i_y;i_z} \alpha$.*

Proof. Similar to the proof of Lemma 10. **Q.E.D.**

Theorem 12 (Semantical embedding). *Let f be the mapping defined in Definition 5. For any formula α in $\mathcal{L}^s$, α is 3SL-valid iff $f(\alpha)$ is LK_ω-valid.*

Proof. By Lemmas 10 and 11. We take 0 for i_x, i_y and i_z. **Q.E.D.**

Theorem 13 (Completeness). *For any sequent S, $3\mathrm{SL} \vdash S$ iff S is 3SL-valid.*

Proof. Let $\Gamma \Rightarrow \Delta$ be S and α be $\Gamma_* \rightarrow \Delta^*$. It is sufficient to show that $3\mathrm{SL} \vdash \Rightarrow \alpha$ iff α is 3SL-valid. We show this as follows. $3\mathrm{SL} \vdash \Rightarrow \alpha$ iff $\mathrm{LK}_\omega \vdash \Rightarrow f(\alpha)$ (by Theorems 6 and 7) iff $f(\alpha)$ is LK_ω-valid (by the completeness theorem for LK_ω) iff α is 3SL-valid (by Theorem 12). **Q.E.D.**

4 Concluding Remarks

In this paper, the new spatial logic 3SL, which can appropriately represent the 3-dimensional space ω^3, was introduced as a Gentzen-type sequent calculus, and the cut-elimination and completeness theorems for 3SL were proved using an embedding-based technique. 3SL is a natural extension and generalization of a sequent calculus LT_ω for LTL. The proposed framework could be extended to the n-dimensional space ω^n, and hence a 4-dimensional *spatio-temporal logic*, i.e., three dimensions are devoted to space and one dimension is devoted to time, could also be formalized naturally.

We remark that the decision problem for 3SL has not been solved yet. But, we can show the CoNP-completeness for the $\{\mathrm{A}, \mathrm{S}\}$-free fragment of 3SL.

Theorem 14 (Decidability). *The* $\{\mathrm{A}, \mathrm{S}\}$*-free fragment of* 3SL *is CoNP-complete.*

Proof. (Sketch). Let L be the $\{\mathrm{A}, \mathrm{S}\}$-free fragment of 3SL. Consider the mapping f defined in Definition 5. By f, we can embed L into classical propositional logic (CL) since we do not need the infintary conjunction and the infinitary disjunction which are the counterparts of A and S, respectively. CL is known to be CoNP-complete. By decidability of CL, for each α, it is possible to decide if $f(\alpha)$ is valid in L. Then, by the theorem for embedding L into CL, L is decidable. Since f is a polynomial-time translation, L is also CoNP-complete. **Q.E.D.**

Acknowledgments. This work was supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 20700015.

References

1. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Formal Models and Semantics (B), pp. 995–1072. Elsevier, MIT Press (1990)
2. Kamide, N.: Embedding linear-time temporal logic into infinitary logic: Application to cut-elimination for multi-agent infinitary epistemic linear-time temporal logic. In: Fisher, M., Sadri, F., Thielscher, M. (eds.) CLIMA IX. LNCS (LNAI), vol. 5405, pp. 57–76. Springer, Heidelberg (2009)
3. Kawai, H.: Sequential calculus for a first order infinitary temporal logic. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 33, 423–432 (1987)
4. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57 (1977)

Intuitionistic Fuzzy Probability

Claudilene G. Da Costa[1], Benjamin C. Bedregal[2], and Adrião D. Doria Neto[3]

[1] Departamento de Ciências Exatas – DCE
Universidade Federal da Paraíba – UFPB
Rio Tinto-PB, Brasil
[2] Grupo de Lógica, Linguagens, Informação, Teoria e Aplicações – LoLITA
Departamento de Informática e Matemática Aplicada – DIMAp
Universidade Federal do Rio Grande do Norte – UFRN
Campus Universitário s/n, 59072-970 Natal, Brazil
[3] Departamento de Engenharia de Computação e Automação – DCA
Universidade Federal do Rio Grande do Norte – UFRN
Campus Universitário s/n, 59072-970 Natal, Brazil
claudilene@ccae.ufpb.br, bedregal@dimap.ufrn.br, adriao@dca.ufrn.br

Abstract. Fuzzy Probabilities are an extension of the concept of probabilities with application in several practical problems. The former are probabilities represented through fuzzy numbers, to indicate the uncertainty in the value assigned to a probability. Moreover, Krassimir Atanassov in 1983 added an extra degree of uncertainty to classic fuzzy sets for modeling the hesitation and uncertainty about the degree of membership. This new theory of fuzzy sets is known today as intuitionistic fuzzy set theory.

This work will extend the notion of fuzzy probabilities by representing probabilities through the intuitionistic fuzzy numbers, in the sense of Atanassov, instead of fuzzy numbers.

1 Introduction

After the introduction of the concept of fuzzy sets by Lotfi Zadeh in [33], several surveys were conducted on possible extensions of the concept of fuzzy set. Among these extensions, one that has called the attention of much research in recent decades is the intuitionistic fuzzy set (IFS) theory, introduced by Krassimir Atanassov in 1983 [1]. The word "intuitionistic" is used here in a "broad" sense to refer to the fact that the law of the excluded middle on the element level is denied (since $\mu_A(x) + \nu_A(x) < 1$ is possible) [11]. Intuitionistic fuzzy logic in a "narrow" sense has been proposed by Takeuti and Titani [30]. This is mainly due to the fact that IFS is consistent with human behavior. IFS add an extra degree to the fuzzy sets in order to modeling the hesitation and uncertainty about the degree of membership. In fuzzy set theory the hesitation degree (or degree of non-membership) of an element of the universe is implicitly defined as one minus the degree of membership, and therefore is fixed. In the IFS theory the degree of hesitation, is somehow independent.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 273–282, 2010.

Since the IFS theory is a generalization of the fuzzy set theory, it is natural to expect that most of the concepts and intrinsic properties of the fuzzy set theory have a counterpart in the IFS theory. One of these, which were generalized to IFS, are the notions of t-norms and t-conorm [13,14].

On the other hand, probability theory, is an old uncertainty method which is appropriate to deal with another kind of uncertainty. However, since probabilities consider an absolute knowledge, and in many real situations these knowledge are partially known or uncertain, there were several ways to extend the notion of Probabilities, to deal with such situations. Among them we highlight the interval ([21,9,31]) and fuzzy. There are in the literature several different approaches of Fuzzy Probabilities (see for example, [32,34,15,5]), we highlight [5]. For James Buckley, the probabilities of events, in practice, should be known exactly, however, many times these values are estimated or provided by experts, and therefore are of vague nature. He modeled this vagueness using fuzzy numbers. This approach have been applied in several subjects (see [5,6]).

This article introduces a generalization of the concept of Fuzzy probabilities representing probabilities as in [5] by using an original notion of intuitionistic fuzzy numbers instead of usual fuzzy numbers. Thus, we give an original generalization, in the context of IFS, to the fuzzy probability approach of James Buckley in [5] and so, we introduce a theory to deal with probabilities in a framework where it does not only model uncertainty in the probability of some events but also model the hesitation which is naturally present in the uncertainty.

In section 2 are given the basic notions of intuitionistic fuzzy sets and intuitionistic fuzzy numbers that are fundamental for the paper. Note that this notion of intuitionistic fuzzy numbers is new. In section 3, are provided the main definitions and results of this work. In section 4 is introduced an intuitionistic fuzzy extension of (fuzzy) conditional probability and it is proved that this extension satisfies analogous properties than conditional probabilities. Notice that all results in this last two sections are new.

We assume that the reader is familiar with fuzzy logic, probability and their fuzzy versions. For more details on these theories see [5] and for intuitionistic fuzzy set theory see [2,26,8,27].

2 Intuitionistic Fuzzy Numbers

An intuitionistic fuzzy set (IFS) A in a universe X is

$$A = \{(x, \mu_A(x), \nu_A(x)) \mid x \in X\}$$

where $\mu_A, \nu_A : X \to [0,1]$ satisfying the condition $\mu_A(x) + \nu_A(x) \leq 1$. $\mu_A(X)$ and $\nu_A(x)$ denote, respectively, the membership degree and the non-membership degree of the element x in the set A.

Glad Deschrijver and Etiene Kerre in [12] give an alternative approach to IFS, they proved that IFS can also be seen as a L^*-valued fuzzy set in the sense of Joseph Goguen [18] for consider the complete lattice $\langle L^*, \leq_{L^*} \rangle$ where

$$L^* = \{(x, y) \in [0,1] \times [0,1] \mid x + y \leq 1\}$$

and

$$(x_1, x_2) \leq_{L^*} (y_1, y_2) \text{ if and only if } x_1 \leq y_1 (and) x_2 \geq y_2.$$

Note that $0_{L^*} = (0.1)$ and $1_{L^*} = (1.0)$. Thus, IFS are nothing more than L^*-valued fuzzy sets.

Given a subset $A \subseteq L^*$ its supremum and ínfimum, with respect to $\leq_{L^*}$, could be obtained as follows:

$$\sup A = (\sup\{\pi_1(x) \mid x \in A\}, \inf\{\pi_2(x) \mid x \in A\})$$
$$\inf A = (\inf\{\pi_1(x) \mid x \in A\}, \sup\{\pi_2(x) \mid x \in A\})$$

where $\pi_1(x_1, x_2) = x_1$ and $\pi_2(x_1, x_2) = x_2$.

Given $(\alpha, \beta) \in L^*$and an IFS A of universe X, the (α, β)-cut of A is the set

$$A_{(\alpha,\beta)} = \{x \in X \mid \mu_A(x) \geq \alpha \text{ and } \nu_A(x) \leq \beta\}. \tag{1}$$

Notice that $A_{(\alpha,\beta)} = A_\alpha \cap A^\beta$ where $A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}$ and $A^\beta = \{x \in X \mid \nu_A(x) \leq \beta\}$.

Notice that every IFS can be recovered from its (α, β)-cuts. In fact

$$(\mu_A(x), \nu_A(x)) = \sup\{(\alpha, \beta) \in L^* \mid x \in A_{(\alpha,\beta)}\}$$

where the supremum is with respect to $\leq_{L^*}$.

Definition 2.1. *Let A be an IFS with universe $\mathbb{R}$. A is **convex** if every (α, β)-cut of A is convex, i.e., intervals of real numbers. A is **normalized** if there exists $x \in \mathbb{R}$ such that $(\mu_A(x), \nu_A(x)) = 1_{L^*}$.*

Thus, for the convex IFS, its (α, β)-cuts are closed intervals of real numbers, and so it is possible to represent them via their end points, which can be obtained as follows:

$$l(A_{(\alpha,\beta)}) = \max(\min \mu_A^{-1}(\alpha), \min \nu_A^{-1}(\beta)) \text{ and} \tag{2}$$
$$r(A_{(\alpha,\beta)}) = \min(\max \mu_A^{-1}(\alpha), \max \nu_A^{-1}(\beta)). \tag{3}$$

That is, $A_{(\alpha,\beta)} = [l(A_{(\alpha,\beta)}), r(A_{(\alpha,\beta)})]$. Since A is convex, $|\nu_A^{-1}(\beta)|, |\mu_A^{-1}(\alpha)| \leq 2$.

Definition 2.2. *An IFS A with the real universe is an **intuitionistic fuzzy number** (IFN) if it is convex, normalized and μ_A and ν_A are continuous by parts. Denote by $\mathcal{N}$ the set of all IFN.*

Notice that this definition is different from other proposals of intuitionistic fuzzy numbers in the literature, as for example [7,25,17,3,28,19].

An important subclass of IFN are those in which μ_A, as well as its complement ν_A, have a triangular shape, that is, are triangular fuzzy numbers in the usual sense (TFN), thats why they will be called triangular intuitionistic fuzzy numbers (TIFN). Thus, as can be seen in figure 1, these IFN are completely determined by the values that characterize the TIFN, μ_A and ν_A^c, where $\nu_A^c(x) = 1 - \nu_A(x)$.

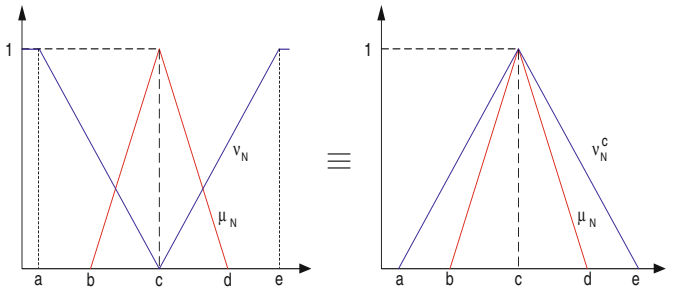


Fig. 1. Triangular intuitionistic fuzzy numbers

Thus, a TIFN like figure 1, will be denoted by $(a, b/c/d, e)$. One advantage of TIFN with respect to any IFS, is that its (α, β)-cuts can be easily determined as follows:

$$(a, b/c/d, e)_{(\alpha,\beta)} = [\max(a + (c-a)\alpha, b + (c-b)\beta), \min(e + (e-c)\alpha, d + (d-c)\beta)] \quad (4)$$

Let A and B be two IFN. Then define the addition, subtraction and multiplication of A with B from the corresponding interval arithmetic operations on their (α, β)-cuts. Let $(\alpha, \beta) \in L^*$,

$$\begin{aligned}(A + B)_{(\alpha,\beta)} &= \{x + y \mid x \in A_{(\alpha,\beta)} \text{ and } y \in B_{(\alpha,\beta)}\} \\ &= [l(A_{(\alpha,\beta)}) + l(B_{(\alpha,\beta)}), r(A_{(\alpha,\beta)}) + r(B_{(\alpha,\beta)})]\end{aligned}$$

$$\begin{aligned}(A - B)_{(\alpha,\beta)} &= \{x - y \mid x \in A_{(\alpha,\beta)} \text{ and } y \in B_{(\alpha,\beta)}\} \\ &= [l(A_{(\alpha,\beta)}) - r(B_{(\alpha,\beta)}), r(A_{(\alpha,\beta)}) - l(B_{(\alpha,\beta)})]\end{aligned}$$

$$\begin{aligned}(A \cdot B)_{(\alpha,\beta)} &= \{x \cdot y \mid x \in A_{(\alpha,\beta)} \text{ and } y \in B_{(\alpha,\beta)}\} \\ &= [\min S, \max S]\end{aligned}$$

where $S = \{l(A_{(\alpha,\beta)}) \cdot l(B_{(\alpha,\beta)}), l(A_{(\alpha,\beta)}) \cdot r(B_{(\alpha,\beta)}), r(A_{(\alpha,\beta)}) \cdot l(B_{(\alpha,\beta)}), r(A_{(\alpha,\beta)}) \cdot r(B_{(\alpha,\beta)})\}$.

Notice that for all $r \in \mathbb{R}$, we can define the following IFN, $\overline{r} = \{(r, 1, 0)\} \cup \{(x, 0, 1) \mid x \in \mathbb{R} \text{ and } x \neq r\}$. This kind of IFN are called crisp.

Proposition 2.3. *Let A and B be two IFN. Then, $A + B$, $A - B$ and $A \cdot B$ are IFN also.*

It is possible to establish several orders on fuzzy numbers (there are more than 40 in the literature) [5] which could be extended to IFN. Here we use the following (new)order on IFN:

$$A \preceq B \Leftrightarrow A_{(\alpha,\beta)} \trianglelefteq B_{(\alpha,\beta)} \text{ for all } (\alpha, \beta) \in L^* \quad (5)$$

where,

$$A_{(\alpha,\beta)} \trianglelefteq B_{(\alpha,\beta)} \Leftrightarrow r(A_{(\alpha,\beta)}) < r(B_{(\alpha,\beta)}) \text{ or } (r(A_{(\alpha,\beta)}) = r(B_{(\alpha,\beta)}) \text{ and } (l(B_{(\alpha,\beta)}) \leq l(A_{(\alpha,\beta)}))$$

Notice that $\trianglelefteq$ is a total order on $\mathbb{IR}$, the set of real intervals. Notice also that if $A, B \in \mathbb{IR}$, then $A \trianglelefteq B$ if and only if $A \subseteq B$ or $A \ll B$, where $\ll$ is the "way below" order defined on $\mathbb{IR}$ (as in domain theory), that is

$$[a,b] \ll [c,d] \text{ if and only if } a < c \text{ and } b < d$$

3 Intuitionistic Fuzzy Probability

Let $X = \{x_1, \ldots, x_n\}$ be a finite set. Let $\mathcal{N}$ be the set of all IFN and let $\overline{a_i} \in \mathcal{N}$, $i = 1, \ldots, n$, with $\overline{0} \preceq \overline{a_i} \preceq \overline{1}$, for all $i = 1, \ldots, n$, a family of fixed IFN such that there are $a_i \in \overline{(a_i)}_{1_{L^*}}$ with $\sum_{l=1}^{n} a_l = 1$. For each $(\alpha, \beta) \in L^*$ denote $(\overline{a_1})_{(\alpha,\beta)} \times \ldots \times (\overline{a_n})_{(\alpha,\beta)}$ by S_α^β. Define also, for each $A \subseteq X$ and $(\alpha, \beta) \in L^*$, the following set:

$$(\overline{P}(A))_{(\alpha,\beta)} = \left\{ \sum_{i \in I_A} a_i \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\} \tag{6}$$

where I_A is the set of indexes of A, that is, $I_A = \{i \in \{1, \ldots, n\} |\ x_i \in A\}$. Conventionally $\sum_{i \in \emptyset} a_i = 0$.

