

Saying It with Pictures:
a logical landscape of conceptual graphs

ILLC Dissertation Series DS-2001-09



INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: illc@wins.uva.nl
homepage: <http://www.illc.uva.nl/>

Saying It with Pictures: a logical landscape of conceptual graphs

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. J.J.M. Franse
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op woensdag 14 november 2001, te 14.00 uur

door

Gwenael Nang Kerdiles

geboren te Saint-Denis, Ile de la Réunion, France

Promotors: prof.dr. J.F.A.K. van Benthem
prof.dr. A.M.K. Preller
Co-promotor: prof.dr. F.J.M.M. Veltman

Faculteit der Geesteswetenschappen
Afdeling Wijsbegeerte
Universiteit van Amsterdam
Nieuwe Doelenstraat 15
1012 CP Amsterdam
The Netherlands

and

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université Montpellier II
161 rue Ada
34392 Montpellier Cedex 5
France

Copyright © 2001 by Gwenael N. Kerdiles

Printed and bound by Print Partners Ipskamp, Enschede.

ISBN: 90-5776-073-8

“J’ai besoin d’un mouton. Dessine-moi un mouton



Alors j’ai dessiné.

Il regarda attentivement, puis :

- Non! celui-là est déjà très malade. Fais-en un autre.



Je dessinai :

Mon ami sourit gentiment, avec indulgence :

- Tu vois bien... ce n’est pas un mouton, c’est un bélier. Il a des cornes [...]



Et je lançai :

- Ça c’est la caisse. Le mouton que tu veux est dedans.

Mais je fus bien surpris de voir s’illuminer le visage de mon jeune juge :

- C’est tout à fait comme ça que je le voulais!”

[SE46]

A Nina

Contents

Acknowledgments	xi
Abstract	xiii
Introduction	1
1 Diagrams and visual information	3
1.1 Diagrams in logic	4
1.1.1 Frege's graphs	4
1.1.2 Peirce's existential graphs	5
1.1.3 Conceptual graphs	6
1.1.3.1 Positive information	6
1.1.3.2 Negation	8
1.1.3.3 Nested conceptual graphs	9
1.1.4 Concluding remarks	10
1.2 Conceptual graph diagrams and artificial intelligence	11
1.2.1 Semantic networks	11
1.2.2 Description logics	11
1.2.3 Contexts in AI	12
1.2.4 Conclusion	13
1.3 The visual impact of CG drawings	13
1.3.1 Overview of the information drawn	14
1.3.1.1 Partition of the space.	14
1.3.1.2 Spinal structure.	15
1.3.2 Faithfulness of drawings	15
1.3.2.1 Simple graphical components.	16
1.3.2.2 Limited abstraction.	16
1.3.3 Additional information	17
1.3.3.1 Generalisation of semantic conventions.	17
1.3.3.2 Interaction of graphical effects and semantic conventions.	18
1.3.4 Conclusion	18

1.4	Conceptual graphs and the structure of discourse	19
1.4.1	Bindings in conceptual graphs and discourse	19
1.4.1.1	Conjunction of conceptual graphs	20
1.4.1.2	Negation	21
1.4.2	Discourse representation theory	22
1.4.3	Dynamic view on conceptual graph semantics	23
1.4.4	Concluding remarks	24
1.5	Conclusions	25
2	The complexity of reasoning with graphs and fragments	27
2.1	Conceptual graph drawings and formal languages	28
2.2	Computation models	30
2.2.1	Turing machines and equivalent models	30
2.2.2	Time and space are chosen as measured resource variables	31
2.2.3	A detour into precise encodings	31
2.2.3.1	The case of formulae.	32
2.2.3.2	Finite structures.	32
2.2.3.3	Graphs and conceptual graphs.	35
2.2.4	Complexity classes that will be encountered	35
2.2.5	Reductions	38
2.3	The classical decision problem	39
2.3.1	The classical decision problem in fragments of FOL	39
2.3.1.1	Monadic versus polyadic.	40
2.3.1.2	Quantifier prefixes.	40
2.3.1.3	Finite signature	40
2.3.1.4	Modal restrictions.	40
2.4	Benchmarks	41
2.4.1	Four problems	42
2.4.1.1	Satisfiability.	42
2.4.1.2	Consequence.	42
2.4.1.3	Model checking.	43
2.4.1.4	Model comparison.	44
2.4.2	Results that set the scene.	44
2.4.3	Conclusions	46
2.4.3.1	A festival of parameters.	46
2.4.3.2	Complexity collapses and divergences between problems.	47
2.5	A key fragment: existential conjunctive FOL	47
2.5.1	Semantic relationships	48
2.5.2	Complexity	50
2.5.3	Harmless meaning postulates	51
2.5.4	Conclusion	53
2.5.4.1	A road map.	53
2.6	Fragments richer than $FOL_{\{\exists, \wedge\}}$	54
2.6.1	Including atomic negation	54
2.6.1.1	Benchmark problems on models	54
2.6.1.2	Satisfiability problem	54

2.6.1.3	Consequence problem	55
2.6.2	Existential first-order logic	55
2.7	Two steps backwards into tractable worlds	58
2.7.1	Simple conceptual trees	58
2.7.2	A bit of negation: non-interlaced positive and negative information	60
2.8	The travelled FOL landscape	62
2.9	The modal perspective: expressivity at low costs	63
2.9.1	The existential conjunctive guarded fragment	64
2.9.1.1	Note on the complexity of the algorithm	67
2.9.2	The modal cousin of $FOL_{\{\exists, \wedge\}}$	69
2.9.3	Atomic negation at no cost	69
2.9.4	Back to untractability	70
2.9.5	Finite bounds	71
2.9.6	Description logics	71
2.9.7	A landscape of modal complexity results	72
2.10	Conclusions	73
3	Positive information	75
3.1	The cornerstone: simple conceptual graphs	76
3.1.1	Language signature	76
3.1.2	Syntax	79
3.1.2.1	Notations	79
3.1.3	Semantics	80
3.1.3.1	Structures	80
3.1.3.2	Assignments	81
3.1.3.3	Truth definition for SCGs	82
3.1.4	Canonical forms	83
3.1.4.1	Normal forms	83
3.1.4.2	Crazed forms	85
3.1.5	Textual correspondences	87
3.1.5.1	Translation of the support	87
3.1.5.2	Translation to $\mathbf{FOL}_{\{\exists, \wedge\}}$, component by component	88
3.2	Consequence proofs by homomorphism	91
3.2.1	Projection	91
3.2.1.1	Independent sub-projections	92
3.2.2	Canonical model	93
3.2.3	Completeness theorem	96
3.2.4	Tractable simple conceptual trees	98
3.2.4.1	Simple conceptual trees	98
3.2.4.2	SCGs transformable into SCTs	99
3.2.4.3	Tree coverings of crazed graphs	100
3.3	Guarded simple conceptual graphs	103
3.3.0.4	Ordering derived from a guarded covering	104
3.3.0.5	On the FOL-translation of guarded simple graphs	105
3.3.1	Complexity of consequence for guarded SCGs	107
3.3.1.1	Description and complexity of the algorithm	108

3.3.1.2	Soundness and completeness of the algorithm	109
3.4	Conclusions	112
4	Richer pictures	113
4.1	Atomic negation	114
4.1.1	Polarised simple graphs	114
4.1.1.1	Embedding semantics	115
4.1.1.2	Positivising	115
4.1.1.3	Projection	118
4.1.2	Insufficiencies of projection	120
4.1.3	Discriminated polarised simple graphs	121
4.1.4	Completeness and tractability	122
4.1.4.1	Discriminated graphs and the splitting of labour	122
4.1.4.2	Projection completeness	125
4.1.4.3	Tractability harvest	126
4.1.5	Concluding remarks	127
4.2	Full classical negation	128
4.2.1	Conceptual Graphs	128
4.2.1.1	Negation box syntax	128
4.2.1.2	Interpretation of negated graphs	131
4.2.1.3	Translation to FOL	132
4.2.2	Combining tableaux with projections	133
4.2.2.1	Decomposition rules	134
4.2.2.2	Completeness of the calculus	137
4.2.3	Related work and further directions	141
4.3	Nested graphs	142
4.3.1	Modularity by nesting	143
4.3.1.1	The traditional conceptual graph approach to nesting	143
4.3.1.2	Exploiting the power of guards in the nested setup	146
4.3.2	Nested conceptual graphs	146
4.3.2.1	Syntax	147
4.3.2.2	Nested structures	149
4.3.3	Complexity and Guards	150
4.3.3.1	From nested graphs to simple ones	151
4.3.3.2	Complexity of reasoning in nested graphs	157
4.3.4	Concluding remarks	158
4.3.4.1	Related work	158
4.3.4.2	Further work.	159
5	Conclusions	161
	Bibliography	167
	Index	181
	Samenvatting	183

Acknowledgments

In the first place, I would like to express my deep gratitude to Johan van Benthem, Anne Preller and Frank Veltman. I could not have dreamt having a better team of supervisors. Their enthusiasm for new visual escapades, their sound guidance through the main tracks and their fruitful complementarity offered me the perfect support to carry out this journey through logical landscapes. Anne made me discover the graphical software *Turing's world* in a basic logic course we taught together. I enjoyed her constant demand for mathematical precision in the blurredness of visual issues. Frank warmly welcomed me in Amsterdam and introduced me to various aspects of reasoning. His daily care and his taste for coffee with “character” gave a pleasant touch to my studies. Johan joined the voyage in its second half. His availability, his rigorous analysis, his vast knowledge of the fields involved and his ability of extracting a clear line from the gestalt view of my muddle-headed propositions helped me tremendously to head for interesting destinations.

I am indebted to many people who have contributed in one way or another to make this exploration possible and enjoyable. I would like to particularly thank some of them.