Theorem 3.1. *For all $A \subseteq X$,$(\overline{P}(A))_{(\alpha,\beta)}$ are the (α, β)-cuts of an IFN, $\overline{P}(A)$, that is, $\overline{P} : \wp(X) \to \mathcal{N}$.*

Proof: Let $S = \{(y_1, \ldots, y_n) \in [0,1]^n \mid \sum_{i=1}^{n} y_i = 1\}$. For each $(\alpha, \beta) \in L^*$, define $Dom[\alpha, \beta] = S \cap \prod_{i=1}^{n} \overline{(a_i)}_{(\alpha,\beta)}$ and $f : Dom[\alpha, \beta] \to [0,1]$ by

$$f(a_1, \ldots, a_n) = \sum_{i \in I_A} a_i. \tag{7}$$

We have that f is continuous and $Dom[\alpha, \beta]$ is connected, bounded and closed. Hence the image of f is a bounded and closed interval of $\mathbb{R}$. For each $(\alpha, \beta) \in L^*$ define $\Gamma[\alpha, \beta] = f(Dom[\alpha, \beta])$.

By equation (6) we have that, for each $(\alpha, \beta) \in L^*$, $(\overline{P}(A))_{(\alpha,\beta)} = \Gamma[\alpha, \beta]$. Therefore, $\overline{P}(A)$ is an IFN since $(\overline{P}(A))_{1_{L^*}} \neq \emptyset$. ■

$\overline{P}$ is called **intuitionistic fuzzy probability function**.

Theorem 3.2. *Let $X = \{x_1, \ldots, x_n\}$ and let $\overline{P} : \wp(X) \to \mathcal{N}$ be an intuitionistic fuzzy probability function. Then, for each $A, B \subseteq X$ the following properties hold:*

1. *If* $A \cap B = \emptyset$ *then* $\overline{P}(A \cup B) \preceq \overline{P}(A) + \overline{P}(B)$
2. *If* $A \subseteq B$ *then* $\overline{P}(A) \preceq \overline{P}(B)$
3. $\overline{P}(\emptyset) = \overline{0} \preceq \overline{P}(A) \preceq \overline{1} = \overline{P}(X)$.
4. $\overline{1} \preceq \overline{P}(A) + \overline{P}(A^c)$
5. *If* $A \cap B \neq \emptyset$ *then* $\overline{P}(A \cup B) \preceq \overline{P}(A) + \overline{P}(B) - \overline{P}(A \cap B)$

Proof: Items 2. and 3. hold easily and 4. follows by 1. Hence, we will only prove items 1 and 5. By definition of the order of IFN (Equation (5)), in order to prove item 1. it is sufficient to prove that

$$\overline{P}(A \cup B)_{(\alpha,\beta)} \subseteq \overline{P}(A)_{(\alpha,\beta)} + \overline{P}(B)_{(\alpha,\beta)}. \tag{8}$$

Notice that $A \cap B = \emptyset$ if and only if $I_A \cap I_B = \emptyset$.

To prove equation (8) it is sufficient to prove that for each $(\alpha, \beta) \in L^*$,

$$\left\{ \sum_{i \in I_A} a_i + \sum_{j \in I_B} a_j \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$

lies in

$$\left\{ \sum_{i \in I_A} a_i \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\} \; + \; \left\{ \sum_{j \in I_B} a_j \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\},$$

which is obvious.

To prove item 5., it is sufficient to check, by an argument similar to the previous one, that

$$\left\{ \sum_{i \in I_A \cup I_B} a_i \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$

lies in

$$\left\{ \sum_{i \in I_A} a_i | (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\} + \left\{ \sum_{j \in I_B} a_j | (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$
$$- \left\{ \sum_{i \in I_A \cap I_B} a_i \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$

which proves that

$$\overline{P}(A \cup B)_{(\alpha,\beta)} \subseteq \overline{P}(A)_{(\alpha,\beta)} + \overline{P}(B)_{(\alpha,\beta)} - \overline{P}(A \cap B)_{(\alpha,\beta)}$$

and the result follows by definition of $\preceq$. ∎

4 Intuitionistic Fuzzy Conditional Probability

Let $A, B \subseteq X$ with I_A and I_B being their set of index, respectively. Define the intuitionistic fuzzy conditional probability of A with B as being the IFS $\overline{P}(A \mid B)$ whose (α, β)-cuts are:

$$\overline{P}(A \mid B)_{(\alpha,\beta)} = \left\{ \frac{\sum\limits_{i \in I_A \cap I_B} a_i}{\sum\limits_{j \in I_B} a_j} \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}. \quad (9)$$

Theorem 4.1. *$\overline{P}(A \mid B)$ is an IFN.*

Proof: Analogous to the proof of Theorem 3.1 by replacing the function f in equation (7) by

$$f(a_1, \ldots, a_n) = \frac{\sum\limits_{i \in I_A \cap I_B} a_i}{\sum\limits_{j \in I_B} a_j}$$

■

Theorem 4.2. *Let $X = \{x_1, \ldots, x_n\}$ and let $\overline{P} : \wp(X) \to \mathcal{N}$ be an intuitionistic fuzzy conditional probability function. Then, for each $A, B \subseteq X$ the following properties hold:*

1. *If $A_1 \cap A_2 = \emptyset$ then $\overline{P}(A_1 \cup A_2 \mid B) \preceq \overline{P}(A_1 \mid B) + \overline{P}(A_2 \mid B)$*
2. *$\overline{0} \preceq \overline{P}(A \mid B) \preceq \overline{1}$.*
3. *$\overline{P}(A \mid A) = \overline{1}$.*
4. *If $B \subseteq A$ then $\overline{P}(A \mid B) = \overline{1}$.*
5. *If $A \cap B = \emptyset$ then $\overline{P}(A \mid B) = \overline{0}$.*

Proof: The proof of items 2,3,4 and 5, is trivial.

Notice that if $A_1 \cap A_2 = \emptyset$ then $I_{A_1} \cap I_{A_2} = \emptyset$. To prove item 1, it is sufficient to prove that for each $(\alpha, \beta) \in L^*$,

$$\overline{P}(A_1 \cup A_2 \mid B)_{(\alpha,\beta)} \subseteq \overline{P}(A_1 \mid B)_{(\alpha,\beta)} + \overline{P}(A_2 \mid B)_{(\alpha,\beta)}.$$

But this follows by the fact that

$$\overline{P}(A_1 \cup A_2 \mid B)_{(\alpha,\beta)} = \left\{ \frac{\sum\limits_{i \in I_{A_1} \cap I_B} a_i + \sum\limits_{i \in I_{A_2} \cap I_B} a_i}{\sum\limits_{j \in I_B} a_j} \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$

lies in

$$\left\{ \frac{\sum\limits_{i \in I_{A_1} \cap I_B} a_i}{\sum\limits_{j \in I_B} a_j} \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\} +$$

$$\left\{ \frac{\sum\limits_{i \in I_{A_2} \cap I_B} a_i}{\sum\limits_{j \in I_B} a_j} \mid (a_1, \ldots, a_n) \in S_\alpha^\beta \text{ and } \sum_{l=1}^{n} a_l = 1 \right\}$$

which is obvious.

Notice that the last term in the above equality coincides with $\overline{P}(A_1 \mid B)_{(\alpha,\beta)} + \overline{P}(A_2 \mid B)_{(\alpha,\beta)}$. ■

5 Final Remark

This work is an extension of the approach of James Buckley [5] to fuzzy probability, since all fuzzy sets (and thus fuzzy numbers and fuzzy probability) is an intuitionistic fuzzy set (with $\nu_A(x) = 1 - \mu_A(x)$). However, this extension consider an order for IFN essentially different from the order considered by Buckley for fuzzy numbers, in the sense that when a fuzzy number is seen as an IFN (making $\nu_A(x) = 1 - \mu_A(x)$) the order of Buckley does not match the order $\preceq$ introduced in this work.

Observe that our approach for intuitionistic fuzzy probability is substantially different of the approach in [16,10], where is defined a probability theory for intuitionistic fuzzy events instead of a probability theory where the probabilities are IFN.

This extension may be useful in some situations where some probabilities are uncertain, but where this uncertainty can be modeled using intuitionistic fuzzy numbers. For example, if $\overline{p}$ is an intuitionistic fuzzy probability (an IFN supported in the range $[0, 1]$) then each probability p will approach the uncertain outlook described by $\overline{p}$ with some degree of certainty, but also have the information of when we are uncertain about this approach. A concrete application can be given in Football Match Result Predictions. An approach for this subject is by using classical probability (see for example [24,29,22,4,20]). Nevertheless, in [23] it is used an intuitionistic fuzzy approach to model the win expectancy and answers the question "Do we expect the home team to win?". For Kolev et al, the win expectancy w_e, with intuitionistic degree $(\mu(w_e), \nu(w_e))$, can be used for predictions, once their system claims that the match result will be most probably in the interval $[\mu(w_e), 1 - \nu(w_e)]$.

As a future work we intend, using this same approach, to generalize the notion of probability spaces as well as other concepts related to probability (eg probability involving random variables and Bayes' formula). We also expect to extend the notions of Markov chains, hidden Markov models, etc.. Finally, we hope to apply this new theory in real situations as has been applied the fuzzy probability in [5,6].

References

1. Atanassov, K.T.: Intuitionistic Fuzzy Sets. Central Tech. Library, Bulgarian Academy Science, Sofia, Bulgaria, Rep. No. 1697/84 (1983)
2. Atanassov, K.T.: Intuitionistic Fuzzy Sets: Theory and Applications. Springer, Heidelberg (1999)
3. Ban, A.I.: Nearest Interval Approximation of an Intuitionistic Fuzzy Number. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications, pp. 229–240. Springer, Heidelberg (2006)
4. Buchdahl, J.: Fixed Odds Sports Betting: Statistical Forecasting and Risk Management. High Stakes Publishing (2003)
5. Buckley, J.J.: Fuzzy Probabilities: A new approach and applications. Springer, Berlin (2005)

6. Buckley, J.J.: Fuzzy Probabilities and Statistics. Springer, Berlin (2006)
7. Burillo, P., Bustince, H., Mohedano, V.: Some definitions of intuitionistic fuzzy number. First properties. In: Lakov, D. (ed.) Proceedings of the 1st Workshop on Fuzzy Based Expert Systems, Sofia, Bulgaria, pp. 53–55 (September 1994)
8. Bustince, H., Montero, J., Orduna, R., Barrenechea, E., Pagola, M.: A survey of Atanassov's Intuitionistic Fuzzy Relations. In: Intuitionistic Fuzzy Sets: Recent Advances. Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2008)
9. Campos, M.A., Dimuro, G.P., Costa, A.C.R., Araújo, J.F., Dias, A.M.: Probabilidade Intervalar e Cadeias de Markov Intervalares no Maple. Tema – Tendências em Matemática Aplicada e Computacional 3(2), 53–62 (2002)
10. Ciungu, L.C., Riecan, B.: Representation theorem for probabilities on IFS-events. Information Sciences 180, 793–798 (2010)
11. Cornelis, C., Deschrijver, G., Kerre, E.E.: Implication in intuitionistic fuzzy and interval-valued fuzzy set theory: construction, classification, application. International Journal of Approximate Reasoning 35, 55–95 (2004)
12. Deschrijver, G., Kerre, E.E.: On the Relationship Between some Extensions of Fuzzy Set Theory. Fuzzy Sets and Systems 133(2), 227–235 (2003)
13. Deschrijver, G., Cornelis, C., Kerre, E.E.: Intuitionistic Fuzzy Connectives Revised. In: Proc. 9th Int. Conf. Information Processing Management Uncertainty Knowledge-based Systems, pp. 1839–1844 (2002)
14. Deschrijver, G., Cornelis, C., Kerre, E.E.: On the Representation of Intuitionistic Fuzzy t-Norms and t-Conorms. IEEE Trans. on Fuzzy Systems 12(1), 45–61 (2004)
15. Dunyak, J., Saad, L.W., Wunsch, D.: A Theory of Independent Fuzzy Probability for System Reliability. IEEE Trans. Fuzzy Systems 7, 286–294 (1999)
16. Grzegorzewski, P., Mrowka, E.: Probability of intuitionistic fuzzy events. In: Grzegorzewski, P., et al. (eds.) Soft Methods in Probability, Statistics and Data Analysis, pp. 105–115. Physica-Verlag, NewYork (2002)
17. Grzegorzewski, P.: Distances and orderings in a family of intuitionistic fuzzy numbers. In: Proc. of EUSFLAT Conf. 2003, pp. 223–227 (2003)
18. Goguen, J.A.: L-fuzzy sets. Journal of Mathematical Analysis and Applications 18(1), 623–668 (1967)
19. Guha, D., Chakraborty, D.: A Theoretical Development of Distance Measure for Intuitionistic Fuzzy Numbers. International Journal of Mathematics and Mathematical Sciences 2010, Article ID 949143, 25 pages
20. Haigh, J.: Taking Chances: Winning with Probability. Oxford University Press, Oxford (2003)
21. Hall, J.W., Blockley, D.I., Davis, J.P.: Uncertain Inference Using Interval Probability Theory. Approximate Reasoning 19, 247–264 (1998)
22. Hirotsu, N., Wright, M.: Using a Markov process model of an association football match to determine the optimal timing of substitution and tactical decisions. Journal of the Operational Research Society 53, 88–96 (2002)
23. Kolev, B., Chountas, P., Petrounias, I., Kodogiannis, V.: An Application of Intuitionistic Fuzzy Relational Databases in Football Match Result Predictions. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications, pp. 281–289. Springer, Heidelberg (2005)
24. Lee, A.J.: Modeling scores in Premier League: is Manchester United really the best. Chance 10, 15–19 (1997)
25. Liu, H., Shi, K.: Intuitionistic fuzzy numbers and intuitionistic distribution numbers. Journal of Fuzzy Mathematics 8(4), 909–918 (2000)

26. Pankowska, A., Wygralak, M.: Intuitionistic fuzzy sets – An alternative approach. In: EUSFLAT 2003 Proceedings, Zittau, Germany, September 10-12, pp. 135–140 (2003)
27. Reconvá, M., Riecan, B.: Observables on Intuitionistic Fuzzy Sets: An Elementary Approach. Cybernetics and Information Technologes 9(2), 38–42 (2009)
28. Nayagam, V.L.G., Venkateshwari, G., Sivaraman, G.: Ranking of intuitionistic fuzzy numbers. In: Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008), Hong Kong, pp. 1971–1974 (June 2008)
29. Rue, H., Salvesen, Ø.: Prediction and retrospective analysis of soccer matches in a league. The Statistician 49(3), 399–418 (2000)
30. Takeuti, G., Titani, S.: Intuitionistic fuzzy logic and intuitionistic fuzzy set theory. Journal of Symbolic Logic 49(3), 851–866 (1984)
31. Weichselberger, K.: The Theory of Interval-Probability as a Unifying Concept for Uncertainty. Approximate Reasoning 24, 149–170 (2000)
32. Yager, R.R.: Probabilities from fuzzy observations. Information Science 32(1), 1–31 (1984)
33. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
34. Zadeh, L.A.: Fuzzy Probabilities. Information Processing and Management 20, 363–372 (1984)

A Proof System for Temporal Reasoning with Sequential Information

Norihiro Kamide

Waseda Institute for Advanced Study, Waseda University,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, Japan
logician-kamide@aoni.waseda.jp

Abstract. A new logic, sequence-indexed linear-time temporal logic (SLTL), is obtained semantically from the standard linear-time temporal logic LTL by adding a sequence modal operator which represents a sequence of symbols. By the sequence modal operator of SLTL, we can appropriately express "sequential information" in temporal reasoning. A Gentzen-type sequent calculus for SLTL is introduced, and the completeness and cut-elimination theorems for this calculus are proved. SLTL is also shown to be PSPACE-complete and embeddable into LTL.

1 Introduction

It is known that *linear-time temporal logic* (LTL) [1,4] is one of the most useful logics for temporal reasoning and system verification. In this paper, a new logic, *sequence-indexed linear-time temporal logic* (SLTL), is obtained from LTL by adding a sequence modal operator which represents a sequence of symbols. By the sequence modal operator of SLTL, we can appropriately express "sequential information" in temporal reasoning.

The sequence modal operator in temporal reasoning was studied by Kamide and Kaneiwa [2] based on a *branching-time temporal logic*, CTLS*. This logic can suitably represent hierarchical tree structures where the sequence modal operator of CTLS* is applied to tree structures. CTLS* is, however, not appropriate for obtaining a proof-theoretical basis for temporal reasoning with sequential information. Indeed, it is difficult to construct a good proof system for CTLS* since the branching-time formalism is not suitable for obtaining a simple proof theory. A proof system such as a *Gentzen-type sequent calculus* with completeness and cut-elimination theorems has been required for developing foundations of automated temporal theorem proving with sequential information.

In the present paper, we improve the demerit for CTLS*, i.e., it has no good proof theory. Compared with CTLS*, the logic SLTL introduced in this paper has the following advantages:

1. SLTL has a natural and simple Gentzen-type sequent calculus, SLT$_\omega$,
2. the completeness and cut-elimination theorems for SLT$_\omega$ hold.

Thus, SLTL allows us to obtain a good proof theory for temporal reasoning with sequential information.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 283–292, 2010.

The reason of underlying the use of the notion of "sequences" in the sequence modal operator is explained below. The notion of "sequences" is fundamental to practical reasoning in computer science, because it can appropriately represent "data sequences," "program-execution sequences," "action sequences," "time sequences," "word (character or alphabet) sequences," "DNA sequences" etc. The notion of sequences is thus useful to represent the notions of "information," "attributes," "trees," "orders," "preferences," "strings," "vectors," and "ontologies." "Sequential information" can appropriately be represented by sequences, because a sequence structure gives a *monoid* $\langle M, ;, \emptyset \rangle$ with *informational interpretation* [6]:

1. M is a set of pieces of (sequential, ordered or prioritized) information (i.e., a set of sequences),
2. ; is a binary operator (on M) that combines two pieces of information (i.e., a concatenation operator on sequences),
3. $\emptyset$ is an empty piece of information (i.e., the empty sequence).

The sequence modal operator $[b]$, which was studied in [2], represents labels as "sequential information." A formula of the form $[b_1 \ ; \ b_2 \ ; \cdots ; \ b_n]\alpha$ intuitively means that "α is true based on a sequence $b_1 \ ; \ b_2 \ ; \cdots ; \ b_n$ of information pieces." Further, a formula of the form $[\emptyset]\alpha$, which coincides with α, intuitively means that "α is true without any information (i.e., it is an eternal truth in the sense of classical logic)." Simple and intuitive satisfaction relations by indexing sequences are required to formalize the sequence modal operator. These satisfaction relations are regarded as natural extensions of the standard satisfaction relation of classical logic. The proposed satisfaction relations, denoted as $\models^{\hat{d}}$, are indexed by a sequence $\hat{d}$, and the special case $\models^{\emptyset}$ corresponds to the classical satisfaction relation. Then, $\models^{\hat{d}} \alpha$ means that "α is true based on a sequence $\hat{d}$ of information pieces" and $\models^{\emptyset} \alpha$ means that "α is eternally true without any information."

The contents of this paper are then summarized as follows. In Section 2, SLTL is introduced semantically, and a Gentzen-type sequent calculus, SLT_ω, is constructed for SLTL. In Section 3, some theorems for embedding SLTL into LTL are proved, and by using these embedding theorems, the cut-elimination and completeness theorems for SLT_ω are shown. SLTL is also shown to be PSPACE-complete.

2 Sequence-Indexed Linear-Time Temporal Logic

2.1 Semantics

In this subsection, firstly, we present a semantical definition of LTL, and secondly, we introduce SLTL by extending LTL with a sequence modal operator.

Formulas of LTL are constructed from countably many propositional variables, $\rightarrow$ (implication), $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), X (next), G (globally) and F (eventually). Lower-case letters $p, q, ...$ are used to denote propositional variables, and Greek lower-case letters $\alpha, \beta, ...$ are used to denote

formulas. We write $A \equiv B$ to indicate the syntactical identity between A and B. The symbol ω is used to represent the set of natural numbers. Lower-case letters i, j and k are used to denote any natural numbers. The symbol $\geq$ or $\leq$ is used to represent a linear order on ω.

Definition 1 (LTL). *Let S be a non-empty set of states. A structure $M := (\sigma, I)$ is a* model *if*

1. *σ is an infinite sequence $s_0, s_1, s_2, ...$ of states in S,*
2. *I is a mapping from the set Φ of propositional variables to the power set of S.*

A satisfaction relation $(M, i) \models \alpha$ for any formula α, where M is a model (σ, I) and i ($\in \omega$) represents some position within σ, is defined inductively by

1. *for any $p \in \Phi$, $(M, i) \models p$ iff $s_i \in I(p)$,*
2. *$(M, i) \models \alpha \wedge \beta$ iff $(M, i) \models \alpha$ and $(M, i) \models \beta$,*
3. *$(M, i) \models \alpha \vee \beta$ iff $(M, i) \models \alpha$ or $(M, i) \models \beta$,*
4. *$(M, i) \models \alpha \rightarrow \beta$ iff $(M, i) \models \alpha$ implies $(M, i) \models \beta$,*
5. *$(M, i) \models \neg\alpha$ iff not-$[(M, i) \models \alpha]$,*
6. *$(M, i) \models \mathrm{X}\alpha$ iff $(M, i + 1) \models \alpha$,*
7. *$(M, i) \models \mathrm{G}\alpha$ iff $\forall j \geq i[(M, j) \models \alpha]$,*
8. *$(M, i) \models \mathrm{F}\alpha$ iff $\exists j \geq i[(M, j) \models \alpha]$.*

A formula α is valid *in* LTL *if $(M, 0) \models \alpha$ for any model $M := (\sigma, I)$.*

Formulas of SLTL are constructed from countably many propositional variables, $\rightarrow$, $\wedge$, $\vee$, $\neg$, X, G, F and $[b]$ (sequence modal operator) where b is a sequence. *Sequences* are constructed from countable atomic sequences, $\emptyset$ (empty sequence) and ; (composition). Lower-case letters $b, c, ...$ are used for sequences. An expression $[\emptyset]\alpha$ means α, and expressions $[\emptyset\ ;\ b]\alpha$ and $[b\ ;\ \emptyset]\alpha$ mean $[b]\alpha$.

Definition 2. *Formulas and sequences are defined by the following grammar, assuming p and e represent propositional variables and atomic sequences, respectively:*

$$\alpha ::= p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \neg\alpha \mid \mathrm{X}\alpha \mid \mathrm{G}\alpha \mid \mathrm{F}\alpha \mid [b]\alpha$$
$$b ::= e \mid \emptyset \mid b\ ;\ b$$

The set of sequences (including $\emptyset$) is denoted as SE. An expression $[\hat{d}]$ is used to represent $[d_0][d_1] \cdots [d_i]$ with $i \in \omega$ and $d_0 \equiv \emptyset$. Note that $[\hat{d}]$ can be the empty sequence. Also, an expression $\hat{d}$ is used to represent $d_0\ ;\ d_1\ ;\ \cdots\ ;\ d_i$ with $i \in \omega$.

Definition 3 (SLTL). *Let S be a non-empty set of states. A structure $M := (\sigma, \{I^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}})$ is a* sequence model *if*

1. *σ is an infinite sequence $s_0, s_1, s_2, ...$ of states in S,*
2. *$I^{\hat{d}}$ ($\hat{d} \in$ SE) are mappings from the set Φ of propositional variables to the power set of S.*

Satisfaction relations $(M,i) \models^{\hat{d}} \alpha$ *(*$\hat{d} \in \mathrm{SE}$*) for any formula* α*, where* M *is a sequence model* $(\sigma, \{I^{\hat{d}}\}_{\hat{d}\in \mathrm{SE}})$ *and* i *(*$\in \omega$*) represents some position within* σ*, is defined inductively by*

1. *for any* $p \in \Phi$, $(M,i) \models^{\hat{d}} p$ *iff* $s_i \in I^{\hat{d}}(p)$,
2. $(M,i) \models^{\hat{d}} \alpha \wedge \beta$ *iff* $(M,i) \models^{\hat{d}} \alpha$ *and* $(M,i) \models^{\hat{d}} \beta$,
3. $(M,i) \models^{\hat{d}} \alpha \vee \beta$ *iff* $(M,i) \models^{\hat{d}} \alpha$ *or* $(M,i) \models^{\hat{d}} \beta$,
4. $(M,i) \models^{\hat{d}} \alpha \rightarrow \beta$ *iff* $(M,i) \models^{\hat{d}} \alpha$ *implies* $(M,i) \models^{\hat{d}} \beta$,
5. $(M,i) \models^{\hat{d}} \neg\alpha$ *iff not-*$[(M,i) \models^{\hat{d}} \alpha]$,
6. $(M,i) \models^{\hat{d}} \mathrm{X}\alpha$ *iff* $(M,i+1) \models^{\hat{d}} \alpha$,
7. $(M,i) \models^{\hat{d}} \mathrm{G}\alpha$ *iff* $\forall j \geq i[(M,j) \models^{\hat{d}} \alpha]$,
8. $(M,i) \models^{\hat{d}} \mathrm{F}\alpha$ *iff* $\exists j \geq i[(M,j) \models^{\hat{d}} \alpha]$.
9. *for any atomic sequence* e, $(M,i) \models^{\hat{d}} [e]\alpha$ *iff* $(M,i) \models^{\hat{d}\,;\,e} \alpha$,
10. $(M,i) \models^{\hat{d}} [b\,;\,c]\alpha$ *iff* $(M,i) \models^{\hat{d}} [b][c]\alpha$.

A formula α *is* valid *in* SLTL *if* $(M,0) \models^{\emptyset} \alpha$ *for any sequence model* $M := (\sigma, \{I^{\hat{d}}\}_{\hat{d}\in \mathrm{SE}})$.

Remark that $\models^{\emptyset}$ of SLTL includes $\models$ of LTL, and hence SLTL is an extension of LTL.

Proposition 4. *The following clauses hold for any formula* α *and any sequences* c *and* $\hat{d}$,

1. $(M,i) \models^{\hat{d}} [c]\alpha$ *iff* $(M,i) \models^{\hat{d}\,;\,c} \alpha$,
2. $(M,i) \models^{\emptyset} [\hat{d}]\alpha$ *iff* $(M,i) \models^{\hat{d}} \alpha$.

Proof. Since (2) is derived from (1), we show only (1) below. (1) is proved by induction on c.

Case ($c \equiv \emptyset$): Obvious.

Case ($c \equiv e$ for an atomic sequence e): By the definition of $\models^{\hat{d}}$.

Case ($c \equiv b_1\,;\,b_2$): $(M,i) \models^{\hat{d}} [b_1\,;\,b_2]\alpha$ iff $(M,i) \models^{\hat{d}} [b_1][b_2]\alpha$ iff $(M,i) \models^{\hat{d}\,;\,b_1} [b_2]\alpha$ (by induction hypothesis) iff $(M,i) \models^{\hat{d}\,;\,b_1\,;\,b_2} \alpha$ (by induction hypothesis). **Q.E.D.**

2.2 Sequent Calculus

In this subsection, firstly, we present a sequent calculus for LTL, and secondly, we introduce a sequent calculus for SLTL.

Greek capital letters $\Gamma, \Delta, ...$ are used to represent finite (possibly empty) sets of formulas. An expression $\mathrm{X}^i\alpha$ for any $i \in \omega$ is defined inductively by $\mathrm{X}^0\alpha \equiv \alpha$ and $\mathrm{X}^{n+1}\alpha \equiv \mathrm{X}^n\mathrm{X}\alpha$. An expression of the form $\Gamma \Rightarrow \Delta$ is called a *sequent*. An expression $L \vdash S$ is used to denote the fact that a sequent S is provable in a sequent calculus L. A rule R of inference is said to be *admissible* in a sequent calculus L if the following condition is satisfied: for any instance

$$\frac{S_1 \cdots S_n}{S}$$

of R, if $L \vdash S_i$ for all i, then $L \vdash S$.

Kawai's sequent calculus LT_ω [3] for LTL is presented below.

Definition 5 ($\mathbf{LT}_\omega$). *The initial sequents of* LT_ω *are of the form: for any propositional variable* p,

$$\mathrm{X}^i p \Rightarrow \mathrm{X}^i p.$$

The structural rules of LT_ω *are of the form:*

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\text{cut})$$

$$\frac{\Gamma \Rightarrow \Delta}{\alpha, \Gamma \Rightarrow \Delta}\ (\text{we-left}) \quad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \alpha}\ (\text{we-right}).$$

The logical inference rules of LT_ω *are of the form:*

$$\frac{\Gamma \Rightarrow \Sigma, \mathrm{X}^i\alpha \quad \mathrm{X}^i\beta, \Delta \Rightarrow \Pi}{\mathrm{X}^i(\alpha \to \beta), \Gamma, \Delta \Rightarrow \Sigma, \Pi}\ (\to\text{left}) \quad \frac{\mathrm{X}^i\alpha, \Gamma \Rightarrow \Delta, \mathrm{X}^i\beta}{\Gamma \Rightarrow \Delta, \mathrm{X}^i(\alpha \to \beta)}\ (\to\text{right})$$

$$\frac{\mathrm{X}^i\alpha, \Gamma \Rightarrow \Delta}{\mathrm{X}^i(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\wedge\text{left1}) \quad \frac{\mathrm{X}^i\beta, \Gamma \Rightarrow \Delta}{\mathrm{X}^i(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\wedge\text{left2})$$

$$\frac{\Gamma \Rightarrow \Delta, \mathrm{X}^i\alpha \quad \Gamma \Rightarrow \Delta, \mathrm{X}^i\beta}{\Gamma \Rightarrow \Delta, \mathrm{X}^i(\alpha \wedge \beta)}\ (\wedge\text{right}) \quad \frac{\mathrm{X}^i\alpha, \Gamma \Rightarrow \Delta \quad \mathrm{X}^i\beta, \Gamma \Rightarrow \Delta}{\mathrm{X}^i(\alpha \vee \beta), \Gamma \Rightarrow \Delta}\ (\vee\text{left})$$

$$\frac{\Gamma \Rightarrow \Delta, \mathrm{X}^i\alpha}{\Gamma \Rightarrow \Delta, \mathrm{X}^i(\alpha \vee \beta)}\ (\vee\text{right1}) \quad \frac{\Gamma \Rightarrow \Delta, \mathrm{X}^i\beta}{\Gamma \Rightarrow \Delta, \mathrm{X}^i(\alpha \vee \beta)}\ (\vee\text{right2})$$

$$\frac{\Gamma \Rightarrow \Delta, \mathrm{X}^i\alpha}{\mathrm{X}^i\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\text{left}) \quad \frac{\mathrm{X}^i\alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \mathrm{X}^i\neg\alpha}\ (\neg\text{right})$$

$$\frac{\mathrm{X}^{i+k}\alpha, \Gamma \Rightarrow \Delta}{\mathrm{X}^i\mathrm{G}\alpha, \Gamma \Rightarrow \Delta}\ (\text{Gleft}) \quad \frac{\{\ \Gamma \Rightarrow \Delta, \mathrm{X}^{i+j}\alpha\ \}_{j\in\omega}}{\Gamma \Rightarrow \Delta, \mathrm{X}^i\mathrm{G}\alpha}\ (\text{Gright})$$

$$\frac{\{\ \mathrm{X}^{i+j}\alpha, \Gamma \Rightarrow \Delta\ \}_{j\in\omega}}{\mathrm{X}^i\mathrm{F}\alpha, \Gamma \Rightarrow \Delta}\ (\text{Fleft}) \quad \frac{\Gamma \Rightarrow \Delta, \mathrm{X}^{i+k}\alpha}{\Gamma \Rightarrow \Delta, \mathrm{X}^i\mathrm{F}\alpha}\ (\text{Fright}).$$

Remark that (Gright) and (Fleft) have infinite premises. The sequents of the form: $\mathrm{X}^i\alpha \Rightarrow \mathrm{X}^i\alpha$ for any formula α are provable in cut-free LT_ω. This fact can be proved by induction on the complexity of α. The cut-elimination and completeness theorems for LT_ω were proved by Kawai [3].

Prior to introduce a sequent calculus for SLTL, we have to introduce some notations. The symbol K is used to represent the set $\{\mathrm{X}\} \cup \{[b] \mid b \in \mathrm{SE}\}$, and the symbol K^* is used to represent the set of all words of finite length of the alphabet K. For example, $\mathrm{X}^i[\hat{b}]\mathrm{X}^j[\hat{c}]$ is in K^*. Remark that K^* includes $\emptyset$, and

hence $\{\dagger\alpha \mid \dagger \in K^*\}$ includes α. An expression $\sharp$ is used to represent an arbitrary member of K^*.

A sequent calculus SLT_ω for SLTL is then introduced below.