Several colleagues and teachers stimulated the tracks followed in this work. As a *compagnon de route*, Patrice Duroux brought to me a note of Jazz and a valuable friendship through these doctoral years. Besides my supervisors, Dick de Jongh also saw the interest in this *Franco-Hollando-Graphico-Logico-project* and helped to make it happen. During a study year spent in Montreal, Gilles Brassard and Pierre McKenzie opened my eyes to the beauty of computational complexity theory. Michel Chein and Marie-Laure Mugnier let me meet conceptual graphs and rightly insisted on the graphical nature of these languages. In interesting discussions, Jean-François Baget, Olivier Guinaldo, Eric Salvat and Geneviève Simonet gave me the opportunity to deepen my knowledge of the formalism. Marco Aiello, Maria Aloni, Carlos Areces, David Beaver, Paul Dekker, Rosella Gennari, Jelle Gerbrandy, Jeroen Groenendijk, Maarten de Rijke, Robert

van Rooy, Martin Stokhof and Allard Tamminga were not only efficient ambassadors of Amsterdam's dynamic vision on logic, language and computation; their own dynamism created a very stimulating working environment. Bé Bernini, Ria Beentjes, Peter Blok, Josette Durante, Erik-Jan van der Linden, Ingrid van Loon, Kees Ostendorf, Marjorie Pigge, Marjan Veldhuisen, Marco de Vries and Reni Webb were efficient skippers on the administrative reef. For their relevant feedback at the conclusion of this journey, I would also like to thank the examiners of the graduation committee: Franz Baader, Michel Chein, Jan van Eijck, Marie-Laure Mugnier, Maarten de Rijke and Martin Stokhof.

Anne, Bert, Bridget, *La fondation Antoine de Saint-Exupéry*, Marieke, Mike, Mireille, Nynke, Oscar and Rienk deserve my thanks for their contribution to the text and pictures in this thesis.

Merci à Edouard and Steffi, for their happiness and for kindly accepting to be my paranimfs; Raphaël, for his forthcoming novel; Bernard, for his friendship; Vince, for his *Breton-Catalan* perspective on life; my badminton-mates for all the smashed shuttles used as a stress-safety-valve; my *Bà Ngoai*, Bert, Diana, Hervé, Jaap, Jorge, Nynke, Stephan, Sytske and Valérie, for being a wonderful family. My parents' encouragement and support mean very much to me.

I dedicate this book to Nina and Marieke. My doctoral experience would not have been as exciting without their love and extraordinary patience.

's Gravenhage
Oktober, 2001.

Gwen Kerdiles

Abstract

Pictorial languages occur in almost every field from road signs to technical design or abstract art. Computer science is no exception. Understanding the reasons for the success of visual information in human communication and exploiting them in an automated fashion has gained a prominent place in the artificial intelligence agenda. By considering several aspects of graphical languages in knowledge representation, this thesis positions conceptual graphs, a specific diagrammatic framework, at a crossroad of logic, language and computation.

Some of the cognitive and linguistic efficient features of drawings play an indisputable role in human and human-machine communication. Besides these interesting representational standpoints, the computational efficiency of reasoning we obtain on some classes of diagrams emphasises the relevance of pictures in automated reasoning.

In this dissertation, computational complexity is understood in traditional symbolic terms. As a result, this lays a common ground for a beneficial interaction between usual textual logics and graphical languages: in the first place, the diagrammatic systems we study reveal the attractive computational complexity of logical fragments that fall outside the usual paths of symbolic logic. Conversely, some symbolic characterisations adapt well to the diagrammatic frameworks. For instance, the notion of guards, which arose from the translation of modal logics into classical ones, defines a new visual notion of tree in the conceptual graph paradigm. Moreover, reasoning techniques can be exchanged between both sides or combined. Finally, cognitive aspects that are recognised in the perception and manipulation of diagrams offer new tracks for expanding established symbolic computational models with additional visual features.

The central issue of this thesis is to explore these interactions between conceptual graph fragments and symbolic logics, in the light of standard symbolic complexity models. The main results that are presented concern graphical proof methods for consequence problems and their complexity analysis in several conceptual graph languages. Furthermore, by bringing the study into the wider

perspective of visual information in artificial intelligence, we aim at contributing to the general issue of a better understanding of some properties of reasoning with diagrams; this appears to be the necessary basis for further promising connections between symbolic and graphical perspectives.

The work is organised in five chapters. The first two chapters position conceptual graphs in the perspective of several disciplines involved in artificial intelligence. Chapter 1 relates conceptual graphs to historical appearances of diagrams in logic, pictorial languages in knowledge representation, cognitive studies of visual information and drawings used in natural language processing. The wide scope of this overview stresses the relevance of fine-grained studies of visual properties to the artificial intelligence community as a whole. Computational logic may be seen as common ground for all these fields when applied to automated reasoning; this is the subject of the next chapter.

Chapter 2 presents the technical framework in which the graphical systems used in the rest of this work will be evaluated. Symbolic complexity analysis offers fine-structure formal analysis of reasoning with the graphs and connects the study of visual reasoning to current interests in expressiveness and complexity in symbolic logic. A geography of complexity results in classical and modal fragments is then depicted. It sets the scene for the study of conceptual graph languages: several decision problems are relevant and homomorphism-based methods rely on problem equivalence (between model comparison and consequence) that occur in low-expressive languages.

Chapter 3 introduces the core fragment of simple conceptual graphs and projection, a consequence calculus based on labelled graph homomorphism. In addition to the usual semantics of simple graphs, which is given by a translation to existential conjunctive FOL, a model-theoretic approach is also provided. It offers a direct handle for associating projections with model comparisons. By defining a notion of meta-acyclicity based on guarded quantification and an appropriate projection algorithm, a tractable guarded fragment of simple graphs is highlighted (Theorem 3.3.7). It includes all previously known tractable fragments of simple conceptual graphs (i.e. graphs that can be transformed into equivalent trees).

Chapter 4 explores different possible extensions of the core language. First, the addition of atomic negation is considered. In the graph representations, a separation criterion of positive from negative information defines a fragment of simple graphs with atomic negation in which projections apply (Theorem 4.1.19). Furthermore, in the guarded restriction of this fragment, consequence is polynomial (Corollary 4.1.22). Secondly, for a language of conceptual graphs equivalent to first-order logic, we propose a complete proof method combining tableau construction rules and projections (Chapter 4.2). Finally, in the remaining part of the chapter, a modal perspective for graph nesting is studied. Reimporting the notion of guards in this modal framework enables us to define a language of nested graphs with a tractable associated projection (Corollary 4.3.15).

In the last chapter, we draw our main conclusions from the complexity results obtained along our chosen route through conceptual graph landscapes. In particular, the successful interaction of graphical aspects with symbolic ones suggests promising further paths towards more visually oriented computation.

Introduction

Mankind has used pictorial representations to convey information since the first prehistoric wall paintings. The resemblance of pictures to what they represent and their universal nature are often put forward as reasons for the efficiency of pictures as a mean of communication. In addition, one can identify graphical features that play a role in the efficiency of reasoning with pictures. The two aspects, representation efficiency and inference efficiency, are complementary in this study of a particular class of graphical languages, conceptual graphs. The aim of this dissertation is to examine some cognitive and computational impacts of representing knowledge with conceptual graph diagrams.

The course of this dissertation is almost linear. The first chapter offers a wide perspective in exploring several facets of the representation of information by conceptual graphs in the light of logic, artificial intelligence, cognitive science and linguistics.

The second chapter introduces the most salient theme of this dissertation, the complexity of logical reasoning in conceptual graph systems. Symbolic complexity theory offers the required fine structure to define the formal problems relevant to the study and to explore the connections between the diagrams and symbolic logics.

The third chapter explores in detail the cornerstone fragment: simple conceptual graphs. A particular interest concerns the complexity of an homomorphism-based calculus that takes advantage of guarded quantification in the diagrams.

In the fourth chapter, the addition of different forms of negation to the simple graphs and the modal nature of nested graphs are investigated. The tractability of reasoning in these fragments and the definition of graphical complete calculi remain our main concerns.

In the final chapter, the main conclusions are drawn from the lessons learned along this systematic investigation.

Chapter 1

Diagrams and visual information

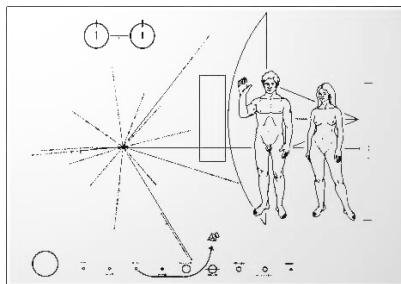


Figure 1.1: Message sent on the Voyager space probe

Diagrams occur in almost every domain where information is communicated. Examples are numerous, from the geometrical demonstration of Pythagoras theorem to a diagrammatic message sent to possible extra-terrestrial life forms. Part of the success of diagrams as a means of communication is due to the simple way in which complex information is represented. This chapter examines several aspects of the diagrammatic representation of knowledge, with a constant focus on a particular point of interest, conceptual graph diagrams.

To begin with, the occurrence of diagrammatic representations in the history of logic is explored. Peirce's predicate logic of existential graphs was introduced at the turn of the nineteenth century. It is a particularly important source of inspiration for the development of conceptual graphs.

In computer science, graphical features are extensively exploited on the representational level as well as on the computational level. Some of these applications to artificial intelligence are examined in part two of this chapter.

The essence of diagram processing resides in their prime perceptual effect. In the third part, the cognitive impact of conceptual graph diagrams is explored.

As a preponderant form of communication between humans, natural language should not be disregarded. The last part of this chapter is an attempt to relate

the structure of conceptual graph drawings to the structure of discourse in natural language.

Diagrams? Before diagrams are explored, a brief presentation of some terms, that all correspond to graphical representations, is necessary.

Picture will be used as a generic term to refer to a graphical representation laid on a delimited zone of a two-dimensional space. *Images* and *drawings* are both pictures. Conventionally, images refer to pictures that can be decomposed into a finite amount of minimal points (e.g., pixels, bitmaps), whereas drawings can be formed of continuous lines. With the assistance of computers for drawing pictures this difference between images and drawings is even more subtle: while a computer picture can be conceived and stored as a drawing (e.g. a vectorial representation), its printing on a screen will be an image limited by the resolution of the screen.

Diagrams correspond to schematised drawings in which graphical constituents are associated with a well-defined semantics. Finally, a *graph* refers to an abstract mathematical object composed of nodes connected by edges. It can be physically represented by a diagram.

The specific nature of diagrams will be discussed in the part dedicated to the cognitive impact of conceptual graph diagrams (Chapter 1.3). Before that, some graphical systems that preceded conceptual graphs in logic are presented.

1.1 Diagrams in logic

Graphical knowledge representation systems are not a new phenomenon. Eighteenth century Euler circles and nineteenth century Venn diagrams are still popular for manipulating sets and boolean operations. Although most languages of modern logic are textual, it is worth noting that the pioneer research for the foundation of predicate logic was presented in graphical forms: at the end of the nineteenth century, Frege and Peirce independently introduced two graphical systems of first-order logic in an attempt to formalise mathematical reasoning.


1.1.1 Frege's graphs

The language proposed by Frege in his *Begriffsschrift* [Fre79] represents sentences by trees derived from four graphical primitives:

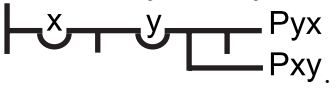
1. "assert A": $\vdash A$

2. "not A": $\neg A$

3. "B implies A": $\supset A$
B

4. “for every x , Px ”: 

For instance, “assert that for every x , there exists y such that Pxy and Pyx ” or

equivalently, “ $\forall x \neg \forall y (Pxy \rightarrow \neg Pyx)$ ” is represented by 

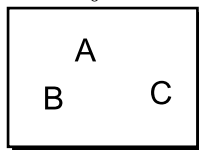
Similar to a tree presentation of a tableau calculus proof, different parts of the representation are distinguished by a disposition on branches. In this particular language, the premiss and the conclusion of an implication occur on different branches. Despite this graphical feature, a representation reads in a linear fashion that clearly resembles its textual counterpart: from left to right and in a depth-first way such that at a branching point, the lowest path –i.e. the premiss of an implication– is first explored.

For predicate logic, Frege’s graphical language has long since been replaced by Peano’s textual notation. Nevertheless, the importance of visual information to computers and robots has brought back another logical system of this period to the research agenda, Peirce’s graphs.

1.1.2 Peirce’s existential graphs

Peirce’s languages and calculi have been studied extensively; see e.g., [Pei58], [Rob73], [Thi75] [Shi93] or [Ham98]. It is not the aim to describe these logical systems in detail, but to point out some features of Peirce’s existential graphs that have been adopted in conceptual graphs.

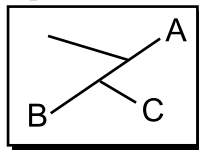
A first feature of Peirce’s graphs, that is fundamental to conceptual graphs, is the role of a primary surface. The sheet of assertion fixes the bounds of the space on which the representations of the different pieces of information that are asserted are disposed. Furthermore, the two dimensionality of the plane is used to represent the conjunction of all drawn components.



For instance,  represents the conjunction of A , B and C .

The symmetry of conjunction is induced by the fact that there is no predefined order of the conjuncts, as opposed to a textual formula read from left to right.

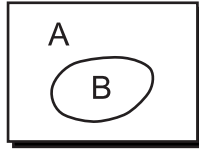
In Peirce’s graphs, existentially quantified variables are represented as lines connecting the predicate occurrences of which they are arguments.



For instance,  is equivalent to $\exists x(Ax \wedge Bx \wedge Cx)$.

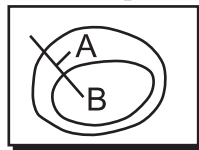
Direct connections through edges will similarly be exploited by conceptual graphs to represent the relationship between predicate occurrences and their arguments.

Finally, negations are represented as closed lines cutting off the negated part from the rest of the assertion.



For instance,  is equivalent to $A \wedge \neg B$.

The place where existential quantification occurs is defined by the outermost zone in which a line that represents the quantified variable in question appears.



For example,  represents $\exists x(Ax \rightarrow Bx)$.

The interaction between existential quantification and negation will be elaborated upon in Chapter 1.4, where some structures occurring in discourse are highlighted in conceptual graph representations.

Peirce [Pei58] proposed some calculi for propositional (alpha system) and predicate (beta system) logic and ideas of a modal framework (gamma systems). In Peirce's systems, a conclusion graph follows from a premiss one if and only if the later can be transformed into the former using an appropriate set of graph transformation rules. Although interesting in themselves, these calculi are not particularly adapted to automatised reasoning. Indeed, they are *not analytical* in the sense that they do not systematically decompose a problem into subproblems, but rest on non-guided rules such as “any graph may be added into a zone enclosed in an odd number of negation lines”. In the light of automated theorem proving, analytical calculi based on graph homomorphisms and analytic tableaux will be studied in this thesis.

1.1.3 Conceptual graphs

Since the late sixties, a graphical knowledge representation formalism equivalent to first-order logic has been developed: conceptual graphs; see e.g., [Sow84, Sow99] for detailed expositions of Sowa's original systems. The syntax and layout were influenced by a combination of Peirce's graphs, linguistic dependency graphs and computer science flow charts. On the semantic and deductive side, order-sorted predicate logic and Peirce's calculi were adopted.

1.1.3.1 Positive information

Departing from the whole first-order language, a sub-formalism for representing positive existential-conjunctive information, simple conceptual graphs, has been carefully studied since Sowa's book [Sow84]. The language is expressive enough to describe factual information with a slight touch of indeterminacy provided by existential quantification. We may distinguish two graphical aspects related to

the fragment: the representation by graph diagrams and a proof method based on labelled graph homomorphism.

Representation Textual symbols of the vocabulary for a conceptual graph language are partially ordered in a predefined classification, called a support in [CM92] or canon in [Sow84].

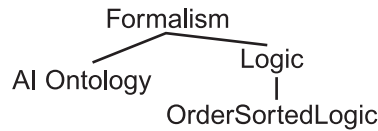


Figure 1.2: A support

For instance, the tree in Figure 1.2 represents the information that “every order-sorted logic is a logic and that every logic or AI ontology is a formalism” or in FOL notation:

$$\begin{aligned} \Phi_0 = & \forall x[OrderSortedLogic(x) \rightarrow Logic(x)] \\ & \wedge \forall x[Logic(x) \rightarrow Formalism(x)] \\ & \wedge \forall x[AIOntology(x) \rightarrow Formalism(x)] \end{aligned}$$

Simple conceptual graphs are bipartite node-edge diagrams, in which square nodes, representing term occurrences, alternate with rounded nodes, representing predicate occurrences. Labelled edges linking a round node (or relation node) to a set of square nodes (or concept nodes) symbolise the ordered relationship between a predicate occurrence and its arguments. Concept nodes are labelled with a concept type and either a constant or a star (standing for an unnamed existentially quantified variable).

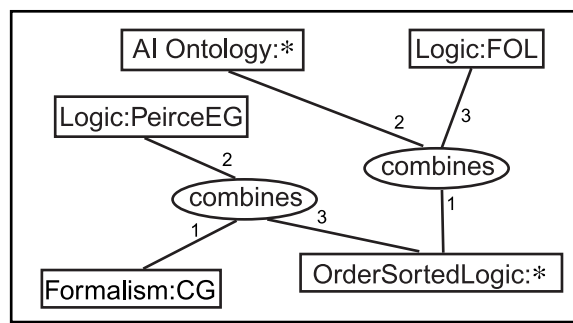


Figure 1.3: A simple conceptual graph diagram

For instance, the graph in Figure 1.3 is a representation of “The CG formalism combines Peirce’s EG logic to an order-sorted logic, which itself combines an AI

ontology to the FOL logic” or the (positive existentially quantified) FOL formula:

$$\begin{aligned} \Phi_1 = \exists x[& \text{Formalism}(CG) \wedge \text{Logic}(EG) \wedge \text{OrderSortedLogic}(x) \\ & \wedge \text{combines}(CG, EG, x) \\ & \wedge \exists y[\text{AIOntology}(y) \wedge \text{Logic}(FOL) \wedge \text{combines}(x, y, FOL)] \end{aligned}$$

Computation Consequence proofs in the simple conceptual graph formalism correspond to labelled graph homomorphisms, called projections (e.g., [CM92]).

The possibility of basing deduction on graph operations has strengthened interest in this alternative to classical calculi of predicate logic.

As in order-sorted logics [SW90], the classification of concepts and relations is exploited in logical consequence. For instance, given the information that “every order-sorted logic is a logic”, represented in the support in Figure 1.2, the information that “the CG formalism combines two (not necessarily different) logics” can be derived from the graph in Figure 1.3 or in FOL notation:

$$\Phi_0 \wedge \Phi_1 \vdash \exists x \exists y [\text{formalism}(CG) \wedge \text{logic}(x) \wedge \text{logic}(y) \wedge \text{combines}(CG, x, y)]$$

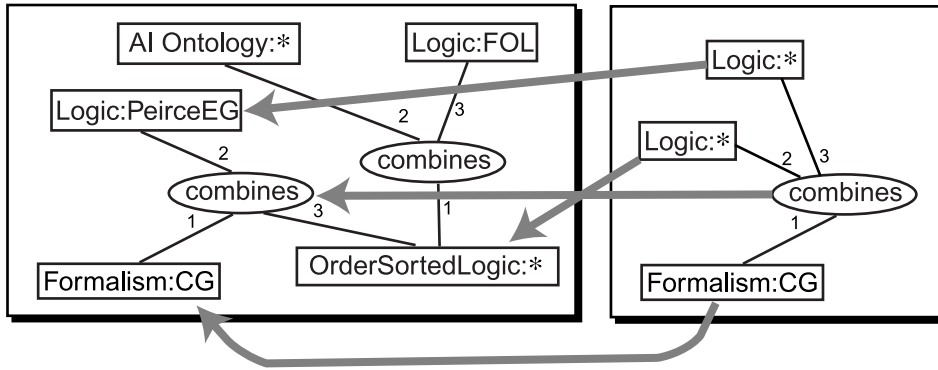


Figure 1.4: A projection from a simple conceptual graph to another one

A proof of this logical consequence is provided by a mapping, pictured in Figure 1.4, preserving both the structure of the source graph (i.e., the conclusion of the logical consequence) and the ordering of labels conveyed by the underlying support.