Definition 6 ($\mathbf{SLT}_\omega$). *The initial sequents of* SLT_ω *are of the form: for any propositional variable* p*,*

$$\sharp p \Rightarrow \sharp p.$$

The structural rules of SLT_ω *are* (cut)*,* (we-left) *and* (we-right) *in Definition 5.*

The logical inference rules of SLT_ω *are of the form:*

$$\frac{\Gamma \Rightarrow \Sigma, \sharp\alpha \quad \sharp\beta, \Delta \Rightarrow \Pi}{\sharp(\alpha{\to}\beta), \Gamma, \Delta \Rightarrow \Sigma, \Pi}\ (\to\mathrm{left}^s) \qquad \frac{\sharp\alpha, \Gamma \Rightarrow \Delta, \sharp\beta}{\Gamma \Rightarrow \Delta, \sharp(\alpha{\to}\beta)}\ (\to\mathrm{right}^s)$$

$$\frac{\sharp\alpha, \Gamma \Rightarrow \Delta}{\sharp(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\wedge\mathrm{left1}^s) \qquad \frac{\sharp\beta, \Gamma \Rightarrow \Delta}{\sharp(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\wedge\mathrm{left2}^s)$$

$$\frac{\Gamma \Rightarrow \Delta, \sharp\alpha \quad \Gamma \Rightarrow \Delta, \sharp\beta}{\Gamma \Rightarrow \Delta, \sharp(\alpha \wedge \beta)}\ (\wedge\mathrm{right}^s) \qquad \frac{\sharp\alpha, \Gamma \Rightarrow \Delta \quad \sharp\beta, \Gamma \Rightarrow \Delta}{\sharp(\alpha \vee \beta), \Gamma \Rightarrow \Delta}\ (\vee\mathrm{left}^s)$$

$$\frac{\Gamma \Rightarrow \Delta, \sharp\alpha}{\Gamma \Rightarrow \Delta, \sharp(\alpha \vee \beta)}\ (\vee\mathrm{right1}^s) \qquad \frac{\Gamma \Rightarrow \Delta, \sharp\beta}{\Gamma \Rightarrow \Delta, \sharp(\alpha \vee \beta)}\ (\vee\mathrm{right2}^s)$$

$$\frac{\Gamma \Rightarrow \Delta, \sharp\alpha}{\sharp\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\mathrm{left}^s) \qquad \frac{\sharp\alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \sharp\neg\alpha}\ (\neg\mathrm{right}^s)$$

$$\frac{\sharp\mathrm{X}^k\alpha, \Gamma \Rightarrow \Delta}{\sharp\mathrm{G}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Gleft}^s) \qquad \frac{\{\ \Gamma \Rightarrow \Delta, \sharp\mathrm{X}^j\alpha\ \}_{j\in\omega}}{\Gamma \Rightarrow \Delta, \sharp\mathrm{G}\alpha}\ (\mathrm{Gright}^s)$$

$$\frac{\{\ \sharp\mathrm{X}^j\alpha, \Gamma \Rightarrow \Delta\ \}_{j\in\omega}}{\sharp\mathrm{F}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Fleft}^s) \qquad \frac{\Gamma \Rightarrow \Delta, \sharp\mathrm{X}^k\alpha}{\Gamma \Rightarrow \Delta, \sharp\mathrm{F}\alpha}\ (\mathrm{Fright}^s)$$

$$\frac{\sharp[b]\mathrm{X}\alpha, \Gamma \Rightarrow \Delta}{\sharp\mathrm{X}[b]\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Xleft}) \qquad \frac{\Gamma \Rightarrow \Delta, \sharp[b]\mathrm{X}\alpha}{\Gamma \Rightarrow \Delta, \sharp\mathrm{X}[b]\alpha}\ (\mathrm{Xright}).$$

The sequence inference rules of SLT_ω *are of the form:*

$$\frac{\sharp[b][c]\alpha, \Gamma \Rightarrow \Delta}{\sharp[b\ ;\ c]\alpha, \Gamma \Rightarrow \Delta}\ (;\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \Delta, \sharp[b][c]\alpha}{\Gamma \Rightarrow \Delta, \sharp[b\ ;\ c]\alpha}\ (;\mathrm{right}).$$

The sequents of the form $\sharp\alpha \Rightarrow \sharp\alpha$ for any formula α are provable in cut-free SLT_ω. This fact can be proved by induction on the complexity of α.

Proposition 7. *The rules of the form:*

$$\frac{\Gamma \Rightarrow \Delta}{[b]\Gamma \Rightarrow [b]\Delta}\ (\mathrm{regu}) \qquad \frac{\sharp\mathrm{X}[b]\alpha, \Gamma \Rightarrow \Delta}{\sharp[b]\mathrm{X}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Xleft}^{-1}) \qquad \frac{\Gamma \Rightarrow \Delta, \sharp\mathrm{X}[b]\alpha}{\Gamma \Rightarrow \Delta, \sharp[b]\mathrm{X}\alpha}\ (\mathrm{Xright}^{-1})$$

are admissible in cut-free SLT_ω*.*

3 Cut-Elimination and Completeness

Firstly, we introduce a translation of SLTL into LTL, and by using this translation, we show the semantical and syntactical embedding theorems of SLTL into LTL. As corollaries of the embedding theorems, we obtain the decidability, cut-elimination and completeness theorems for SLTL and SLT_ω.

Definition 8. *Let Φ be a non-empty set of propositional variables and $\Phi^{\hat{d}}$ be the set $\{p^{\hat{d}} \mid p \in \Phi\}$ ($\hat{d} \in \mathrm{SE}$) of propositional variables where $p^{\emptyset} := p$ (i.e., $\Phi^{\emptyset} := \Phi$). The language $\mathcal{L}^s$ (the set of formulas) of* SLTL *is defined using Φ, $[b]$, $\wedge, \vee, \rightarrow, \neg$,* X, F *and* G *by the same way as in Definition 2. The language $\mathcal{L}$ of* LTL *is obtained from $\mathcal{L}^s$ by adding $\Phi^{\hat{d}}$ and deleting $[b]$.*

A mapping f from $\mathcal{L}^s$ to $\mathcal{L}$ is defined by

1. *for any $p \in \Phi$, $f([\hat{d}]p) := p^{\hat{d}} \in \Phi^{\hat{d}}$, esp., $f(p) = p \in \Phi^{\emptyset}$,*[1]
2. *$f(\sharp(\alpha \circ \beta)) := f(\sharp\alpha) \circ f(\sharp\beta)$ where $\circ \in \{\wedge, \vee, \rightarrow\}$,*
3. *$f(\sharp\dagger\alpha) := \dagger f(\sharp\alpha)$ where $\dagger \in \{\neg, \mathrm{X}, \mathrm{G}, \mathrm{F}\}$,*
4. *$f(\sharp[b\ ;\ c]\alpha) := f(\sharp[b][c]\alpha)$.*

Remark that we can derive the following clause for f:

5. $f(\sharp[b]\mathrm{X}\alpha) := f(\sharp\mathrm{X}[b]\alpha)$.

Lemma 9. *Let f be the mapping defined in Definition 8, and S be a non-empty set of states. For any sequence model $M := (\sigma, \{I^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}})$ of* SLTL*, any satisfaction relations $\models^{\hat{d}}$ ($\hat{d} \in \mathrm{SE}$) on M, and any state s_i in σ, we can construct a model $N := (\sigma, I)$ of* LTL *and a satisfaction relation $\models$ on N such that for any formula α in $\mathcal{L}^s$, $(M, i) \models^{\hat{d}} \alpha$ iff $(N, i) \models f([\hat{d}]\alpha)$.*

Proof. Let Φ be a non-empty set of propositional variables and $\Phi^{\hat{d}}$ be the set $\{p^{\hat{d}} \mid p \in \Phi\}$. Suppose that M is a sequence model $(\sigma, \{I^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}})$ where

$I^{\hat{d}}$ ($\hat{d} \in \mathrm{SE}$) are mappings from Φ to the power set of S.

Suppose that N is a model (σ, I) where

I is a mapping from $\bigcup_{\hat{d} \in \mathrm{SE}} \Phi^{\hat{d}}$ to the power set of S.

Suppose moreover that M and N satisfy the following condition: for any s_i in σ and any $p \in \Phi$,

$s_i \in I^{\hat{d}}(p)$ iff $s_i \in I(p^{\hat{d}})$.

[1] For example, we have: $f([e_1][e_2]p) = p^{e_1\ ;\ e_2} = f([e_1\ ;\ e_2]p)$ for any propositional variable p and any atomic sequences e_1 and e_2.

Then, the lemma is proved by induction on the complexity of α.

• Base step:

Case $\alpha \equiv p \in \Phi$: We obtain: $(M,i) \models^{\hat{d}} p$ iff $s_i \in I^{\hat{d}}(p)$ iff $s_i \in I(p^{\hat{d}})$ iff $(N,i) \models p^{\hat{d}}$ iff $(N,i) \models f([\hat{d}]p)$ (by the definition of f).

• Induction step:

Case $\alpha \equiv \beta \wedge \gamma$: We obtain: $(M,i) \models^{\hat{d}} \beta \wedge \gamma$ iff $(M,i) \models^{\hat{d}} \beta$ and $(M,i) \models^{\hat{d}} \gamma$ iff $(N,i) \models f([\hat{d}]\beta)$ and $(N,i) \models f([\hat{d}]\gamma)$ (by induction hypothesis) iff $(N,i) \models f([\hat{d}]\beta) \wedge f([\hat{d}]\gamma)$ iff $(N,i) \models f([\hat{d}](\beta \wedge \gamma))$ (by the definition of f).

Case $\alpha \equiv \beta \vee \gamma$: Similar to Case $\alpha \equiv \beta \wedge \gamma$.

Case $\alpha \equiv \beta \rightarrow \gamma$: We obtain: $(M,i) \models^{\hat{d}} \beta \rightarrow \gamma$ iff $(M,i) \models^{\hat{d}} \beta$ implies $(M,i) \models^{\hat{d}} \gamma$ iff $(N,i) \models f([\hat{d}]\beta)$ implies $(N,i) \models f([\hat{d}]\gamma)$ (by induction hypothesis) iff $(N,i) \models f([\hat{d}]\beta) \rightarrow f([\hat{d}]\gamma)$ iff $(N,i) \models f([\hat{d}](\beta \rightarrow \gamma))$ (by the definition of f).

Case $\alpha \equiv \neg\beta$: We obtain: $(M,i) \models^{\hat{d}} \neg\beta$ iff not-$[(M,i) \models^{\hat{d}} \beta]$ iff not-$[(N,i) \models f([\hat{d}]\beta)]$ (by induction hypothesis) iff $(N,i) \models \neg f([\hat{d}]\beta)$ iff $(N,i) \models f([\hat{d}]\neg\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{X}\beta$: We obtain: $(M,i) \models^{\hat{d}} \mathrm{X}\beta$ iff $(M,i+1) \models^{\hat{d}} \beta$ iff $(N,i+1) \models f([\hat{d}]\beta)$ (by induction hypothesis) iff $(N,i) \models \mathrm{X}f([\hat{d}]\beta)$ iff $(N,i) \models f([\hat{d}]\mathrm{X}\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{G}\beta$: We obtain: $(M,i) \models^{\hat{d}} \mathrm{G}\beta$ iff $\forall j \geq i[(M,j) \models^{\hat{d}} \beta]$ iff $\forall j \geq i[(N,j) \models f([\hat{d}]\beta)]$ (by induction hypothesis) iff $(N,i) \models \mathrm{G}f([\hat{d}]\beta)$ iff $(N,i) \models f([\hat{d}]\mathrm{G}\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{F}\beta$: Similar to Case $\alpha \equiv \mathrm{G}\beta$.

Case $(\alpha \equiv [b]\beta)$: $(M,i) \models^{\hat{d}} [b]\beta$ iff $(M,i) \models^{\hat{d}\,;\,b} \beta$ (by Proposition 4) iff $(N,i) \models f([\hat{d}\,;\,b]\beta)$ (by induction hypothesis) iff $(N,i) \models f([\hat{d}][b]\beta)$ by the definition of f. **Q.E.D.**

Lemma 10. *Let f be the mapping defined in Definition 8, and S be a non-empty set of states. For any model $N := (\sigma, I)$ of* LTL*, any satisfaction relation $\models$ on N, and any state s_i in σ, we can construct a sequence model $M := (\sigma, \{I^{\hat{d}}\}_{\hat{d} \in \mathrm{SE}})$ of* SLTL *and satisfaction relations $\models^{\hat{d}}$ ($\hat{d} \in$ SE) on M such that for any formula α in $\mathcal{L}^s$, $(N,i) \models f([\hat{d}]\alpha)$ iff $(M,i) \models^{\hat{d}} \alpha$.*

Proof. Similar to the proof of Lemma 9. **Q.E.D.**

Theorem 11 (Semantical embedding). *Let f be the mapping defined in Definition 8. For any formula α, α is valid in* SLTL *iff $f(\alpha)$ is valid in* LTL.

Proof. By Lemmas 9 and 10. **Q.E.D.**

An expression $f(\Gamma)$ denotes the result of replacing every occurrence of a formula α in Γ by an occurrence of $f(\alpha)$.

Theorem 12 (Syntactical embedding). *Let Γ and Δ be sets of formulas in $\mathcal{L}^s$, and f be the mapping defined in Definition 8. Then:*

1. $\mathrm{SLT}_\omega \vdash \Gamma \Rightarrow \Delta$ *iff* $\mathrm{LT}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta)$.

2. $\mathrm{SLT}_\omega - (\mathrm{cut}) \vdash \Gamma \Rightarrow \Delta$ *iff* $\mathrm{LT}_\omega - (\mathrm{cut}) \vdash f(\Gamma) \Rightarrow f(\Delta)$.

Proof. Since the case (2) can be obtained as the subproof of the case (1), we show only (1) in the following. In the following proof, we assume that the total number of X in $\sharp$ is i, and that the sequence which is obtained from $\sharp$ by deleting all X is $[\hat{d}]$.

• ($\Longrightarrow$) : By induction on the proofs P of $\Gamma \Rightarrow \Delta$ in SLT_ω. We distinguish the cases according to the last inference of P, and show some cases.

Case ($\sharp p \Rightarrow \sharp p$): The last inference of P is of the form: $\sharp p \Rightarrow \sharp p$. In this case, we obtain the required fact $\mathrm{LT}_\omega \vdash f(\sharp p) \Rightarrow f(\sharp p)$ since $f(\sharp p)$ coincides with $\mathrm{X}^i p^{\hat{d}}$ by the definition of f.

Case (Fleft^s): The last inference of P is of the form:

$$\frac{\{\ \sharp \mathrm{X}^j \alpha, \Gamma \Rightarrow \Delta\ \}_{j\in\omega}}{\sharp \mathrm{F}\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Fleft}^s).$$

By induction hypothesis, we have: $\mathrm{LT}_\omega \vdash f(\sharp \mathrm{X}^j \alpha), f(\Gamma) \Rightarrow f(\Delta)$ for any $j \in \omega$, where $f(\sharp \mathrm{X}^j \alpha)$ coincides with $\mathrm{X}^{i+j} f([\hat{d}]\alpha)$ by the definition of f. Then, we obtain:

$$\frac{\begin{array}{c}\vdots\\ \{\ \mathrm{X}^{i+j} f([\hat{d}]\alpha), f(\Gamma) \Rightarrow f(\Delta)\ \}_{j\in\omega}\end{array}}{\mathrm{X}^i \mathrm{F} f([\hat{d}]\alpha), f(\Gamma) \Rightarrow f(\Delta)}\ (\mathrm{Fleft})$$

where $\mathrm{X}^i \mathrm{F} f([\hat{d}]\alpha)$ coincides with $f(\sharp \mathrm{F}\alpha)$ by the definition of f.

Case (;left): The last inference of P is of the form:

$$\frac{\sharp [b][c]\alpha, \Gamma \Rightarrow \Delta}{\sharp [b\ ;\ c]\alpha, \Gamma \Rightarrow \Delta}\ (;\mathrm{left}).$$

By induction hypothesis, we obtain: $\mathrm{LT}_\omega \vdash f(\sharp [b][c]\alpha), f(\Gamma) \Rightarrow f(\Delta)$ where $f(\sharp [b][c]\alpha)$ coincides with $f(\sharp [b\ ;\ c]\alpha)$ by the definition of f.

Case (Xleft): The last inference of P is of the form:

$$\frac{\sharp [b]\mathrm{X}\alpha, \Gamma \Rightarrow \Delta}{\sharp \mathrm{X}[b]\alpha, \Gamma \Rightarrow \Delta}\ (\mathrm{Xleft}).$$

By induction hypothesis, we obtain: $\mathrm{LT}_\omega \vdash f(\sharp [b]\mathrm{X}\alpha), f(\Gamma) \Rightarrow f(\Delta)$ where $f(\sharp [b]\mathrm{X}\alpha)$ coincides with $f(\sharp \mathrm{X}[b]\alpha)$ by the definition of f.

• ($\Longleftarrow$) : By induction on the proofs Q of $f(\Gamma) \Rightarrow f(\Delta)$ in LT_ω. We distinguish the cases according to the last inference of Q, and show only the following case.

Case (cut): The last inference of Q is of the form:

$$\frac{f(\Gamma_1) \Rightarrow f(\Delta_1), \beta \quad \beta, f(\Gamma_2) \Rightarrow f(\Delta_2)}{f(\Gamma_1), f(\Gamma_2) \Rightarrow f(\Delta_1), f(\Delta_2)}\ (\mathit{cut}).$$

Since β is in $\mathcal{L}$, we have the fact $\beta = f(\beta)$. This fact can be shown by induction on β. Then, by induction hypothesis, we have: $\mathrm{SLT}_\omega \vdash \Gamma_1 \Rightarrow \Delta_1, \beta$ and $\mathrm{SLT}_\omega \vdash \beta, \Gamma_2 \Rightarrow \Delta_2$. We then obtain the required fact: $\mathrm{SLT}_\omega \vdash \Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$ by using (cut) in SLT_ω. **Q.E.D.**

Theorem 13 (Cut-elimination). *The rule* (cut) *is admissible in cut-free* SLT_ω.

Proof. Suppose $\mathrm{SLT}_\omega \vdash \Gamma \Rightarrow \Delta$. Then, we have $\mathrm{LT}_\omega \vdash f(\Gamma) \Rightarrow f(\Delta)$ by Theorem 12 (1), and hence $\mathrm{LT}_\omega - (\mathrm{cut}) \vdash f(\Gamma) \Rightarrow f(\Delta)$ by the cut-elimination theorem for LT_ω. By Theorem 12 (2), we obtain $\mathrm{SLT}_\omega - (\mathrm{cut}) \vdash \Gamma \Rightarrow \Delta$. **Q.E.D.**

Theorem 14 (Completeness). *For any formula* α, $\mathrm{SLT}_\omega \vdash \Rightarrow \alpha$ *iff* α *is valid in* SLTL.