In subsequent chapters, the computational efficiency of this calculus will be explored for different structural fragments of simple conceptual graphs and extensions to negations and modalities.

1.1.3.2 Negation

For a full predicate logic language, Peirce’s closed negation lines are used to enclose negated zones. For instance, the graph in Figure 1.5 is a representation

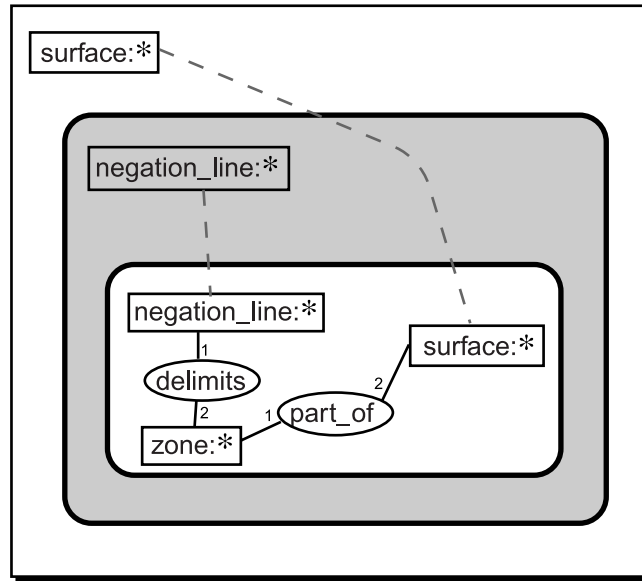


Figure 1.5: Negated regions in a conceptual graph

of “there is a surface such that every negation line delimits a zone which is part of that surface” or

$$\exists x[surface(x) \wedge \neg(\exists y[negationline(y) \wedge \neg(\exists z[zone(z) \wedge delimits(y, z) \wedge partof(z, x)])]]]$$

In Chapter 4, we will explore some possibilities and limitations of adapting the projection calculus to the representation of negation in conceptual graphs. In particular, an interlacing of projections and semantic tableaux will be proposed as a predicate logic calculus.

1.1.3.3 Nested conceptual graphs

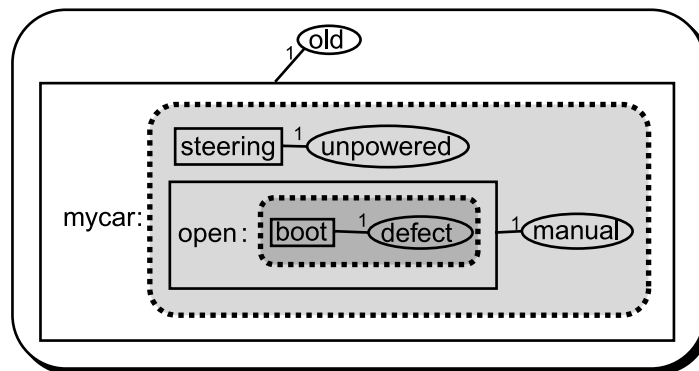


Figure 1.6: A nested conceptual graph

An additional structural level is obtained by nesting a description (that is itself a nested graph) in concept nodes. To set the ground of the recurrence, the empty

graph that corresponds to the logical constant *True*, is considered as a nested graph (in order not to overload the picture, empty descriptions of concept nodes in Figure 1.6 have not been represented). The nested conceptual graph formalism, which has a modal flavour, can be exploited to distinguish different groups of localised pieces of information or different levels of knowledge. A “zooming in effect” enables to focus on one local description.

For instance, the nested diagram in Figure 1.6 illustrates a boot failure occurring in the context of the open of my car.

The study of several semantics that can be associated to these nested drawings will be the subject of Chapter 4.3.

1.1.4 Concluding remarks

As graph theory is an extensively studied field in computer science, it is not surprising that many other logical formalisms have chosen graphical features.

Kripke models of modal logics are often represented as labelled graphs and model comparisons, such as bisimulations (e.g., [Ben96]), are naturally defined in terms of graph homomorphisms. Applied modal logics, such as attribute value logics (e.g., [Spa93]) or feature logics (e.g., [Rou97]) exploit the tree structures of their frame languages.

Research by the team of Barwise and Etchemendy at the Visual Inference Laboratory¹ has concentrated on the process of learning logical reasoning by graphical model construction (Hyperproof project² and the pieces of software Turing’s world and Tarski’s world) and on the formalisation of the graphical systems of Euler, Venn and Peirce; see, e.g., [Shi93, Shi95, Ham95, Ham98] and [BE98] for a collection of articles on different aspects of learning and practicing diagrammatic logic.

This introduction to conceptual graphs has exemplified the fact that there is not a single conceptual graph formalism, but a multitude of possible ways to combine and interpret a chosen group of primitive graphical artefacts. Therefore, it is important to identify some criteria, that may guide us in favouring one system over another. In the remainder of this chapter and in the next chapter too, facets of the conceptual graph paradigm are explored under the light of several fields related to logical reasoning, such as artificial intelligence, cognitive science, linguistics and computational logic.

¹<http://www-vil.cs.indiana.edu>

²<http://www-vil.cs.indiana.edu/Projects/hyperproof.html>

1.2 Conceptual graph diagrams and artificial intelligence

Diagrams have gained an indisputable importance in computer science and artificial intelligence (AI). They occur in almost every field related to computers, ranging from the actual chips to the abstract representation of knowledge. For example, circuit designs, data structures, algorithms, human-machine interfaces, inheritance in object programming languages or knowledge bases can be represented as trees, graphs, flow-charts or other specific diagrammatic forms.

If research in logic has long been concerned with the distinction between what is provable or not, the application to AI has somehow shifted the focus to determining what kind of reasoning can feasibly be carried out by a computer in a “reasonable” amount of time. Following this line of thinking, adapted representation languages and deductive systems have been invented for automated reasoning. Semantic networks are one example of this.

1.2.1 Semantic networks

Semantic networks, a family of node-edge graphs in AI, have been popular for trying to represent knowledge in a way that is as close to natural language as possible. The proliferation of graphical systems lacking formal semantics has led to criticism such as McDermott’s “Artificial intelligence meets natural stupidity” [McD76], but also to the development of a family of formal semantic networks originated by Brachman’s KL-ONE system.

Besides the fact that conceptual graph formalisms belong to the class of (formal) semantic networks, they have also borrowed a central notion of classification from artificial intelligence.

- On one hand, the ordering of archived representations of pieces of information, with respect to logical entailment, is relevant to efficient information retrieving from conceptual graph knowledge bases. The logical consequence relationship is sometimes called subsumption and, its symmetrical counterpart, generalisation.
- On the other hand, in AI the classification of the basic terms of a language to describe a particular application domain is called an ontology. Conceptual graph languages exploit such ontologies for efficiency purposes by restraining search spaces to subdomains smaller than the whole domain of a given knowledge base.

1.2.2 Description logics

When artificial intelligence and logic meet, description logics are successful logical formalisms applied to the representation of knowledge. They inherited the two

notions of classification, ontologies and knowledge classification, from semantic network and terminological logic ancestors.

By adopting the semantics of a modal formalism, called hybrid logic (see [Are00] for a detailed analysis of hybrid logics and their relation to description logics), description logics benefit from the efficient computational behaviour of modal logics.

Building on the tree characteristics of models for modal logics, there has been a recent return to graphical features in the syntactic and deductive side of description logics: Baader et al. [BKM99, BMT99] propose a translation of some description logics into a language of trees that is exploited in homomorphism calculi. We shall return to description logics with the complexity study of logical reasoning in Chapters 2 and 3 and with the modal direction taken for nested conceptual graphs in Chapter 4.3.

1.2.3 Contexts in AI

Many researches in AI have questioned the context dependency of information. Giunchiglia and Bouquet[GB97] metaphorically present a context in AI as “a sort of box which is part of the structure of an individual’s representation of the world and which draws a sort of boundary between what is *in* and what is *out*”. In J. McCarthy’s pioneering work on the formalisation of context (see e.g. [McC87] and [MB97] for a recent survey), such a box is a *rich object* (a collection of parameters) upon which a representation depends. Typically, a representation can be true in some contexts and false in others. For instance, the piece of information “It is raining” calls for a context of utterance to be interpreted and that context can include among the parameters the time and place of utterance (In the context of Amsterdam, that sentence is often true and particularly on Sunday April 4, 1999). A context, as part of the cognitive state of an agent (the hearer), is used in the interpretation process.

The box metaphor resembles the two kinds of closed lines of the conceptual graph syntax: negation lines and nested boxes. Indeed, from a linguistic point of view, negations play the role of a border line for anaphoric bindings by surrounding a context of discourse interpretation and being permeable in specific conditions. We will elaborate this linguistic argument by examining some structural properties of discourse in Chapter 1.4.

For nested graphs, the meaning of enclosing information into a box can be captured by adapting an applied modal logic: the context logic of Buvac [Buv98]. This point of view will be undertaken in the study of nested conceptual graphs (Chapter 4.3).

1.2.4 Conclusion

Artificial intelligence is at the crossroad of logic, linguistics, computer and cognitive sciences. Therefore, it is almost impossible to avoid such pluridisciplinary references. Conceptual graphs also dwell at this multicultural crossroads.

Returning to ontologies and without disputing terms, Peirce's graphs, semantic networks and a fortiori, conceptual graphs, make an ontological commitment to graphical items. We now turn to the cognitive impact of these primitive graphical artefacts that, when combined, form conceptual graphs.

1.3 The visual impact of CG drawings

Drawings have many visual properties. Three properties that are particularly pertinent to this study of knowledge representation by conceptual graphs have been chosen to be elaborated upon.

The gestalt feature of diagrams, their faculty to provide an overview of what is represented, is the first visual subject. The perception of the global shape of the information represented results from the different uses of the two dimensionality of drawings. In particular, we distinguish the spatial disposition of pieces of information and the agglomeration of lines to form skeletal structures on which some components hang.

The second visual feature of drawings, that will be discussed in Chapter 1.3.2, is their faithfulness to what they represent. Drawings are often easy to grasp because they are somehow close to what they depict. This property is linked to the expressive power of the drawings, which is relatively limited compared to the high level of abstraction conveyed in sentences of classical linear textual logic languages.