Proof. $\mathrm{SLT}_\omega \vdash \Rightarrow \alpha$ iff $\mathrm{LT}_\omega \vdash \Rightarrow f(\alpha)$ (by Theorem 12) iff $f(\alpha)$ is valid in LTL (by the completeness theorem for LTL) iff α is valid in SLTL (by Theorem 11). **Q.E.D.**

Theorem 15 (Complexity). SLTL *is PSPACE-complete.*

Proof. (Propositional) LTL (without until operator) is known to be PSPACE-complete [5]. By decidability of LTL, for each α, it is possible to decide if $f(\alpha)$ is valid in LTL. Then, by Theorem 11, SLTL is decidable. Since f is a polynomial-time reduction, SLTL is also PSPACE-complete. **Q.E.D.**

Acknowledgments. This research was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) No. 20700015.

References

1. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Formal Models and Semantics (B), pp. 995–1072. Elsevier, MIT Press (1990)
2. Kamide, N., Kaneiwa, K.: Extended full computation-tree logic with sequence modal operator: representing hierarchical tree structures. In: Nicholson, A., Li, X. (eds.) AI 2009. LNCS (LNAI), vol. 5866, pp. 485–494. Springer, Heidelberg (2009)
3. Kawai, H.: Sequential calculus for a first order infinitary temporal logic. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 33, 423–432 (1987)
4. A. Pnueli, The temporal logic of programs. In: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57 (1977)
5. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the ACM 32(3), 733–749 (1985)
6. Wansing, H.: The Logic of Information Structures. LNCS, vol. 681, 163 pages. Springer, Heidelberg (1993)

A Refuted Conjecture on Probabilistic Satisfiability*

Marcelo Finger** and Glauber De Bona***

Institute of Mathematics and Statistics, University of São Paulo,
São Paulo, Brazil
{mfinger,debona}@ime.usp.br

Abstract. In this paper, we investigate the Probabilistic Satisfiability Problem, and its relation with the classical Satisfiability Problem, looking for a possible polynomial-time reduction. For this, we present an Atomic Normal Form to the probabilistic satisfiability problem and then we define a Probabilistic Entailment relation, showing its inherent properties. At the end, we enunciate and refute a conjecture that could lead to the desired polynomial-time reduction.

Keywords: probabilistic logic, probabilistic satisfiability.

1 Introduction

The study of reasoning under uncertainty is a subject from many fields, and, in computer science, it has been useful on distributed system analysis and program analysis under probabilistic assumptions. In the XIX century, Boole [1] has already studied the probability assignment to logical sentences, and we see his influence on de Finetti's theory of subject probability [3]. In 1965, Hailperin [5] revisited the problem, giving it a linear programming form. In 1986, Nilsson [8] formalized the probabilistic satisfiability problem as we know it today: given logical sentences, and a probability assignment to them, we want to know if this assignment is consistent. The main analytical and numerical solutions to this problem, as well its detailed history, can be seen in [6].

The probabilistic satisfiability problem (PSAT) is NP-complete. Thus the Cook-Levin Theorem [2] tells us that there is a polynomial reduction from PSAT to classical satisfiability (SAT). Such reduction might be interesting to exploit the efficiency of SAT solvers. Another reason to study reduction from PSAT to SAT is the seek for a better understanding on the relation between logic and probability. The objective of this work is to investigate the relation between PSAT and SAT, looking for paths that enable the desired reduction.

In Section 2 we formally present the PSAT problem, in Nilsson's linear programming formulation [8]. In Section 3 we introduce the Atomic Normal Form

* This work was supported by Fapesp Project 2008/03995-5 (LogProb).
** Partly supported by CNPq/Brazil, grant PQ 304607/2007-0.
*** Grant recipient from CNPq - Brazil.

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 293–302, 2010.

for PSAT and in Section 4 a probabilistic entailment relation is suggested. Then in Section 5 we present a conjecture, under the concepts developed in the previous sections, on a possible reduction from PSAT to SAT. Such conjecture is exhaustively refuted by a counterexample.

2 The Problem

The probabilistic satisfiability (PSAT) is a decision problem, where we explore the consistency of a probability assignment over logical formulas.

Formally, let $S = \{s_1, \ldots, s_k\}$ be a set with k logical sentences, defined on a set of n Boolean variables, $X = \{x_1, \ldots, x_n\}$, with the usual operators from the classical propositional logic. A truth assignment (or valuation) v is initially defined as a function that associates truth values to Boolean variables, formally $v : X \to \{0, 1\}$. Then we can extend its domain to the set of formulas S, as usual in classical logic[1], $v : S \to \{0, 1\}$.

Let $V = \{v_1, \ldots, v_{2^n}\}$ be the set of possible truth assignments over X. A probability distribution over propositional valuations $\pi : V \to [0, 1]$ is a function that maps every valuation to a value in the real interval $[0, 1]$ such that $\sum_{i=1}^{2^n} \pi(v_i) = 1$. The probability of a formula s according to π is giving by $p_\pi(s) = \sum\{\pi(v_j) | v_j(s) = 1\}$.

Let $P = \{p_i | 0 \leq p_i \leq 1,\ 1 \leq i \leq k\}$ be a set of probabilities. We say that the probability assignment $p(s_i) = p_i$ is consistent if, and only if, there is a distribution π over V that makes $p_\pi(s_i) = p_i, 1 \leq i \leq k$. Finally a PSAT instance, defined by the set S and by $p(s_i) = p_i$, with $1 \leq i \leq k$, is satisfiable iff this probability assignment is consistent.

Now PSAT can be expressed as a linear programming problem, as introduced in [8]. Let Δ be the PSAT instance made by assigning the probabilities in P to the k formulas in S, $\Delta = \{p(s_i) = p_i | 1 \leq i \leq k\}$. We define the matrix $A_{k \times 2^n} = [a_{ij}]$, such that $a_{ij} = v_j(s_i)$, and the matrix $p_{k \times 1} = [p_i]$. So Δ is satisfiable iff there is a vector π that hold the restrictions:

$$A\pi = p \ ; \tag{1}$$

$$\pi \geq 0 \ ; \tag{2}$$

$$\sum \pi = 1 \ . \tag{3}$$

If there is a feasible solution π, then we say that π satisfies Δ, else we say that Δ is unsatisfiable. The restrictions (2) and (3) force π to be a probability distribution. The restriction (3) can be omitted if we add an entire row of $1's$ to the matrix A, $a_{k+1,j} = 1, 1 \leq j \leq 2^n$, and if we add an element $p_{k+1,1} = 1$ to the vector p, what will be done from here until the end of this paper.

According to Carathéodory's Lemma [9], if the linear programming problem (1-3) has feasible solution, then there is a solution with only $k + 1$ elements of

[1] Let α and β be formulas from classical propositional logic, we have: $v(\alpha \wedge \beta) = 1$ iff $v(\alpha) = 1$ and $v(\beta) = 1$; $v(\alpha \vee \beta) = 1$ iff $v(\alpha) = 1$ or $v(\beta) = 1$; $v(\neg\alpha) = 1$ iff $v(\alpha) = 0$; $v(\alpha \to \beta) = 1$ iff $v(\alpha) = 0$ or $v(\beta) = 1$; and $v(\alpha \leftrightarrow \beta) = 1$ iff $v(\alpha) = v(\beta)$.

π different from zero. As pointed in [4], this result places PSAT in NP, because we can take a matrix $A_{k+1,k+1}$ and a vector $\pi_{k+1,1}$ as NP-certificate. Besides that, any instance from classical satisfiability (SAT), made from a set S with k sentences, can be reduced to a PSAT instance, in polynomial time, by making $p(s_i) = p_i = 1, 1 \leq i \leq k$. It follows that PSAT is NP-hard, hence NP-complete.

3 The Atomic Normal Form

Let $S = \{s_1, \ldots, s_k\}$ be a set of sentences from classical propositional logic, over the set $X = \{x_1, \ldots, x_n\}$ of Boolean variables. We say that a PSAT instance, $\Delta = \{p(s_i) = p_i | 1 \leq i \leq k\}$, $0 \leq p_i \leq 1$, is in the *Atomic Normal Form* if it can be partitioned in two sets, (Γ, Ψ), where $\Gamma = \{p(s_i) = 1 | 1 \leq i \leq m\}$ and $\Psi = \{p(y_i) = p_i | y_i$ is an atom and $1 \leq i \leq l\}$, with $0 \leq p_i \leq 1$, where $k = m + l$. The partition Γ is the SAT part of the atomic normal form, usually represented as a set of formulas, and the partition Ψ is the atomic probability assignment part. The following Theorem shows how any PSAT instance can be brought to the atomic normal form, by adding a linear number of new variables.

Theorem 1 (Atomic Normal Form). *Let $\Delta = \{p(s_i) = p_i | 1 \leq i \leq k\}$ be a PSAT instance. Then a PSAT instance (Γ, Ψ) in the atomic normal form can be built, in polynomial time on k, such that Δ is satisfiable iff (Γ, Ψ) is satisfiable.*

Proof. To build a PSAT instance (Γ, Ψ) in the atomic normal form, from the instance $\Delta = \{p(s_i) = p_i, 1 \leq i \leq k\}$, we first add k new variables, $y_1, \ldots, y_k$. Then we make $\Gamma = \{p(y_i \leftrightarrow s_i) = 1 | 1 \leq i \leq k\}$ and $\Psi = \{p(y_i) = p_i | 1 \leq i \leq k\}$. Clearly, this can be done in polynomial time on k.

Suppose there is a probability distribution π over the truth assignments $v : \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_k\} \rightarrow \{0, 1\}$ that satisfies (Γ, Ψ). Because π satisfies (Γ, Ψ), we have $p_\pi(y_i) = p_i$, $1 \leq i \leq k$. By the construction of Γ and the laws of probability, $p_\pi(y_i) = p_\pi(s_i)$, thus $p_\pi(s_i) = p_i$, $1 \leq i \leq k$. Over the truth assignments $v' : \{x_1, \ldots, x_n\} \rightarrow \{0, 1\}$, we define a probability distribution π':

$$\pi'(v') = \sum \{\pi(v) | v(x_i) = v'(x_i), 1 \leq i \leq n\} \ .$$

Hence π' is a probability distribution over $\{x_1, \ldots, x_n\}$ that satisfies $p_{\pi'}(s_i) = p_i$, $1 \leq i \leq k$, and consequently π' satisfies Δ.

Now suppose there is a probability distribution π' over the truth assignments $v' : \{x_1, \ldots, x_n\} \rightarrow \{0, 1\}$ that satisfies Δ. Because π' satisfies Δ, $p_{\pi'}(s_i) = p_i$, $1 \leq i \leq k$. We define a probability distribution π over the truth assignments $v : \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_k\} \rightarrow \{0, 1\}$:

$$\pi(v) = \begin{cases} \pi'(v') \text{ , if } v(x_i) = v'(x_i) \text{ and } v(y_j) = v(s_j),\ 1 \leq i \leq n \text{ and } 1 \leq j \leq k \\ 0 \text{ , other cases} \end{cases} .$$

Clearly, we have $p_\pi(s_i) = p_{\pi'}(s_i)$ and $p_\pi(y_i) = p_\pi(s_i)$, $1 \leq i \leq k$. It follows that $p_\pi(y_i) = p_{\pi'}(s_i) = p_i$, $1 \leq i \leq k$, and then π satisfies Ψ. For all v, such that $\pi(v) \neq 0$, we have $v(y_i) = v(s_i)$, $1 \leq i \leq k$, thus $p_\pi(y_i \leftrightarrow s_i) = 1$, $1 \leq i \leq k$, and π satisfies Γ. Finally, π satisfies (Γ, Ψ). □

The atomic normal form allows us to see a PSAT instance (Γ, Ψ) as an interaction between a probability assignment (Ψ) and a SAT instance (Γ). Solutions to (Γ, Ψ) can be seen as solutions to Ψ constrained by the SAT instance Γ.

4 A Probabilistic Entailment Relation

Let Ψ be an atomic probability assignment and let α be a formula. We say that Ψ probabilistically entails α, denoted by $\Psi \mathrel{|\!\approx} \alpha$, iff the PSAT instance in the atomic normal form $(\{\neg\alpha\}, \Psi)$ is (probabilistically) unsatisfiable. In other words, if $(\{\neg\alpha\}, \Psi)$ is unsatisfiable, then $p(\neg\alpha) = 1$ and $p(\alpha) = 0$ are not consistent with Ψ; and because the probabilities are non-negative, $\Psi \mathrel{|\!\approx} \alpha$ implies $p(\alpha) > 0$, for any probability distribution that satisfies Ψ.

We denote by $\Psi^{\mathrel{|\!\approx}}$ the set of all formulas α such that $\Psi \mathrel{|\!\approx} \alpha$. The following Theorem shows, by this set, the role probabilistic entailment plays in PSAT study.

Theorem 2. *Let $\Sigma = (\Gamma, \Psi)$ be a PSAT instance in the atomic normal form. Σ is satisfiable iff for each $\alpha \in \Psi^{\mathrel{|\!\approx}}$, $\Gamma \cup \{\alpha\}$ is classically satisfiable.*

Proof. Suppose there is an $\alpha \in \Psi^{\mathrel{|\!\approx}}$, but $\Gamma \cup \{\alpha\}$ is classically unsatisfiable, then $\Gamma \vDash \neg\alpha$. From $\alpha \in \Psi^{\mathrel{|\!\approx}}$, we obtain that $(\{\neg\alpha\}, \Psi)$ is probabilistically unsatisfiable, and thus (Γ, Ψ) is also probabilistically unsatisfiable.

Conversely, suppose (Γ, Ψ) is probabilistically unsatisfiable. Let γ be the conjunction of all formulas in Γ. Obviously $(\{\gamma\}, \Psi)$ is also unsatisfiable. It follows that $\neg\gamma \in \Psi^{\mathrel{|\!\approx}}$ and, clearly, $\Gamma \cup \{\neg\gamma\}$ is classically unsatisfiable. □

This motivates the study of the probabilistic entailment properties.

4.1 Probabilistic Entailment Properties

We first note an initial relation between $\mathrel{|\!\approx}$ and $\vDash$:

Lemma 1. *If $\Psi \mathrel{|\!\approx} \alpha$ and $\alpha \vDash \beta$, then $\Psi \mathrel{|\!\approx} \beta$.*

Proof. From $\Psi \mathrel{|\!\approx} \alpha$, we know that $p(\alpha) > 0$, and $\alpha \vDash \beta$ yields $p(\alpha) \leq p(\beta)$. So $p(\beta) > 0$, and therefore $\Psi \mathrel{|\!\approx} \beta$. □

However we note that $\Psi \mathrel{|\!\approx} \alpha$ and $\Psi \mathrel{|\!\approx} \beta$ don't imply $\Psi \mathrel{|\!\approx} \alpha \wedge \beta$. As counterexample, we take $p(\alpha) = p(\beta) = 0.4$ and $p(\alpha \vee \beta) = 0.8$, from where we obtain $p(\alpha \wedge \beta) = 0$ and thus $\Psi \not\mathrel{|\!\approx} \alpha \wedge \beta$. In this counterexample, we made $p(\alpha \vee \beta) = p(\alpha) + p(\beta)$, but it's only possible when $p(\alpha) + p(\beta) \leq 1$. This leads us to the next Lemma.

Because in the atomic normal form the probabilities are assigned to atoms, and consequently to their negations, it's useful to define literal, so we can talk about probabilities overs literals. A literal x is an atom or its negation, and $\bar{x}$ denotes the negation of x.

Lemma 2. *Let Ψ be an atomic probability assignment such that, for literals y and z, $p(y) + p(z) > 1$. Then $\Psi \not\approx y \wedge z$.*

Proof. As direct consequence of Kolmogorov's probability axioms, we know that

$$p(y) + p(z) = p(y \vee z) + p(y \wedge z) \ .$$

As $p(y) + p(z) > 1$, and always $p(y \vee z) \leq 1$, we obtain that $p(y \wedge z) > 0$, and thus $\Psi \not\approx y \wedge z$. □

However, as $p(y_1) + \ldots + p(y_k) > 1$ doesn't imply $\Psi \not\approx y_1 \wedge \ldots \wedge y_k$, we look for a suitable generalization for the Lemma 2.

Let $y_1, \ldots, y_k$ be literals and let j be an integer, with $1 \leq j \leq k$. We define:

$$C^j(y_1, \ldots, y_k) = \bigvee \{ y_{i_1} \wedge \ldots \wedge y_{i_j} | 1 \leq i_1 < \ldots < i_j \leq k \} \ . \tag{4}$$

For example, $C^1(y, z, w) = y \vee z \vee w$, $C^2(y, z, w) = (y \wedge z) \vee (y \wedge w) \vee (z \vee w)$ and $C^3(y, z, w) = y \wedge z \wedge w$. It is useful to define $C^0(y_1, \ldots, y_k) = 1$, as the conjunction neutral element. We call formulas in the format (4) a *C-formula*.

From the commutativity of the logical operators $\wedge$ and $\vee$, we obtain that the literals order in (4) is irrelevant. Besides that, let's look at the following C-formulas properties, related to the entailment:

Lemma 3. *Let $y_1, \ldots, y_k$ be literals and let j, j', k and k' be non-negative integers:*

(a) if $0 \leq j' < j$ then $C^j(y_1, \ldots, y_k) \vDash C^{j'}(y_1, \ldots, y_k)$;
(b) if $k' > k$ then $C^j(y_1, \ldots, y_k) \vDash C^j(y_1, \ldots, y_{k'})$.

Proof

(a) Let $Y = \{y_1, \ldots, y_k\}$ be a set of literals. Note that the truth assignment v satisfies $C^j(y_1, \ldots, y_k)$ iff there is a set $Y' \subseteq Y$ such that $|Y'| = j$ and $v(y) = 1$, for all $y \in Y'$. Obviously, if v satisfies $C^j(y_1, \ldots, y_k)$, then, for each $1 \leq j' < j$, there is a set $Y'' \subseteq Y' \subseteq Y$ such that $|Y''| = j'$ and $v(y) = 1$ for all $y \in Y''$. So such truth assignment v must also satisfy $C^{j'}(y_1, \ldots, y_k)$.
(b) When $k' > k$, $C^j(y_1, \ldots, y_{k'})$ can be written in the format $\alpha \vee C^j(y_1, \ldots, y_k)$. So each truth assignment that satisfies $C^j(y_1, \ldots, y_k)$ also satisfies $C^j(y_1, \ldots, y_{k'})$. □

Let $Y = \{y_1, \ldots, y_k\}$ be a set of literals. Another important C-formulas property, to be used in section 5, is related to adding opposite literals, $z, \bar{z} \notin Y$, in $C^j(Y)$, where $C^j(Y)$ denotes $C^j(y_1, \ldots, y_k)$:

Lemma 4. *Let Y be a set of literals, and let z be a literal, such that $z, \bar{z} \notin Y$. Then $C^j(Y) \equiv C^{j+1}(Y \cup \{z, \bar{z}\})$.*

Proof. Expanding the formula $C^{j+1}(Y \cup \{z, \bar{z}\})$ and ruling out the conjunctions that imply $z \wedge \bar{z}$, we have 2 kinds of conjunctions: the ones where j literals are in Y and the remaining literal is in $\{z, \bar{z}\}$, and those with all literals in Y. Or: $C^{j+1}(Y \cup \{z, \bar{z}\}) \equiv z \wedge C^j(Y) \vee \bar{z} \wedge C^j(Y) \vee C^{j+1}(Y) \equiv C^j(Y)$, by Lemma 3a. □

Before we enunciate the next Theorem, we need the following Lemma, where C_k^j denotes $C^j(y_1, \ldots, y_k)$:

Lemma 5. *Let i and k be integers, with $0 \leq i < k$. Then*

$$p(y_{k+1} \wedge C_k^i) + p(C_k^{i+1}) = p(C_{k+1}^{i+1}) + p(y_{k+1} \wedge C_k^{i+1}) \ .$$

Proof. Directly from Kolmogorov's axioms, we have:

$$p(y_{k+1} \wedge C_k^i) + p(C_k^{i+1}) = p(y_{k+1} \wedge C_k^i \vee C_k^{i+1}) + p(y_{k+1} \wedge C_k^i \wedge C_k^{i+1}) \ .$$

From Lemma 3, we know that $C_k^{i+1} \vDash C_k^i$. Then:

$$p(y_{k+1} \wedge C_k^i \wedge C_k^{i+1}) = p(y_{k+1} \wedge C_k^{i+1}) \ .$$

From the C-formulas definition, we note that:

$$y_{k+1} \wedge C_k^i \vee C_k^{i+1} \equiv C_{k+1}^{i+1} \ .$$

Finally, we obtain:

$$p(y_{k+1} \wedge C_k^i) + p(C_k^{i+1}) = p(C_{k+1}^{i+1}) + p(y_{k+1} \wedge C_k^{i+1}) \ . \qquad \square$$

Theorem 3. *Let $\{y_1, \ldots, y_k\}$ be a set of literals. Then:*

$$p(y_1) + \ldots + p(y_k) = p(C^1(y_1, \ldots, y_k)) + \ldots + p(C^k(y_1, \ldots, y_k)) \ .$$

Proof. The proof proceeds by induction in k, with C_k^j denoting $C^j(y_1, \ldots, y_k)$:
Induction basis: $k = 1$, $p(y_1) = C^1(y_1)$ trivially.
Induction hypothesis: $k = j \geq 1$, $p(y_1) + \ldots + p(y_j) = p(C_j^1) + \ldots + p(C_j^j)$.
Induction step: $k = j+1$; starting from the induction hypothesis, we sum $p(y_{j+1})$ to both sides of equality:

$$p(y_1) + \ldots + p(y_j) + p(y_{j+1}) = p(C_j^1) + \ldots + p(C_j^j) + p(y_{j+1}) \ .$$

As y_{j+1} is equivalent to $y_{j+1} \wedge C_j^0$, we apply Lemma 5:

$$p(C_j^1) + p(y_{j+1}) = p(C_{j+1}^1) + p(y_{j+1} \wedge C_j^1) \ .$$

So we obtain:

$$p(y_1) + \ldots + p(y_{j+1}) = p(C_{j+1}^1) + p(y_{j+1} \wedge C_j^1) + p(C_j^2) + \ldots + p(C_j^j) \ .$$

In an analogous way, we apply Lemma 5 $j-1$ times, obtaining:

$$p(y_1) + \ldots + p(y_{j+1}) = p(C_{j+1}^1) + \ldots + p(C_{j+1}^j) + p(y_{j+1} \wedge C_j^j) \ .$$

Noting that $y_{j+1} \wedge C_j^j \equiv C_{j+1}^{j+1}$, we finally have:

$$p(y_1) + \ldots + p(y_{j+1}) = p(C^1(y_1, \ldots, y_{j+1})) + \ldots + p(C^{j+1}(y_1, \ldots, y_{j+1})) \ ,$$

as desired. $\square$

Having presented the C-formulas, and with Theorem 3 in hand, we can enunciate the Theorem that finally generalizes the Lemma 2:

Theorem 4. *Let $\{y_1, \ldots, y_k\}$ be a set of literals, and let Ψ be a probability assignment to these literals. If $\sum_{i=1}^{k} p(y_i) > j-1$, then $C^j(y_1, \ldots, y_k) \in \Psi^{\approx}$.*

Proof. In one hand, from Theorem 3, we have:

$$\sum_{i=1}^{k} p(y_i) = \sum_{i=1}^{k} p(C^i(y_1, \ldots, y_k)) > j-1 \ .$$

As $p(\alpha) \leq 1$ for all formula α, $\sum_{i=1}^{j-1} p(C^i(y_1, \ldots, y_k)) \leq j-1$, for any $j \leq k$, it follows that $\sum_{i=j}^{k} p(C^i(y_1, \ldots, y_k)) > 0$.

In other hand, from Lemma 3, it follows that $C^k(y_1, \ldots, y_k) \models \ldots \models C^j(y_1, \ldots, y_k)$, because $j \leq k$, and hence:

$$\begin{gathered} p(C^j(y_1, \ldots, y_k)) \geq \ldots \geq p(C^k(y_1, \ldots, y_k)) \ ; \\ (k-j+1)p(C^j(y_1, \ldots, y_k)) \geq \textstyle\sum_{i=j}^{k} p(C^i(y_1, \ldots, y_k)) > 0 \ ; \\ p(C^j(y_1, \ldots, y_k)) > 0 \ . \end{gathered}$$

We conclude that $\Psi \mid\!\approx C^j(y_1, \ldots, y_k)$, and therefore $C^j(y_1, \ldots, y_k) \in \Psi^{\approx}$. □

5 A Conjecture and Its Refutation

Let (Γ, Ψ) be a PSAT instance. In one hand, from Theorem 2, if $\{\alpha\} \cup \Gamma$ is classically unsatisfiable, for a C-formula $\alpha \in \Psi^{\approx}$, then (Γ, Ψ) is probabilistically unsatisfiable. In other hand, if each formula $\alpha \in \Psi^{\approx}$ were implied by a formula $C^j(y_1, \ldots, y_k)$, such that $\sum_{i=1}^{k} p(y_i) > j-1$, the probabilistic unsatisfiability of (Γ, Ψ) would yield the classical unsatisfiability of $\{C^j(y_1, \ldots, y_k)\} \cup \Gamma$, for one C-formula in that condition. Having this in mind, we conjecture the following:

Conjecture 1. *If (Γ, Ψ) is an unsatisfiable PSAT instance, then there is a C-formula $C^j(y_1, \ldots, y_k)$, with $\sum_{i=1}^{k} p(y_i) > j-1$, such that $\{C^j(y_1, \ldots, y_k)\} \cup \Gamma$ is classically unsatisfiable.*

Refutation. Our refutation will be built by presenting a counterexample PSAT instance. Let's consider the PSAT instance $\Delta = (\Gamma, \Psi)$, where Γ is a set with 1 formula, from classical propositional logic, over 4 boolean variables $x_1, \ldots, x_4$. To simplify the writing, if α and β are formulas, then $\alpha\beta$ denotes $\alpha \wedge \beta$, and $\bar{\alpha}$ denotes $\neg\alpha$:

$$\Gamma = \{x_1x_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4\} \ .$$

And Ψ is the following probability assignment to the boolean variables:

$$\Psi = \{p(x_1) = 0.47, \quad p(x_2) = 0.40, \quad p(x_3) = 0.46 \quad \text{and} \quad p(x_4) = 0.05\} \ .$$

It follows that, from Kolmogorov's axioms:

$$p(\bar{x}_1) = 0.53, \quad p(\bar{x}_2) = 0.60, \quad p(\bar{x}_3) = 0.54 \quad \text{and} \quad p(\bar{x}_4) = 0.95 \ .$$

Let $v_k : \{x_1, \ldots, x_4\} \to \{0, 1\}$, $1 \leq k \leq 5$, be the only truth assignments to satisfy Γ, such that v_1 satisfies $x_1x_2x_3x_4$, v_2 satisfies $x_1\bar{x}_2\bar{x}_3\bar{x}_4$, v_3 satisfies $\bar{x}_1x_2\bar{x}_3\bar{x}_4$, v_4 satisfies $\bar{x}_1\bar{x}_2x_3\bar{x}_4$, and v_5 satisfies $x_1\bar{x}_2x_3x_4$.

To note the unsatisfiability of $\Delta = (\Gamma, \Psi)$, we will show a formula $\alpha \in \Psi^{\approx}$, such that $\{\alpha\} \cup \Gamma$ is unsatisfiable. As $p(x_1) + p(x_2) + p(x_3) = 1.33$, by Theorem 3 we have $p(C^1(x_1, x_2, x_3)) + p(C^2(x_1, x_2, x_3)) + p(C^3(x_1, x_2, x_3)) = 1.33$. With $p(C^1(x_1, x_2, x_3)) \leq 1$ and, by Lemma 3a, $p(C^2(x_1, x_2, x_3)) \geq p(C^3(x_1, x_2, x_3))$, it follows $p(C^2(x_1, x_2, x_3)) \geq 0.165$. Thus $p(\bar{x}_4) + p(C^2(x_1, x_2, x_3)) > 1$ and $p(\bar{x}_4 \wedge C^2(x_1, x_2, x_3)) > 0$. Let α be $\bar{x}_4 \wedge C^2(x_1, x_2, x_3)$. We note that $\alpha \in \Psi^{\approx}$ and $\{\alpha\} \cup \Gamma$ is unsatisfiable. Then, by Theorem 2, Δ is probabilistically unsatisfiable.

Now we have to exhaustively show that, for each formula $C^j(y_1, \ldots, y_k)$ with $\sum_{i=1}^{k} p(y_i) > j - 1$, $\{C^j(y_1, \ldots, y_k)\} \cup \Gamma$ is satisfiable. Remembering Lemma 3a, being Y a set of literals, if $0 \leq j' < j$, then $C^j(Y) \vDash C^{j'}(Y)$. For each set of literals $Y = \{y_1, \ldots, y_k\}$, we define $j_{max}(Y) = \lceil \sum_{i=1}^{k} p(y_i) \rceil$ and denote $C^{j_{max}(Y)}(Y)$ by $C^{j_{max}}(Y)$. So if $0 \leq j < j_{max}(Y)$, then $C^{j_{max}}(Y) \vDash C^j(Y)$, and if $j_{max}(Y) < j$, then $\sum_{i=1}^{k} p(y_i) \leq j - 1$. Thus for each set of literals Y, it's enough to verify the satisfiability of $\{C^{j_{max}}(Y)\} \cup \Gamma$.

With 4 variables, we have 8 different literals, that yield $2^8 = 256$ possible sets of literals to be checked. We easily note that the empty set doesn't need to be verified, because $\sum_{y \in \emptyset} p(y) = 0 = j_{max}(\emptyset)$, and $C^0(Y) = TRUE$ is satisfied by any truth assignment. Considering now sets with one literal, whose $j_{max} = 1$, note that each of the 8 literals is true either in v_3, or in v_5, so we don't need to check theses sets also. Furthermore, if $z, \bar{z} \notin Y$, then $j_{max}(Y \cup \{z, \bar{z}\}) = \lceil p(z) + p(\bar{z}) + \sum_{y \in Y} p(y) \rceil = j_{max}(Y) + 1$. So, by Lemma 4, $C^{j_{max}}(Y \cup \{z, \bar{z}\}) \equiv C^{j_{max}}(Y)$, thus we don't need to check sets with 2 opposite literals. The 3 tables below show the 72 remaining possible sets of literals over $\{x_1, \ldots, x_4\}$, organized by the set length. Each row presents a set Y of literals, the sum of these literals probabilities, $\sum_{y \in Y} p(y)$, the j_{max} defined by this set and the truth assignment that satisfies $\{C^{j_{max}}(Y)\} \cup \Gamma$. Each truth assignment is represented by the conjunction it is the only one to satisfy.

And so we finish the refutation, with an unsatisfiable PSAT instance, (Γ, Ψ), where for each formula $C^j(Y)$, such that $\sum_{y \in Y} p(y) > j - 1$, we have shown a truth assignment that satisfies $\{C^j(Y)\} \cup \Gamma$. □

Note that Theorems 3 and 4 also hold for formulas, not only for atoms. The formula α presented as "witness" for the unsatisfiability of PSAT instance (Γ, Ψ), from Conjecture 1 refutation, could be written as a C-formula of formulas: $\alpha \equiv \neg x_4 \wedge C^2(x_1, x_2, x_3) \equiv C^2(\neg x_4, C^2(x_1, x_2, x_3))$. As $p(\neg x_4) + (C^2(x_1, x_2, x_3)) > 1$, by Theorem 4, $\Psi \not\approx C^2(\neg x_4, C^2(x_1, x_2, x_3))$, but this C-formula is inconsistent with Γ, as we have shown.

Table 1. Sets with 4 literals

set of literals Y	$\sum_{y\in Y} p(y)$	j_{max}	v_i	**set of literals Y**	$\sum_{y\in Y} p(y)$	j_{max}	v_i
$\{\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4\}$	2.62	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, \bar{x}_2, \bar{x}_3, x_4\}$	1.72	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$
$\{\bar{x}_1, \bar{x}_2, x_3, \bar{x}_4\}$	2.54	3	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{\bar{x}_1, \bar{x}_2, x_3, x_4\}$	1.64	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$
$\{\bar{x}_1, x_2, \bar{x}_3, \bar{x}_4\}$	2.42	3	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, x_2, \bar{x}_3, x_4\}$	1.52	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$
$\{\bar{x}_1, x_2, x_3, \bar{x}_4\}$	2.34	3	$\bar{x}_1x_2\bar{x}_3\bar{x}4$	$\{\bar{x}_1, x_2, x_3, x_4\}$	1.44	2	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_2, \bar{x}_3, \bar{x}_4\}$	2.56	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, \bar{x}_2, \bar{x}_3, x_4\}$	1.66	2	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_2, x_3, \bar{x}_4\}$	2.48	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, \bar{x}_2, x_3, x_4\}$	1.58	2	$x_1x_2x_3x_4$
$\{x_1, x_2, \bar{x}_3, \bar{x}_4\}$	2.36	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, x_2, \bar{x}_3, x_4\}$	1.46	2	$x_1x_2x_3x_4$
$\{x_1, x_2, x_3, \bar{x}_4\}$	2.28	3	$x_1x_2x_3x_4$	$\{x_1, x_2, x_3, x_4\}$	1.38	2	$x_1x_2x_3x_4$

Table 2. Sets with 3 literals

set of literals Y	$\sum_{y\in Y} p(y)$	j_{max}	v_i	**set of literals Y**	$\sum_{y\in Y} p(y)$	j_{max}	v_i
$\{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$	1.67	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, \bar{x}_2, x_3\}$	1.59	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$
$\{\bar{x}_1, x_2, \bar{x}_3\}$	1.47	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, x_2, x_3\}$	1.39	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$
$\{x_1, \bar{x}_2, \bar{x}_3\}$	1.61	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, \bar{x}_2, x_3\}$	1.53	2	$x_1\bar{x}_2x_3x_4$
$\{x_1, x_2, \bar{x}_3\}$	1.41	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, x_2, x_3\}$	1.33	2	$x_1x_2x_3x_4$
$\{\bar{x}_1, \bar{x}_2, \bar{x}_4\}$	2.08	3	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{\bar{x}_1, \bar{x}_2, x_4\}$	1.18	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$
$\{\bar{x}_1, x_2, \bar{x}_4\}$	1.88	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, x_2, x_4\}$	0.98	1	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_2, \bar{x}_4\}$	2.02	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, \bar{x}_2, x_4\}$	1.12	2	$x_1x_2x_3x_4$
$\{x_1, x_2, \bar{x}_4\}$	1.82	2	$x_1x_2x_3x_4$	$\{x_1, x_2, x_4\}$	0.92	1	$x_1x_2x_3x_4$
$\{\bar{x}_1, \bar{x}_3, \bar{x}_4\}$	2.02	3	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, \bar{x}_3, x_4\}$	1.12	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$
$\{\bar{x}_1, x_3, \bar{x}_4\}$	1.94	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{\bar{x}_1, x_3, x_4\}$	1.04	2	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_3, \bar{x}_4\}$	1.96	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, \bar{x}_3, x_4\}$	1.06	2	$x_1x_2x_3x_4$
$\{x_1, x_3, \bar{x}_4\}$	1.88	2	$x_1x_2x_3x_4$	$\{x_1, x_3, x_4\}$	0.98	1	$x_1x_2x_3x_4$
$\{\bar{x}_2, \bar{x}_3, \bar{x}_4\}$	2.09	3	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_2, \bar{x}_3, x_4\}$	1.19	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$
$\{\bar{x}_2, x_3, \bar{x}_4\}$	2.01	3	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{\bar{x}_2, x_3, x_4\}$	1.11	2	$x_1x_2x_3x_4$
$\{x_2, \bar{x}_3, \bar{x}_4\}$	1.89	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_2, \bar{x}_3, x_4\}$	0.99	1	$x_1x_2x_3x_4$
$\{x_2, x_3, \bar{x}_4\}$	1.81	2	$x_1x_2x_3x_4$	$\{x_2, x_3, x_4\}$	0.91	1	$x_1x_2x_3x_4$