Finally, Chapter 1.3.3 examines how some drawings can provide additional information to the semantic conventions.



Figure 1.7: The evolution of stock-quotes over time

For example, these three themes appear in a kind of diagram that is com-

monly found in the economic pages of newspapers; stock-charts. Figure 1.7 is a space economical presentation of a large matrix of numbers (i.e., 859 bidimensional coordinates). The diagram stresses the overall characteristics of the data, such as a price that globally follows a downward slope over a three month period. Moreover, the use of conventional scales for price and time facilitates our understanding of the chart. Finally, the intersections of curves are typical pieces of information that are not part of the initial data, but are directly read on the diagram and can be interpreted by investors as signals for changes of tendencies.

We now undertake our first subject in visual matters, the perception of a global perspective of diagrammatic information.

1.3.1 Overview of the information drawn

A generally acknowledged property of diagrams is that they offer a synoptic representation. The possibility of visualising the global structure of a large set of data, takes advantage of our prime perception of visual notions such as density or direction. In particular, the global information perceived in a conceptual graph drawing is a sort of large scale map of the represented relational network. This map has two main components: a partition of the space into areas and a skeletal structure.

1.3.1.1 Partition of the space.

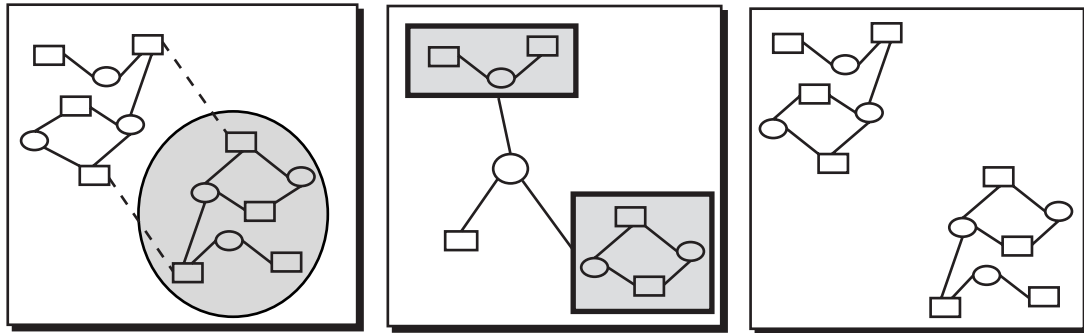


Figure 1.8: Closed lines and empty-spaces dividing the sheet of assertion

How is the spatial division of the plane on which a conceptual graph is drawn perceived? Outlines of the areas must be found. The most effective symbol to represent a borderline between zones is to draw a line. For example, a Peirce's cut, the representations of a negation in conceptual graphs, is a closed line imprisoning pieces of information into a negated area. Nested boxes in nested conceptual graphs also divide the plane of a drawing into areas symbolising different levels of information. An additional way of defining areas is provided by the perceptual effect of density. In particular, emptiness appears as a discriminating feature

between zones of high-density. To summarise, the first overall impression of a conceptual graph is some partition of the space into zones containing pieces of information.

1.3.1.2 Spinal structure.

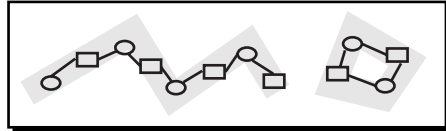


Figure 1.9: Spinal structures

Just as important for the overview is the impact of edges that are perceived as agglomerated into a spine linking different pieces of information. This skeleton does not necessarily have a beginning or end; it is merely central to the different components. The global structure of such a network provides some assistance for navigating the drawing, for moving our point of focus along a path or jumping to an information island. This idea of a support for navigation is reinforced in nested conceptual graphs because they represent different levels of relational structures in one picture: like a road map that includes enlargements for cities provides a representation of a road network at the top level and of some street networks at a lower level. A nested conceptual graph drawing stimulates our visual faculty for discriminating levels and grouping what is connected, in order to safely convey an understandable picture of a complex multi-level network.

To recapitulate, the ingenious human visual machinery capitalises on the perception of density, groups, discontinuities and line structures, to extract, at a glance, the overall information conveyed by drawings. This information can further be employed to guide a search for more details. The overview is a large-scale guide for further in-depth observations. Nevertheless, efficiently using it may require the same kind of training as the reading of a road map does.

Structures formed by lines on the drawings are perceived, but what makes us recognise a shape in a drawing? This is the subject of the next section.

1.3.2 Faithfulness of drawings

Graphics are often labelled as efficient information conveyors because of the facility to understand them. They somehow mirror the information represented. In the case of conceptual graph drawings, two factors influencing this resemblance property can be distinguished: (i) the use of graphical basic components that have a conventional meaning closely related to what they represent and (ii) a deliberately limited level of abstraction.

1.3.2.1 Simple graphical components.

Basic graphical items in conceptual graph drawings are nodes, edges (lines connecting nodes) and closed lines defining frontiers. Different kinds of frontiers can be distinguished by the chosen shape conventions, such as thickness. For instance, lines surrounding negated zones and those defining the outlines of modal worlds are drawn differently.

1.3.2.2 Limited abstraction.

It has been argued that the graphical signs have a standard simple semantics, but what makes a conceptual graph diagram easy to grasp, also lies in limited expressive power of the drawings.

First, there are very few graphical signs used and they all have a clear significance. This fact implies the need for only few simple rules of interpretation, which is certainly to the advantage of the reader.

A second factor of simplicity is the direct nature of the graphical message: what is left unsaid is really not represented. The sole exception to this rule is the use of the indefinite marker *, a place holder for an indefinite object. It corresponds to an existentially quantified variable in a textual logic language. Nevertheless, other connectives commonly used in logic, such as universal quantification, disjunction and implication, are left out of the picture. These connectives have the disadvantage of summarising complex information into single symbols. For example, universal quantification conveys the message that all the individuals living in the represented model have some property. In other words, it abstracts some information to the level of the whole population instead of directly showing facts for each individual. It conveys a large amount of information with very few syntactic items and the expansion of the compacted information is left to the reader. Similarly, disjunctions and implications call upon the reader's interpretation process to build several alternative models at once.

Existential quantification alone does not have these drawbacks. It provides the reader with an unnamed object, but guides the interpretation process by showing a one-on-one correspondence between the syntactic objects and the represented ones. Hammer [Ham95] has studied a similar type of matching for different forms of diagrams; *the isomorphism thesis*. [AB96] and [CSO94] discuss how the low level of abstraction in some graphical representations of logical sentences can influence a logic learning process.

To summarise this point on the faithfulness of conceptual graph drawings to what they represent, we can relate this advantage to the small amount of graphical signs used, and to their intuitive meaning. A drawing with a relatively low level of abstraction presents a one-on-one correspondence to the represented.

Until this point, it has been argued that drawings have prominent perceptual features. Some of them, like resemblance, have an obvious semantic use. Others,

like the overview spinal structure, have a less direct meaning. The study of the semantics of these graphical properties that provide extra information, is the subject of the next section.

1.3.3 Additional information

Graphical features can be perceived. Some of them are given meaning according to defined interpretation rules. For instance, an edge between two nodes is known to represent a binary relationship between two objects. Other features fall outside the interpretation conventions, but are nevertheless informative if associated with a meaning.

This section is devoted to the study of the additional information that is perceived from conceptual graph diagrams. Basic semantic conventions give rise to new interpretation rules. Two themes may be distinguished. The first concerns some generalisation of a particular convention to a larger domain. The second is related to the modification of a particular convention by some typically graphical feature, which initially had an obvious intuitive meaning.

1.3.3.1 Generalisation of semantic conventions.

In the previous section, conceptual graph drawings were observed to be composed of a small amount of distinct graphical signs (nodes, edges and closed lines). It has been argued that the small amount of signs is a cognitive strength, as only very few interpretation conventions are required to understand a drawing.

Among these graphical signs, the edge has a preponderant role, one of representing relational information. An edge is a local object. It connects its extremities at a particular place in the representation. However, this role of representing *direct connectedness* is intuitively generalised to the level of the whole representation. Agglomeration or concatenation of edges convey a global notion of *indirect connectedness*.

This is first visible in the graphical representation of the vocabulary classification. Edges correspond to implications and paths provide their transitive closure (e.g., from reading the branch on the right-hand side in Figure 1.2, we can conclude that any order-sorted logic is a formalism).

Similarly, in conceptual graph diagrams, the notion of indirect connectedness has a meaning of relatedness. Relatedness is useful in applications like information retrieval, enabling the connection of objects that are not in direct relation to each other. Interpreting distinct connected compounds as unrelated pieces of information provides a simple guide for breaking down a problem into smaller sub-problems that can be solved independently of each other. Salvat's experiments [Sal97] in an application of a meta-resolution rule for a language of conceptual graphs, which includes implication, have shown that applying this obvious selection function (i.e., *if possible, take a successor in the connected compound at*

stake) does often reduce the number of backtracks. Tree structures will also prove essential to efficient calculi.

To summarise, the meaning of edges, as being representations of connectedness, can be generalised in a weaker significance for paths: one of relatedness.

1.3.3.2 Interaction of graphical effects and semantic conventions.

From the basic interpretation convention, the meaning of the occurrence of two distinct pieces of information on the same plane (or area in the presence of Peirce's cuts) is known: the conjunction of the components is represented. However, this significance can be strengthened by a perceptual effect, namely density. Indeed, spatial grouping of pieces of information can corrupt the neutral conjunctive information and represent a second form of relatedness. As noted above, density capitalises on innate human perception to make salient information relevant. This second notion of relatedness can prove useful in order to organise the presentation of information in packets. These are groups either conveying a semantic message or simply being a practical help (for example, a division of the space between multiple users of a knowledge base).

Another use of density occurs in homomorphism proof drawings. The significance of a proof diagram can be enriched by information about the location of the pieces of information that are utilised on the density map.

We have seen that information that is not considered in the basic semantic conventions can be perceived from graphics. The meaning of this information is intuitive because it results either from the generalisation of the semantics of local items to a larger scale, or from the interaction of meaningful graphical effects with basic semantic rules.

In his thesis [Shi95], Shimojima studies a related phenomenon: free-rides. Free-rides are additional information resulting from the matching of graphical constraints with some constraints of the represented. The derived meaning postulate, read from the transitive closure on support paths, would fit this definition, but free-rides are more specific. The additional information can be directly interpreted using the basic semantic rules of the graphical system. The phenomena examined in this section are of a slightly different nature. They concern extra information which is obtained by derived interpretation rules.

1.3.4 Conclusion

In this survey of some visual properties of conceptual graph drawings, which by no means claims to be exhaustive, three main themes that participate in the cognitive efficiency of diagrams have been distinguished.

The first issue concerns the gestalt feature of diagrams and their faculty of offering a synoptic representation of both the partition of the information space and the spinal structure linking pieces of information.

The second theme is an attempt to recognise the features that make conceptual graph drawings faithful representations of relational structures. The nearly iconic nature of the graphic components and the limited abstraction represented in drawings have been identified as reasons for this mirroring property.

Finally, the usefulness of perceptible additional information is linked to the intuitive adaptation of local semantic conventions to large-scale graphical effects.

By using expressions such as *easy perception*, the ingenuity of the human visual machinery is taken for granted. However, it is far from clear how complex basic perception operations would function for an artificial visual machine. Despite the lack of formal visual models for efficiency measures, we are not totally clueless. In the forthcoming chapters, the use of classical complexity theory for textual translations of graphs will provide a first handle in a formal attempt to answer the question.

Closer to the previous cognitive concerns than computation models, the study of relationships between natural language and conceptual graphs is the next subject focused on.

1.4 Conceptual graphs and the structure of discourse

Natural language is the pervasive medium for cognitive activities. Despite the fact that it is transcribed in a linear way with the use of symbols (letters or characters), a discourse shares some characteristics with conceptual graph diagrams.

Primarily, a discourse is structured. First, we will examine some correspondences between the binding of term occurrences in the process of conceptual graph construction and anaphoric phenomena in natural language.

Some linguistic theories also use pictures. In a second section, the features that bring a specific linguistic formalism, discourse representation theory, and conceptual graphs closer will be considered.

Finally, by viewing conceptual graphs and discourse representation structures together, the same innovations may be applied to both. As an illustration, so-called dynamic interpretations will be quickly discussed.

1.4.1 Bindings in conceptual graphs and discourse

It has been observed that the linguistic counterparts of logical connectives behave as structuring items in discourse, with different permeability properties to pronominal coreferences.

For example, the conjunction of two sentences can be expressed in English by the use of the term ‘and’ or just the concatenation of these sentences: “A man entered. He took a chair.” or “A man came in and he took a chair.”. In the second sentence, a pronominal reference to an object introduced in the first

sentence is possible. On the other hand, the use of negation appears to block the possibility of such binding: It seems unacceptable to continue the sentence “It is not the case that a man came in.” with “He took a chair.” as the pronoun ‘he’ cannot be resolved by any object previously introduced in this piece of discourse. Conceptual graph construction rules present similar properties of bindings.

1.4.1.1 Conjunction of conceptual graphs

Conjunction in conceptual graphs obeys two simple rules:

- (i) the conjunction of two pieces of information is represented by their juxtaposition on the sheet of assertion
- (ii) in the absence of negation line, a concept node can be made coreferent to another concept node occurring in the same graph.

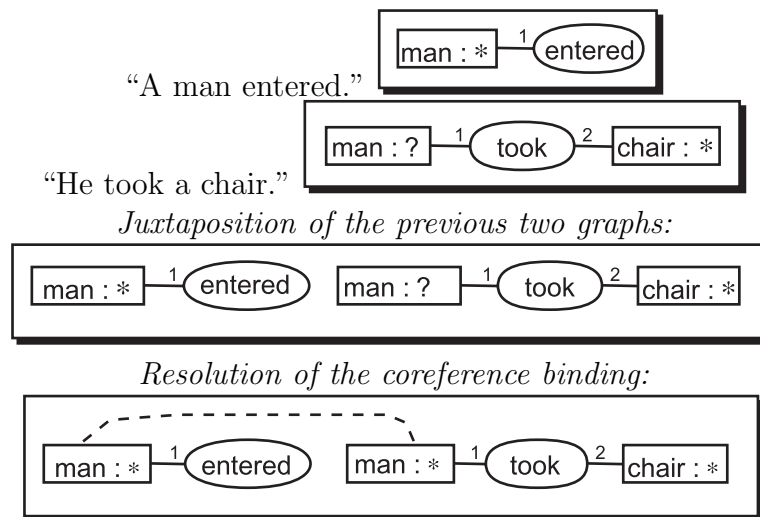


Figure 1.10: Conjunction of conceptual graphs

For example, in Figure 1.10, the pronoun ‘he’ is represented by a concept node labelled with the marker ‘?’ symbolising that it needs to be made coreferent to another accessible concept node. After the two graphs have been juxtaposed, the concept node ‘man:?’ becomes available for coreference to the node labelled with the question mark.

The resolution of the anaphoric binding is a problem that is beyond the scope of this thesis. What is important is the fact that, after juxtaposition of the two initial graphs, the representation of the indefinite noun phrase ‘a man’ becomes available for coreference to the representation of the pronoun ‘he’.

According to the second rule, in the absence of negation, the application of coreference is, in principle, free for any pair of concept nodes in a graph. Of course, one can add some additional constraints. For instance, it could be requested that two nodes made coreferent should have concept types sharing a common subtype.

It could also be forbidden to link two nodes labelled with different constants, respecting a common assumption for many AI systems that different constants represent different individuals.

If conjunction in conceptual graphs is, as conjunction is in discourse, permeable to coreferences, what about negation?

1.4.1.2 Negation

Closed lines, representing negations in conceptual graphs, delimit zones that are included in the outermost zone: the sheet of assertion. These zones and frontiers remind the metaphoric image of “context as a sort of box” discussed in Chapter 1.2.3.

By construction, negation lines do not intersect each other. Thus, the nesting of zones has the structure of a tree whose root is the sheet of assertion. This partial order is called domination. A zone is said to dominate another zone if the later is included in the first one. We may now restate the rule for coreference as follows:

(ii') a concept node can be made coreferent to another concept node occurring in a dominating zone of the same graph.

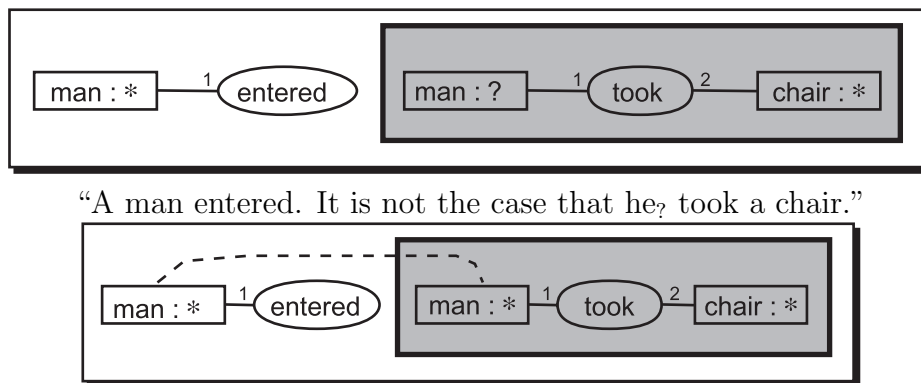
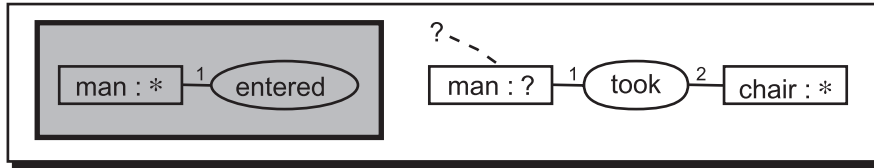


Figure 1.11: Negation boxes are permeable for coreferences from outside-in

Figure 1.10 is an example of concept nodes occurring in the same zone (the sheet of assertion). Let us consider an example with a negation: “A man entered. It is not the case that he took a chair.”. In Figure 1.11, the concept node representing the pronoun ‘he’ occurs in a zone dominated by the sheet of assertion, in which the concept node for ‘a man’ occurs. According to the rule (ii’), we are allowed to bridge those two nodes with a coreference link.

Conversely, the binding of the pronoun ‘he’ is not resolvable in the (unacceptable) discourse “It is not the case that a man came in. He took a chair.”.



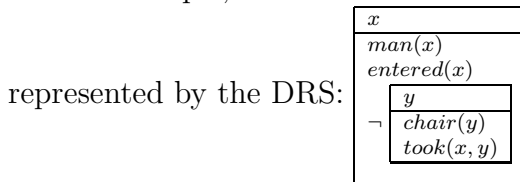
In other words, the scope of existential quantification in conceptual graphs is limited to all the zones that are included in the zone where the quantifier occurs. Such scoping rules are familiar to a linguistic theory that also makes use of pictures: discourse representation theory.

1.4.2 Discourse representation theory

Discourse representation structures (DRS) combine nested boxes with notations of predicate logic for representing the structure of natural language sentences. We refer the reader to Kamp's foundation article [Kam81] and to the extended treatment of DRT in [KR93]. [BB98] is a comprehensive introduction to DRT and some background in computational linguistics.

In a language where all connectives are expressed in terms of conjunction, negation and existential quantification, a DRS is a box divided into two parts. These parts are a set of discourse referents and a set of conditions where a condition has either the form of a predicate logic atom or the negation of a DRS. Discourse referents in the first part of a box correspond to existentially quantified variables which are accessible to the conditions in the second part and, by transitivity of nesting, to all conditions occurring deeper in the nesting.