Table 3. Sets with 2 literals

set of literals Y	$\sum_{y\in Y} p(y)$	j_{max}	v_i	**set of literals Y**	$\sum_{y\in Y} p(y)$	j_{max}	v_i
$\{\bar{x}_1, \bar{x}_2\}$	1.13	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{\bar{x}_1, x_2\}$	0.93	1	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_2\}$	1.07	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, x_2\}$	0.87	1	$x_1x_2x_3x_4$
$\{\bar{x}_1, \bar{x}_3\}$	1.07	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, x_3\}$	0.99	1	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_3\}$	1.01	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, x_3\}$	0.93	1	$x_1x_2x_3x_4$
$\{\bar{x}_1, \bar{x}_4\}$	1.48	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_1, x_4\}$	0.58	1	$x_1x_2x_3x_4$
$\{x_1, \bar{x}_4\}$	1.42	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{x_1, x_4\}$	0.52	1	$x_1x_2x_3x_4$
$\{\bar{x}_2, \bar{x}_4\}$	1.55	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_2, x_4\}$	0.65	1	$x_1x_2x_3x_4$
$\{x_2, \bar{x}_4\}$	1.35	2	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\{x_2, x_4\}$	0.45	1	$x_1x_2x_3x_4$
$\{\bar{x}_2, \bar{x}_3\}$	1.14	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_2, x_3\}$	1.06	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$
$\{x_2, \bar{x}_3\}$	0.94	1	$x_1x_2x_3x_4$	$\{x_2, x_3\}$	0.86	1	$x_1x_2x_3x_4$
$\{\bar{x}_3, \bar{x}_4\}$	1.49	2	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\{\bar{x}_3, x_4\}$	0.59	1	$x_1x_2x_3x_4$
$\{x_3, \bar{x}_4\}$	1.41	2	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\{x_3, x_4\}$	0.51	1	$x_1x_2x_3x_4$

6 Conclusion

In this paper, we investigated the relation between probabilistic satisfiability and classical satisfiability. For this, we presented the Atomic Normal Form and defined a Probabilistic Entailment relation ($\models\!\!\!\approx$). We defined C-formulas, which have shown ubiquity in the probabilistic satisfiability study. Finally, we conjectured the completeness of only looking at C-formulas probabilistically entailed to decide the satisfiability of a PSAT instance, and we refuted it with a counterexample.

The exponential number of C-formulas to investigate doesn't allow its exhaustive use in a possible polynomial reduction from PSAT to SAT. However, the founded results seem to be useful on PSAT study, bringing it back to logic. The Atomic Normal Form might be useful to standardize PSAT instances, in order to compare numeric outputs from algorithms that solve the problem. The introduced probabilistic entailment relation enables the presentation of a "witness" formula for the unsatisfiability of a PSAT instance, which can be used in proving probabilistic unsatisfiability, using classical unsatisfiability.

As the efficiency of algorithms that solve PSAT is considerably lower than those from algorithms for another NP-complete problems (like SAT), we believe there is a lot of work to be done. A possible approach would be the polynomial reduction to SAT, using the concepts we presented here together with linear algebra techniques to explore PSAT, as it can be seen as a linear programming problem.

References

1. Boole, G.: An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities. Walton and Maberley, Londres (1854); reprint: Dover, Nova York (1958)
2. Cook, S.: The Complexity of Theorem Proving Procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158. ACM, New York (1971)
3. de Finetti, B.: Problemi determinati e indeterminati nel calcolo delle probabilita. Rendiconti Reale Accademia dei Lincei 6(XII), 367–373 (1930)
4. Georgakopoulos, G., Kavvadias, D., Papadimitriou, C.H.: Probabilistic Satisfiability. Journal of Complexity 4, 1–11 (1988)
5. Hailperin, T.: Best possible inequalities for the probability of a logical function of events. Amor. Math. Monthly 72, 343–359 (1965)
6. Hansen, P., Jaumard, B.: Probabilistic satisfiability. Technical Report G-96-31, Les Cahiers du GERAD, École Polytechnique de Montréal (1996)
7. Hansen, P., Jaumard, B., Nguetse, G.D., Aragão, M.P.: Models and Algorithms for Probabilistic and Bayesian Logic. In: IJCAI 1995: Proceedings of the 14th International Joint Conference on Artificial Intelligence, p. 1868. Morgan Kaufmann, San Francisco (1995)
8. Nilsson, N.J.: Probabilistic Logic. Artificial Intelligence 28, 71–87 (1986)
9. Prasolov, V.V., Tikhomirov, V.M.: Geometry, Translations of Mathematical Monographs, vol. 200, pp. 43–44. American Mathematical Society, Providence (2001)

A Logic for Conceptual Hierarchies

Norihiro Kamide

Waseda Institute for Advanced Study, Waseda University,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, Japan
logician-kamide@aoni.waseda.jp

Abstract. We propose a proof-theoretical way of obtaining detailed and precise information on conceptual hierarchies. The notion of concept finding proof, which represents a hierarchy of concepts, is introduced based on a substructural logic with mingle and strong negation. Mingle, which is a structural inference rule, is used to represent a process for finding a more general (or specific) concept than some given concepts. Strong negation, which is a negation connective, is used to represent a concept inverse operator. The problem for constructing a concept finding proof is shown to be decidable in PTIME.

1 Introduction

Formal concept analysis (FCA) [4] is well-known as a useful theory to analyze hierarchies of concepts by *concept lattices* (CLs). Drawing CLs as diagrams has been grown into a flourishing industry. CLs are thus regarded as a useful algebraic (or model-theoretical) interpretation of conceptual hierarchies. Compared with CLs, a proof-theoretical interpretation of conceptual hierarchies has not yet been studied. Since some proof-theoretical methods are useful for providing an intuitive interpretation and an automated (or mechanical) theorem proving tool, establishing a proof-theoretical foundation of conceptual hierarchies is useful for both relevant understanding and systematic mechanization of conceptual hierarchies.

In this paper, we propose a proof-theoretical way of obtaining detailed and precise information on conceptual hierarchies. The notion of *concept finding proof*, which represents a hierarchy of concepts, is introduced based on a *substructural logic with mingle and strong negation*. A concept finding proof represents a "process" or "mechanism" for finding a more general (or specific) concept C than given concepts $C_1, ..., C_n$. Finding more general concepts is known to be an important issue in the area of *description logic* (DL) [2]. Such a general concept discussed in the DL community is called a *least common subsumer*. A proof-theoretical formalization of the finding process of general (or specific) concepts is a new approach introduced in this paper. The result of this paper formally explains a process or mechanism for finding a more general (or specific) concept.

Substructural logics with mingle [9,1,7,8] are logics weaker than Gentzen's sequent calculus LK for classical logic. *Mingle*, which was originally introduce in [9], is a structural rule of the form:

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 303–312, 2010.

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \quad \Gamma_2 \Rightarrow \Delta_2}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2} \text{ (mingle).}$$

This rule and the corresponding axiom scheme $\alpha \rightarrow \alpha \rightarrow \alpha$ has been studied in formalizing relevant human reasoning [9,1], grammatical reasoning [7] and reasoning about communication-merge in process algebras [8]. In this paper, we show that a substructural logic with mingle is more appropriate than classical logic for representing hierarchies of concepts. Mingle is used to represent a process for finding a more general (or specific) concept than some given concepts.

Substructural logics with mingle and strong negation, which are extensions of substructural logics with mingle by adding strong negation, were studied in [6]. *Strong negation* is a negation connective originally proposed in [3] and has various applications to inconsistency-tolerant reasoning [10]. In this paper, we show that strong negation is useful to represent a *concept inverse operator*.

The contents of this paper are then summarized as follows. In Section 2, a fragment LM of a substructural logic with mingle and strong negation is introduced, and the cut-elimination and decidability theorems for LM are reviewed. A rule, which is required to formalize a concept inverse operator, is also introduced, and the admissibility of this rule is presented as a new result of this paper. In Section 3, some definitions with a practical example are given, and a problem called a *concept finding problem* is informally explained. This problem is a problem for finding a more general (or specific) concept and for drawing a *concept finding tree* and a *concept finding proof*. A concept finding tree is a tree which represents a hierarchy of concepts. A concept finding proof, which is a proof in LM, is a proof-theoretical counter part of a concept finding tree. In Section 4, the formal definition of the concept finding problem is given, and the concept finding problem is shown to be decidable in PTIME.

2 Substructural Logic with Mingle and Strong Negation

Formulas are constructed from countably many propositional variables, $\wedge$ (conjunction), $\vee$ (disjunction) and $\sim$ (strong negation). Lower-case letters $p, q, ...$ are used to denote propositional variables, Greek lower-case letters $\alpha, \beta, ...$ are used to denote formulas and Greek capital letters $\Gamma, \Delta, ...$ are used to represent finite (possibly empty) sets of formulas. An expression $\sim\Gamma$ is used to denote the set $\{\sim\gamma \mid \gamma \in \Gamma\}$. We write $A \equiv B$ to indicate the syntactical identity between A and B. An expression of the form $\Gamma \Rightarrow \Delta$ is called a *sequent*. An expression $L \vdash S$ is used to denote the fact that a sequent S is provable in a sequent calculus L. A rule R of inference is said to be *admissible* in a sequent calculus L if the following condition is satisfied: for any instance

$$\frac{S_1 \cdots S_n}{S}$$

of R, if $L \vdash S_i$ for all i, then $L \vdash S$.

A fragment LM of a substructural logic with mingle and strong negation is defined as follows.

Definition 1. *The initial sequents of* LM *are of the form:*

$$\alpha \Rightarrow \alpha.$$

The structural rules of LM *are of the form:*

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\text{cut}) \qquad \frac{\Gamma_1 \Rightarrow \Delta_1 \quad \Gamma_2 \Rightarrow \Delta_2}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2}\ (\text{mingle}).$$

The logical inference rules of LM *are of the form:*

$$\frac{\alpha, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge\text{left1}) \qquad \frac{\beta, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge\text{left2})$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \wedge \beta}\ (\wedge\text{right}) \qquad \frac{\alpha, \Gamma \Rightarrow \Delta \quad \beta, \Gamma \Rightarrow \Delta}{\alpha \vee \beta, \Gamma \Rightarrow \Delta}\ (\vee\text{left})$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\vee\text{right1}) \qquad \frac{\Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\vee\text{right2})$$

$$\frac{\alpha, \Gamma \Rightarrow \Delta}{\sim\sim\alpha, \Gamma \Rightarrow \Delta}\ (\sim\text{left}) \qquad \frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma \Rightarrow \Delta, \sim\sim\alpha}\ (\sim\text{right})$$

$$\frac{\sim\alpha, \Gamma \Rightarrow \Delta \quad \sim\beta, \Gamma \Rightarrow \Delta}{\sim(\alpha \wedge \beta), \Gamma \Rightarrow \Delta}\ (\sim \wedge \text{left})$$

$$\frac{\Gamma \Rightarrow \Delta, \sim\alpha}{\Gamma \Rightarrow \Delta, \sim(\alpha \wedge \beta)}\ (\sim \wedge \text{right1}) \qquad \frac{\Gamma \Rightarrow \Delta, \sim\beta}{\Gamma \Rightarrow \Delta, \sim(\alpha \wedge \beta)}\ (\sim \wedge \text{right2})$$

$$\frac{\sim\alpha, \Gamma \Rightarrow \Delta}{\sim(\alpha \vee \beta), \Gamma \Rightarrow \Delta}\ (\sim \vee \text{left1}) \qquad \frac{\sim\beta, \Gamma \Rightarrow \Delta}{\sim(\alpha \vee \beta), \Gamma \Rightarrow \Delta}\ (\sim \vee \text{left2})$$

$$\frac{\Gamma \Rightarrow \Delta, \sim\alpha \quad \Gamma \Rightarrow \Delta, \sim\beta}{\Gamma \Rightarrow \Delta, \sim(\alpha \vee \beta)}\ (\sim \vee \text{right}).$$

Remark that (mingle) is weaker than the following weakening rules, which are standard structural rules in Gentzen's LK for classical logic:

$$\frac{\Gamma \Rightarrow \Delta}{\alpha, \Gamma \Rightarrow \Delta}\ (\text{we-left}) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \alpha}\ (\text{we-right}).$$

A sequent $q, p \Rightarrow p$ where p and q are distinct propositional variables is provable in LK, but not provable in LM.

Remark that the following contraction rules, which are also standard structural rules in LK, are implicitly assumed in LM since in sequents $\Gamma \Rightarrow \Delta$, the expressions Γ and Δ are *sets* of formulas:

$$\frac{\alpha, \alpha, \Gamma \Rightarrow \Delta}{\alpha, \Gamma \Rightarrow \Delta}\ (\text{co-left}) \qquad \frac{\Gamma \Rightarrow \Delta, \alpha, \alpha}{\Gamma \Rightarrow \Delta, \alpha}\ (\text{co-right}).$$

Proposition 2. *The following rules are admissible in cut-free* LM*:*

$$\frac{\sim\sim\alpha, \Gamma \Rightarrow \Delta}{\alpha, \Gamma \Rightarrow \Delta}\ (\sim\text{left}^{-1}) \quad \frac{\Gamma \Rightarrow \Delta, \sim\sim\alpha}{\Gamma \Rightarrow \Delta, \alpha}\ (\sim\text{right}^{-1}).$$

The cut-elimination and decidability theorems for LM are known (see [6] for some extensions of LM, and see [9,7,8] for some $\sim$-free versions).

We need the following new theorem in order to formalize concept finding proofs with a concept inverse operator.

Theorem 3. *The following rule is admissible in cut-free* LM*:*

$$\frac{\Delta \Rightarrow \Gamma}{\sim\Gamma \Rightarrow \sim\Delta}\ (\text{twist})$$

Proof. By induction on the proofs P of $\Delta \Rightarrow \Gamma$ in cut-free LM. We use Proposition 2. Since the proof is straightforward, the detailed proof is omitted here. ∎

Remark that Theorem 3 does not hold for an extension of LM with the addition of the implication connective $\rightarrow$. Thus, Theorem 3 is regarded as a characteristic property of LM.

3 Examples of Concept Finding Proofs

3.1 Solar System Example

The following explanation (for concepts) and example (for solar system) are from the textbook [5]. The terminologies and definitions concerning concepts are taken from FCA [4], but the definitions obtained are not the same as those of FCA.

A *concept* is considered to be determined by its *extent* and its *intent*: the extent consists of all *objects* belonging to the concept (as the reader belongs to the concept "living person") while the intent is the collection of all *attributes* shared by the objects (as all living persons share the attribute "can breath"). As it is often difficult to list all the objects belonging to a concept and usually impossible to list all its attributes, it is natural to work within a specific *context* in which the objects and attributes are fixed.

The information presented in Table 1 gives a context for the planets of our solar system. The objects are the planets while the attributes are the seven indicated properties relating to size, distance from the sun and existence of a moon. That the i-th object possesses the j-th attribute is indicated by a "x" in the ij-position of the table.

A concept of this context will consist of an ordered pair (α, β), where α (the extent) is a subset of the nine planets and β (the intent) is a subset of the seven properties. To demand that the concept is determined by its extent and by its intent means that β should contain just those properties shared by all the planets in α and, similarly, the planets in α should be precisely those sharing all the properties in β.

Table 1. Solar system

	size			distance from sun		moon	
	small	medium	large	near	far	yes	no
Mercury	x			x			x
Venus	x			x			x
Earth	x			x		x	
Mars	x			x		x	
Jupiter			x		x	x	
Saturn			x		x	x	
Uranus		x			x	x	
Neptune		x			x	x	
Pluto	x				x	x	

We now present the following precise definitions.