For example, "A man entered. It is not the case that he took a chair." can be



There is a notable difference between conceptual graphs and DRSs. By inheriting Peirce's lines of identity and contrary to DRSs, conceptual graphs provide a notation of equivalent expressive power that is free of variables. This feature is relevant in theorem proving. Indeed, efficient methods for constructing proofs like free-variable tableaux or resolution require pure representations (representations in which a variable is not quantified twice). Hence, a renaming pre-process can be required. By avoiding variable names, CGs are always pure. However, the absence of variables is mostly relevant for an incremental construction of the representations. If the representations of the constituents of a text can be drawn independently, then building a representation of the whole text consists in merging the representations of the constituents.

To illustrate a problem associated to merging, we present an example from [Eij98]. A DRS for "A man entered. A woman entered." can be obtained by

merging the following two DRSs:

$$\begin{array}{|c|} \hline x \\ \hline man(x) \\ \hline entered(x) \\ \hline \end{array} \oplus \begin{array}{|c|} \hline y \\ \hline woman(y) \\ \hline entered(y) \\ \hline \end{array} = \begin{array}{|c|} \hline x \\ y \\ \hline man(x) \\ entered(x) \\ woman(y) \\ entered(y) \\ \hline \end{array}.$$

But merging is not defined in case of a variable clash:

$$\begin{array}{|c|} \hline x \\ \hline man(x) \\ \hline entered(x) \\ \hline \end{array} \oplus \begin{array}{|c|} \hline x \\ \hline woman(x) \\ \hline entered(x) \\ \hline \end{array} = ?$$

In discourse representation theory, the problem of variable clashes is solved by always building the representation of a new sentence in the context of an existing DRS. Another solution consists in first renaming the variables occurring in different DRSs before merging them. Van Eijck [Eij98] proposes another alternative: the replacement of variable names in DRSs by De Bruijn's indices. Conceptual graphs do not make use of variables, so variable clashes cannot occur and two graphs can always be merged by only juxtaposing them. The simplicity of this safe merging operation is an attractive feature of Conceptual Graphs.

Syntactically, DRSs and conceptual graphs share the same structure for displaying the representations and the same scoping rules for existential quantifiers. Furthermore, semantically, both formalisms rely on a notion of embedding of pictures into classical models for predicate logic. These similarities may be exploited to bridge the differences. On one hand, discourse representation theory has achieved numerous results in the study of natural language phenomena and being able to adapt these results would be beneficial to conceptual graph theory. On the other hand, conceptual graph theory has achieved some computational results, based on the use of graph calculi, which could serve the interest of the deductive side of DRT.

Finally, there is an alternative semantics to the embedding of DRSs into classical models, which brings us to our next subject, dynamic semantics.

1.4.3 Dynamic view on conceptual graph semantics

It is often assumed that discourse interpretation is related to a dynamic process of discourse context evolution. When successively uttered sentences are processed by a hearer, they bring successive changes into the interpretation context of that hearer. When a simple conceptual graph is asserted, it introduces two kinds of information: a relation occurrence between concepts provides some factual information and a concept node introduces in the context an item that is available for further references. A semantics for conceptual graphs could take into account these two kinds of information and follow the way paved by dynamic semantics [GS91].

1.4.1. EXAMPLE. Imagine the following situation: the vocabulary is composed of three individual markers a , b and c , and two relations *entered* and *spoke*. Let $M = (D = \{A, B, C\}, F)$ be a classical model such that $F(a) = A, F(b) = B, F(c) = C, F(\textit{entered}) = \{A, B\}$ and $F(\textit{spoke}) = \{(A, B)\}$. We are at the

beginning of a conceptual graph discourse, only aware of our formal vision of the

world, M . Our discourse context is empty: $\langle \emptyset \rangle$

Suppose that we are told that “Someone entered.” $\boxed{*} \xrightarrow{1} \text{entered}$

We process this information by creating a record for that person in our information state and associating to it all objects provided by the model, which satisfies the graph utterance: A and B .

Our new discourse context is: $\langle \boxed{*} \begin{matrix} \nearrow A \\ \searrow B \end{matrix} \rangle$

If we are now told that “He spoke to b.” $\boxed{?} \xrightarrow{1} \text{Spoke} \xrightarrow{2} \boxed{b}$,

We can resolve the pronoun ‘he’ to the sole item in our discourse context and, given our model M , eliminate the possibility that the person at stake is referring to B .

If the last utterance had been “He spoke to a.”, no possible interpretation for the context item would have remained and we, as hearer, would have ended up in an ‘absurd state’. A possible escape would then be a rejection of (pieces of) the discourse or the revision of previously accepted information.

The meaning of a graph is given by the change its assertion brings into a context. The example only sketches the possibility of storing terms (and their possible interpretation) as contextual information. However far richer contexts could be conceived. For example, a context could also contain relational information between the stored discourse items with the consequence of a shift from the notion of test in a model to a notion of model construction.

The dynamic view of conceptual graph interpretation can be related to the process of constructing a complex representation by successive updates, each of them refining a previous representation with additional knowledge. This procedural view is common to conceptual graph incremental construction rules and to calculi based on successive graph derivations (see e.g., [Sow84] or [CM92]).

1.4.4 Concluding remarks

The aim of this brief jaunt into natural language semantics was twofold. First, to relate the layout of conceptual graph drawings to some structural properties of discourse and in particular, to the interaction of boolean connectives and quantifiers. Secondly, to provide the first steps toward stronger interactions between conceptual graph theory and linguistic theories. Such an exchange could benefit both sides. On one hand, it would strengthen the foundations of conceptual graphs in artificial intelligence, as natural language remains the most common way of communication between humans. On the other hand, such a link could

provide new insights into efficient computation for theories that are aimed at automated natural language processing.

However, despite the promising perspectives of further connections between conceptual graphs and natural language processing, a longer exposition of linguistic features would bring us out of our main trail, which is concerned with the use of conceptual graphs for efficient logical reasoning. We refer the reader to [KR93], [BB98], [GS91], [GSV96] and [Ben96] for deeper insights into discourse representation theory and dynamic semantics and their relation to logic and computer science. For a formal proposition of dynamic interpretation in conceptual graphs, see [Ker99a].

1.5 Conclusions

Languages of modern logics are essentially textual. It comes as no surprise that knowledge representation systems often adopt textual notations similar to those of their underlying logics. However, on the semantic side, sentences of these languages are interpreted with respect to structures. It is somehow paradoxical that these languages describe and refer to structured information by means of linear text, while linking sentences and structures is left to the interpretation process alone. To represent structured knowledge, it seems sensible to import as much as possible the object structure in the layout of the representation language. Conceptual graphs take this path by combining textual labels with node-edge drawings.

Along this introductory chapter, we have described influential graphical ancestors of conceptual graphs at the historical foundations of modern logic.

The importance of graphics and graphs to logic and its application to artificial intelligence has become a fact with the development of robotics, artificial vision and computers that are now provided with great graphical abilities. For instance, almost no operating system would now be commercialised with a single purely textual interface, no web-browser would be limited to the display of textual information. Pictures have found applications in logic and applied logics for educational purposes (e.g., the Hyperproof project), for representational and computational purposes (e.g., semantic networks, description logics or conceptual graphs).

Another reason for this regain of interest in pictures, which, after all, have always been used for every day communication since prehistoric times, might simply come from their cognitive power: perspicuity and efficiency. Such features appear in conceptual graphs under different aspects: primary gestalt view, distinction of structural components, simplicity of interpretation and visual derivative meanings of the representations. With the experience of working with conceptual graphs, acquired in the subsequent chapters, we will come back to this cognitive theme in the concluding chapter of the dissertation.

Finally, we have explored a linguistic aspect of conceptual graphs: their faithfulness to natural language structures. The perspective of rich interaction with fruitful linguistic theories has just been scratched and is promising for further research. In particular, the dynamic turn in semantics does not only confine itself to natural language semantics, it is also of great interest to computer science for which the dynamic notion of process is central. This brings us to the theme of our next chapter: the computational aspect of logical reasoning.

Chapter 2

The complexity of reasoning with graphs and fragments

We have previously argued that knowledge graph drawings enjoy some cognitive efficiency in representation. In this chapter, another form of (in)efficiency is considered, namely the difficulty of a typical computation task over these representations, viz. logical reasoning. Indeed, the drawings would not prove very useful if we could not exploit them to answer questions about the knowledge represented or to infer new knowledge from the existing one.

Now, the judgments of difficulty of inference tasks, and of relative merits of different representations, that one finds in the literature on graphical versus symbolic reasoning are often unsystematic and “impressionistic”. To advance beyond this stage, and get more definite insights, one needs a mathematical complexity analysis.

So far, the only successful style of analysis which has been developed is that of complexity theory in computer science, based on symbolic computation using Turing machines. We will adopt this here, and see what it tells us about the reasoning tasks in our area of interest. In particular, we find that we are operating in a much larger landscape of decidable calculi, with various subtle thresholds in complexity behaviour, e.g. from P to NP. Thus, we get a much more systematic picture of the potential of conceptual graph-based reasoning methods.

Of course, there is more to actual performance than abstract complexity, as the latter concerns worst cases. There is also average complexity, which needs to be explored in greater detail. But still, we regard this chapter as a necessary first step to operating at all: “si vis pacem, para bellum”.

Equally of course, complexity analysis based on a symbol-processing paradigm may seem inappropriate to analysing *graphical* reasoning. The matter is indeed delicate, and we will delay the discussion to the concluding chapter. For now, we will just say this: the analysis in this chapter is not sufficient. But, it is necessary.

In order to do complexity analysis, we need two ingredients. Conceptual graphs need to be represented symbolically in some textual language and we

need to fix our computation models over that language.

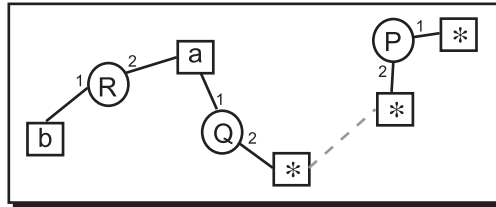
2.1 Conceptual graph drawings and formal languages

We have already seen, in Chapter 1, how some conceptual graphs (CG) correspond to first-order formulae. More precisely, different conceptual graph languages correspond to different fragments of classical first-order logic (FOL).