Definition 4. *A* finite context *is a triple* (G, M, I) *where* G *and* M *are finite sets and* $I \subseteq G \times M$*. The elements of* G *and* M *are called* objects *and* attributes*, respectively. Then, a* concept *of the finite context* (G, M, I) *is defined to be a pair* (α, β) *where* $\alpha \subseteq G$ *and* $\beta \subseteq M$*. The* extent *of the concept* (α, β) *is* α *while its* intent *is* β*. The set of all concepts of the finite context* (G, M, I) *is denoted by* $B(G, M, I)$*. For concepts* (α_1, β_1) *and* (α_2, β_2) *in* $B(G, M, I)$*,* $(\alpha_1, \beta_1) \leq (\alpha_2, \beta_2)$ *is defined by* $\alpha_1 \subseteq \alpha_2$ *(or equivalently* $\beta_1 \supseteq \beta_2$*). A concept* C_1 *is called a* general concept *of a concept* C_2 *if* $C_2 \leq C_1$ *holds. A concept* C_1 *is called a* specific concept *of a concept* C_2 *if* $C_1 \leq C_2$ *holds.*

Remark that Definition 4 is not the same as that for FCA. The definition of the concept (α, β) in FCA needs the following condition: $\alpha' = \beta$ and $\beta' = \alpha$ where $\alpha' := \{m \in M \mid \forall g \in \alpha\ (g, m) \in I\}$ and $\beta' := \{g \in G \mid \forall g \in \beta\ (g, m) \in I\}$. Moreover, in the definition of concepts in FCA, we do not assume the finiteness condition, i.e., all concepts are finite. We need this finiteness condition in order to obtain an algorithm for concept finding.

3.2 Finding More General Concepts by Mingle

We now consider the following problem:

> For some given concepts (α_i, β_i) $(i \in \{1, 2, ...\})$, find a more general concept (α, β) than the concepts (α_i, β_i). A process for finding such a general concept is called *concept finding*.

In order to find a more general concept (α, β) than two concepts (α_1, β_1) and (α_2, β_2), the following informal rule is useful:

$$\frac{(\alpha_1, \beta_1) \quad (\alpha_2, \beta_2)}{(\alpha_1 \cup \alpha_2, \beta_1 \cap \beta_2)}\ \text{(i-mingle)}$$

where $\alpha = \alpha_1 \cup \alpha_2$ and $\beta = \beta_1 \cap \beta_2$. In this rule, if (α_1, β_1), (α_2, β_2) and (α, β) are concepts, then the condition: $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \leq (\alpha, \beta)$ is satisfied. As we will see, this informal rule is precisely formalized using LM.

In the following, some examples for concept finding are presented based on the planet example displayed in Table 1. Assume the concepts $(\alpha_1, \beta_1), (\alpha_2, \beta_2)$ and (α_3, β_3) where

$\alpha_1 := \{Earth, Mars\}$,
$\beta_1 := \{sizeSmall, distanceNear, moonYes\}$,
$\alpha_2 := \{Mercury, Venus\}$,
$\beta_2 := \{sizeSmall, distanceNear, moonNo\}$,
$\alpha_3 := \{Pluto\}$,
$\beta_3 := \{sizeSmall, distanceFar, moonYes\}$.

Consider the following concrete problem:

Problem 1: Find a more general concept (α, β) than $(\alpha_1, \beta_1), (\alpha_2, \beta_2)$ and (α_3, β_3).
The following is a *concept finding tree* by (i-mingle):

$$\dfrac{\dfrac{(\{Ea, Ma\}, \{sm, ne, yes\}) \quad (\{Me, Ve\}, \{sm, ne, no\})}{(\{Ea, Ma, Me, Ve\}, \{sm, ne\})} \qquad (\{Pl\}, \{sm, far, yes\})}{(\{Ea, Ma, Me, Ve, Pl\}, \{sm\})}$$

where Ea, Ma, Me, Ve, Pl, sm, ne, far, yes and no are abbreviations of $Earth$, $Mars$, $Mercury$, $Venus$, $Pluto$, $sizeSmall$, $distanceNear$, $distanceFar$, $moonYes$ and $moonNo$, respectively. A more general concept than (α_1, β_1), (α_2, β_2) and (α_3, β_3) is thus the concept (α, β) where

$\alpha = \{Earth, Mars, Mercury, Venus, Pluto\}$,
$\beta = \{sizeSmall\}$.

In the following, we will give a logical presentation of the concept finding tree displayed above. Based upon the same example, we will obtain a *concept finding proof* in cut-free LM. Before obtaining such a proof, the following interpretations must be assumed:

1. a concept (α_i, β_i) is interpreted as the sequent $\beta_i \Rightarrow \alpha_i$,
2. the set operations $\cap$ and $\cup$ are interpreted as the connectives $\wedge$ and $\vee$, respectively.

Then, the following concept finding proof in cut-free LM is obtained:[1]

$$\dfrac{\dfrac{\dfrac{\dfrac{\beta_1 \Rightarrow \alpha_1 \quad \beta_2 \Rightarrow \alpha_2}{\beta_1, \beta_2 \Rightarrow \alpha_1, \alpha_2}\,(\vee\text{right}1, 2)}{\beta_1, \beta_2 \Rightarrow \alpha_1 \vee \alpha_2}\,(\wedge\text{left}1, 2)}{\beta_1 \wedge \beta_2 \Rightarrow \alpha_1 \vee \alpha_2} \qquad \beta_3 \Rightarrow \alpha_3}{\beta_1 \wedge \beta_2, \beta_3 \Rightarrow \alpha_1 \vee \alpha_2, \alpha_3}\,(\text{mingle})}{\beta_1 \wedge \beta_2, \beta_3 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3}\,(\vee\text{right}1, 2)}{\beta_1 \wedge \beta_2 \wedge \beta_3 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3}\,(\wedge\text{left}1, 2)$$

[1] In the proof, the expression ($\vee$right1, 2) means some applications of ($\vee$right1), ($\vee$right2) and the contraction rule. The expression ($\wedge$left1, 2) is also in the same manner. These expressions are just abbreviations and similar expressions are also used in the following discussions.

where $\beta \equiv \beta_1 \wedge \beta_2 \wedge \beta_3$ and $\alpha \equiv \alpha_1 \vee \alpha_2 \vee \alpha_3$ are interpreted as

$$\beta = \beta_1 \cap \beta_2 \cap \beta_3 = \{sizeSmall\},$$
$$\alpha = \alpha_1 \cup \alpha_2 \cup \alpha_3 = \{Earth, Mars, Mercury, Venus, Pluto\},$$

respectively. We thus obtain the same result as in the concept finding tree by (i-mingle).

Moreover, consider the following concrete problem:

Problem 2: Find a more general concept (α', β') than $(\alpha_1, \beta_1), (\alpha_2, \beta_2), (\alpha_3, \beta_3)$ and (α_4, β_4) where

$$\alpha_4 := \{Jupiter, Saturn\}.$$
$$\beta_4 := \{sizeLarge, distanceFar, moonYes\}.$$

Let Q be the concept finding tree for Problem 1. The following is a concept finding tree for Problem 2:

$$\frac{\begin{array}{c}\vdots\ Q \\ (\{Ea, Ma, Me, Ve, Pl\}, \{sm\}) \quad (\{Ju, Sa\}, \{la, far, yes\})\end{array}}{(\{Ea, Ma, Me, Ve, Pl, Ju, Sa\}, \emptyset)}$$

where Ju, Sa and la are abbreviations of $Jupiter$, $Saturn$ and $sizeLarge$, respectively. We can see that the resulting intent β' is $\emptyset$. This means that there is no such a general concept (α', β').

Let P be the concept finding proof for Problem 1. Then, the following concept finding proof for Problem 2 is obtained:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ P \\ \beta_1 \wedge \beta_2 \wedge \beta_3 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3 \quad \beta_4 \Rightarrow \alpha_4\end{array}}{\beta_1 \wedge \beta_2 \wedge \beta_3, \beta_4 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3, \alpha_4}\ (\text{mingle})}{\beta_1 \wedge \beta_2 \wedge \beta_3, \beta_4 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4}\ (\vee\text{right}1, 2)}{\beta_1 \wedge \beta_2 \wedge \beta_3 \wedge \beta_4 \Rightarrow \alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4}\ (\wedge\text{left}1, 2)$$

where $\beta' \equiv \beta_1 \wedge \beta_2 \wedge \beta_3 \wedge \beta_4$ is interpreted as $\emptyset$.

3.3 Finding More Specific Concepts by Strong Negation

We now consider the following problem:

> For some given concepts (α_i, β_i) $(i \in \{1, 2, ...\})$, find a more specific concept (α, β) than the concepts (α_i, β_i). A process for finding such a specific concept is also called *concept finding*.

We can find a more specific concept C than some concepts C_i in a similar way as for a more general concept. To find such a specific concept, the notion of the *inverse concept*

$$(\alpha, \beta)^{-1} := (\beta, \alpha)$$

of a concept (α, β) is introduced, where $\cdot^{-1}$ is called the *concept inverse operator*. In order to find a more specific concept C than concepts C_i, we can apply (i-mingle) to the inverses C_i^{-1}. This derives the inverse C^{-1} of the required concept C.

In order to formalize the concept inverse operator $\cdot^{-1}$, the following informal rule is useful:

$$\frac{(\beta, \alpha)}{(\alpha, \beta)^{-1}}\ \text{(i-twist)}$$

which corresponds to the inference rule (twist) in LM, which was shown to be admissible in Theorem 3.

Consider the following concrete problem:

Problem 3: Find a more specific concept (α'', β'') than (α_5, β_5) and (α_6, β_6) where

$\alpha_5 := \{Earth, Mars, Mercury, Venus\}$,
$\beta_5 := \{sizeSmall, distanceNear\}$,
$\alpha_6 := \{Earth, Mars, Mercury, Venus, Pluto\}$.
$\beta_6 := \{sizeSmall\}$.

The following is a concept finding tree based on the same abbreviations as in the previous examples:

$$\frac{\dfrac{(\{Ea, Ma, Me, Ve\}, \{sm, ne\})^{-1} \quad (\{Ea, Ma, Me, Ve, Pl\}, \{sm\})^{-1}}{(\{sm, ne\}, \{Ea, Ma, Me, Ve\})}\ \text{(i-mingle)}}{(\{Ea, Ma, Me, Ve\}, \{sm, ne\})^{-1}}\ \text{(i-twist).}$$

Then, a more specific concept than (α_5, β_5) and (α_6, β_6) is (α'', β'') where

$\alpha'' = \{Earth, Mars, Mercury, Venus\}$,
$\beta'' = \{sizeSmall, distanceNear\}$.

Prior to give the corresponding concept finding proof, the following interpretation must be assumed:

An inverse concept $(\beta_i, \alpha_i)^{-1}$ is interpreted as the sequent $\sim\alpha_i \Rightarrow \sim\beta_i$.

Then, the following concept finding proof is obtained:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\sim\alpha_5 \Rightarrow \sim\beta_5 \quad \sim\alpha_6 \Rightarrow \sim\beta_6}{\sim\alpha_5, \sim\alpha_6 \Rightarrow \sim\beta_5, \sim\beta_6}\ \text{(mingle)}}{\sim\alpha_5, \sim\alpha_6 \Rightarrow \sim(\beta_5 \wedge \beta_6)}\ (\sim \wedge \text{right}1, 2)}{\sim(\alpha_5 \vee \alpha_6) \Rightarrow \sim(\beta_5 \wedge \beta_6)}\ (\sim \vee \text{left}1, 2)}{\sim\sim(\beta_5 \wedge \beta_6) \Rightarrow \sim\sim(\alpha_5 \vee \alpha_6)}\ \text{(twist)}}{\sim\sim(\beta_5 \wedge \beta_6) \Rightarrow \alpha_5 \vee \alpha_6}\ (\sim\text{right}^{-1})}{\beta_5 \wedge \beta_6 \Rightarrow \alpha_5 \vee \alpha_6}\ (\sim\text{left}^{-1})$$

where $\beta'' \equiv \beta_5 \wedge \beta_6$ and $\alpha'' \equiv \alpha_5 \vee \alpha_6$ are interpreted as

$\beta'' = \beta_5 \cap \beta_6 = \{sizeSmall, distanceNear\}$,
$\alpha'' = \alpha_5 \cup \alpha_6 = \{Earth, Mars, Mercury, Venus\}$.

4 Decidability of Concept Finding Problem

We now obtain the following precise definitions by summarizing the previous discussions.

Definition 5. *A* concept finding tree *w.r.t. the concepts* (α_i, β_i) $(i \in \{1, 2, ..., n\})$ *is a tree which is generated from all the concepts* (α_i, β_i) *by applying the following rule:*

$$\frac{(\alpha_j, \beta_j) \quad (\alpha_k, \beta_k)}{(\alpha_j \cup \alpha_k, \beta_j \cap \beta_k)} \text{ (i-mingle).}$$

A concept finding proof *w.r.t. the concepts* (α_i, β_i) $(i \in \{1, 2, ..., n\})$ *is a proof in cut-free* LM *such that all the assumptions are the sequents* $\beta_i \Rightarrow \alpha_i$*, and the conclusion is the sequent* $\beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha_1 \vee \cdots \vee \alpha_n$.

Definition 6. *A* concept finding tree *w.r.t. the inverse concepts* $(\beta_i, \alpha_i)^{-1}$ $(i \in \{1, 2, ..., n\})$ *is a tree which is obtained from a concept finding tree w.r.t. the concepts* (α_i, β_i) *by applying the following rule to the root of the tree:*

$$\frac{(\beta, \alpha)}{(\alpha, \beta)^{-1}} \text{ (i-twist).}$$

A concept finding proof *w.r.t. the inverse concepts* $(\beta_i, \alpha_i)^{-1}$ $(i \in \{1, 2, ..., n\})$ *is a proof in cut-free* LM *such that all the assumptions are the sequents* $\sim\alpha_i \Rightarrow \sim\beta_i$*, and the conclusion is the sequent* $\beta_1 \wedge \cdots \wedge \beta_n \Rightarrow \alpha_1 \vee \cdots \vee \alpha_n$.

Definition 7. *The concept finding problem is defined as follows:*

1. *for a finite context* (G, M, I) *and for some concepts* C_i $(i \in \{1, 2, ..., n\})$ *in* $B(G, M, I)$*, find a more general (or specific) concept* C *than the concepts* C_i*,*
2. *construct a concept finding tree w.r.t. the concepts* C_i *or the inverse concepts* C_i^{-1}*,*
3. *construct a concept finding proof w.r.t. the concepts* C_i *or the inverse concepts* C_i^{-1}.

Finding such a general (or specific) concept and constructing a concept finding tree and a concept finding proof are called concept finding.

We then have the following theorem.

Theorem 8. *The concept finding problem is decidable in PTIME, namely, for a finite context* (G, M, I) *and for some concepts* (α_i, β_i) $(i \in \{1, 2, ..., n\})$ *in* $B(G, M, I)$*, we can, in some polynomial steps,*

1. *verify the existence of a more general (or specific) concept* (α, β) *than the concepts* (α_i, β_i)*,*
2. *construct a concept finding tree which has the root* (α, β) *and the leafs* (α_i, β_i) *or* $(\alpha_i, \beta_i)^{-1}$*,*

3. *construct a concept finding proof of* $\beta \Rightarrow \alpha$ *in cut-free* LM *from the assumption sequents* $\beta_i \Rightarrow \alpha_i$ *or* $\sim\alpha_i \Rightarrow \sim\beta_i$.

Proof. We show only (3) and (1) simultaneously. Consider the following procedure:

1. Replace the given concepts (α_i, β_i) by the assumption sequents $\beta_i \Rightarrow \alpha_i$ (for finding a general concept) or $\sim\alpha_i \Rightarrow \sim\beta_i$ (for finding a specific concept).
2. Construct a concept finding proof of a sequent $\beta \Rightarrow \alpha$ from the assumption sequents $\beta_i \Rightarrow \alpha_i$ (for finding a general concept) or $\sim\alpha_i \Rightarrow \sim\beta_i$ (for finding a specific concept), by using cut-free LM.
3. Compute $\alpha \equiv \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$ and $\beta \equiv \beta_1 \vee \beta_2 \vee \cdots \vee \beta_n$ by using the interpretations $\wedge := \cap$ and $\vee := \cup$.
4. If β is not empty, then the resulting pair (α, β) is a required concept. If β is empty, then there is no such a concept.

In this procedure, all steps are finitely performed since, in particular for the step 3, the underlying concepts are finite. Thus, the concept finding problem is decidable. Moreover, we can easily verify that the algorithm presented is in PTIME. Therefore the concept finding problem is decidable in PTIME. ∎

Acknowledgments. I would like to thank Dr. Ken Kaneiwa for his valuable comments. This work was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 20700015.

References

1. Anderson, A.R., Belnap Jr., N.D.: Entailment: The logic of relevance and necessity, vol. 1. Princeton University Press, Princeton (1975)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook: Theory, implementation and applications. Cambridge University Press, Cambridge (2003)
3. Nelson, D.: Constructible falsity. Journal of Symbolic Logic 14, 16–26 (1949)
4. Ganter, B., Wille, R.: Formal concept analysis: Mathematical foundations. Springer, Heidelberg (1999)
5. Davey, B.A., Priestley, H.A.: Introduction to lattice and order, 2nd edn. Cambridge University Press, Cambridge (2002)
6. Kamide, N.: Relevance principle for substructural logics with mingle and strong negation. Journal of Logic and Computation 12(6), 913–928 (2002)
7. Kamide, N.: Substructural logic with mingle. Journal of Logic, Language and Information 11(2), 227–249 (2002)
8. Kamide, N.: Linear logics with communication-merge. Journal of Logic and Computation 15(1), 3–20 (2005)
9. Ohnishi, M., Matsumoto, K.: A system for strict implication. Annals of the Japan Association for Philosophy of Science 2(4), 183–188 (1964)
10. Wansing, H.: The Logic of Information Structures. LNCS, vol. 681, 163 pages. Springer, Heidelberg (1993)

Author Index