The connection to FOL, the prominent modern logic, can be traced back to Peirce’s first-order calculus of existential graphs. For the specific language of conceptual graph, [Sow84] offers a translation to FOL. The anchor to FOL is so strong that, in most of the conceptual graph literature, the semantics of conceptual graph fragments are only presented as translations to FOL languages. In this dissertation, we prefer to present both model theoretic analysis and translations following the line of [Ker96] or [Pre98].

The graph fragment of simple conceptual graphs is central to most CG systems. It corresponds to existential conjunctive FOL, i.e., the set of FOL formulae whose boolean connectives are conjunctions and whose quantifiers are existential ones, noted $FOL_{\{\exists, \wedge\}}$.

2.1.1. EXAMPLE.



A star within box represents an existentially quantified variable and two boxes linked by a dashed edge represent the same object. An occurrence of a relation symbol is encircled and connected by ordered edges to its arguments. A translation of the graph is the following sentence of $FOL_{\{\exists, \wedge\}}$:

$$Rba \wedge \exists x(Qax \wedge \exists y(Pyx))$$

To highlight the occurrence of the dashed edge (also called a coreference edge), we could alternatively choose to translate it into an equality atom and associate a “fresh” variable to each occurrence of a star marker. Thus, in $FOL_{\{\exists, \wedge, =\}}$:

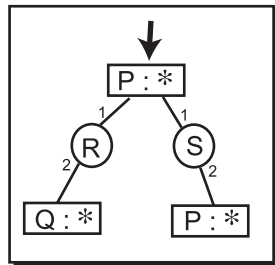
$$Rba \wedge \exists x(Qax \wedge \exists yz(Pyz \wedge x = z))$$

However, as the syntax of simple graphs forbids two constants to be connected by a coreference edge, equality symbols can always be eliminated to obtain an equivalent sentence in $FOL_{\{\exists, \wedge\}}$.

This example suggests another point: full first-order logic is not always needed and mostly, often not desired. Indeed, its nice high expressivity has a heavy counterpart for practical applications: its undecidability (i.e., the impossibility to guaranty an answer to some questions). The quest of efficient reasoning (where efficiency is expressed in terms of the amount of resources needed to solve a problem, such as time, space or the number of parallel processors) has been challenged in almost every field of computer science (artificial intelligence, knowledge representation, databases, software engineering, robotics, computational linguistics, etc.). From the diversity of applications with their associated specific reasoning tasks has resulted a multiplication of propositions; for instance, constrained query languages in databases [AHV95, KV00] or applied modal logics [DLNS94, Rou97, Mas98].

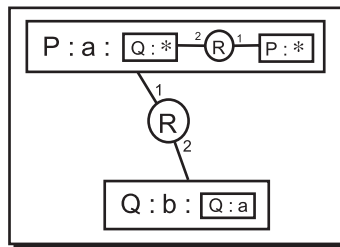
The connections between graph systems and FOL or fragments of FOL are not the sole ones to formal languages. Already in Peirce’s work on Gamma graphs (e.g., [Rob92]) appears the notion of intensional logic. Recent work in description logics (e.g., [BKM99] or [BMT99]) makes use of graph-homomorphism calculi for modal logics applied to knowledge representation. Therefore, it makes sense to look at connections with modal logics.

2.1.2. EXAMPLE. In [BMT99], a rooted simple conceptual graphs on some binary relational vocabulary is interpreted as a modal formula describing binary accessibility relations in a pointed Kripke model:



translates to $P \wedge \diamond_R(Q) \wedge \diamond_S(P)$

2.1.3. EXAMPLE. Nested simple conceptual graphs can also be translated to usual (textual) modal languages. For instance, in the following graph, the description associated to the constant ‘*a*’ may be interpreted as the representation of a world accessible by an ‘*a*-link’ from the world corresponding to the outermost box:



translates to $Pa \wedge Qb \wedge Rab \wedge \diamond_a(\exists x \exists y(Qx \wedge Py \wedge Ryx)) \wedge \diamond_b(Qa)$

By highlighting some problems relevant to knowledge representation systems and systematically studying them for fragments of common logics, this chapter will attempt to picture a landscape of (sometimes understudied) low fragments and provide a guideline for choosing extensions and restrictions of conceptual graph languages. But, in order to compare the behaviour of different logics, we need some measurement tools on a common ground. It is where complexity theory comes at stake.

2.2 Computation models

The complexity of a computational task is measured with respect to an abstract computation model which is independent from the kind of computers, the programming languages and the precise algorithms that may be put at work to solve the problem in real life (many of such models were introduced before the conception of the first computer). The most well-known of those models is the Turing machine.

2.2.1 Turing machines and equivalent models

The Turing machine [Tur37] is a primitive computer whose memory is a tape on which is written a string of symbols and whose set of actions is limited to moving a read/write-head on the tape and reading and writing the symbol placed under the head. Furthermore, the processor is programmed to conditionally perform these primitive actions (e.g., "if symbol x is read under the head then move one step to the right and write the symbol y "). Despite these apparent simple settings, Turing machines are capable of expressing any algorithm or simulating any rich programming language. It is precisely the simplicity of the execution steps that enables a sharp analysis of the processor's behaviour under a given program.

One of the most fundamental result of complexity theory is the equivalence between different computation models. In the early years of computation, motivated by the classical decision problem of first-order logic (the Entscheidungsproblem), several independent and equivalent models were proposed (e.g., Post's finite combinatory processes, Kleene's recursive functions, Church's λ -calculus or Markov's string-oriented algorithmic notation). The lesson to draw from these equivalences is that it is not the choice of a particular computation model that matters. Indeed, in the rest of the chapter, any specific choice of a computation model will be left aside, but instead, we shall apply reduction techniques that enable us to rely on known complexity results to draw new ones. What matters more is the kind of parameter that we intend to measure.

2.2.2 Time and space are chosen as measured resource variables

The amount of resources needed to solve a problem is expressed in terms of the size of a particular encoding of the input of a the given problem. Storing operations already bring to the fore the salience of space: how much space do we need to store the encoding of the input? How long are the intermediate results that do we need to write down during the execution of a program?

These space measurements suggest time related questions: How long does it take to read the input? How many of these intermediate results do we need to calculate and how many basic operations takes each of them? Just as modern physics teach us, time and space are intrinsically related. For instance, a problem that can be solved using polynomial time –i.e., the number of needed steps can be expressed as a polynomial of the size of the input, with constant exponents– can also be solved using polynomial space. This property forms a small stitch in the hierarchy of complexity classes which is partially described in Figure 2.1.

The class of problems that can be solved in polynomial time (P) is (strictly) included in the class of problems that can be solved in polynomial space (PSPACE).

Time and space are not the only variables that may enter the complexity dance. For example, with the recent development of networks and computers with multiple independent processors, the number of processors can be taken as another relevant parameter (see e.g., [JJ92] for an introduction to parallel processing complexity). Another trend in complexity theory, descriptive complexity (see e.g., [Imm99]), studies the complexity of expressing a property; roughly stated, language expressive power comes as another variable. As an example, polynomial time (P) corresponds to the class of queries describable by second-order-Horn formulae [Grä92].

For simplicity, we will remain in the classical trend and focus on space and time consumption. If the space required by an encoding of an input is so important, a crucial detail has been left unrevealed: how are formulae and structures going to be represented?

2.2.3 A detour into precise encodings

Although we often want to keep the freedom of choosing no finite bound on the amount of symbols available in our vocabulary, we face the evidence that any description of a finite input –i.e., formula, structure, graph, conceptual graph, etc.– uses a finite set of symbols. However, this set may vary with different instances and with different kind of inputs.

2.2.3.1 The case of formulae.

On one hand, the logical vocabulary and delimiters form a finite set which is fixed for all instances: \wedge, \vee, \neg_a (atomic negation), $\neg, \rightarrow, \leftrightarrow, \exists, \forall, (,), \diamond, \square$ ¹. On the other hand, relation symbols and terms may vary with different inputs and a way to obtain a uniform encoding method is to assign an index –i.e., a natural number– to every relation or term symbol of the input. For instance, the FOL-formula $\exists x \exists y (Px \wedge Qxy)$ could be represented by $\exists v_1 \exists v_2 (R_1 v_1 \wedge R_2 v_1 v_2)$ in which the i -th (from left to right) variable is represented by v_i and the i -th relational symbol is represented by R_i .

It should be noted that, if indexes were written in unary, then an encoding of a formula would require exponentially more occurrences of symbols than the formula in the usual base 10 notation and complexity results would deceptively be favourable. Indexes are therefore required to be encoded in any base n notation for $n > 1$. Binary notation is the usual choice for indexes and will be the chosen notation subsequently.

*Hence, the size of the encoding of a formula composed of \mathbf{m} symbols is bound by $\mathcal{O}(\mathbf{m} * \log \mathbf{m})$.*

Among formulae, conjunctions of atoms have some special features: first, syntactically, a conjunction of atoms is a sequence in which atoms and conjunction symbols alternate (i.e., contrary to the general case of formulae, logical symbols may not be concatenated) and also, semantically, a conjunction of atoms may be an equivalent representation to the interpretation of relations in a structure (i.e., for a structure M , $\llbracket \cdot \rrbracket_M$ may be represented by $\varphi \llbracket \cdot \rrbracket_M = \bigwedge_{1 \leq i \leq l} r_i \vec{t}_i$ such that $\vec{t} \in \llbracket r \rrbracket$ iff $r \vec{t} \in \varphi \llbracket \cdot \rrbracket_M$). It is therefore worth considering the size of an encoding of a conjunction of atoms.

*Let l be the number of atoms in a conjunction of atoms φ , u be the number of distinct terms in φ , r be the number of distinct relation symbols in φ and k be the maximal arity of a relation occurring in φ , it holds that the size of an encoding of φ is bound by $\mathcal{O}(l * (\log r + k * \log u))$.*

It is easy to verify that, even though more space than needed might be reserved for each atom (i.e., the place for k terms), as l, u, r , and k are smaller than the number m of symbols in φ , it holds that $l * (\log r + k * \log u)$ is polynomially related to $m * \log m$.

2.2.3.2 Finite structures.

We only consider function free languages. Hence, a finite structure is given by a finite set of objects U , a finite set of constants C , an interpretation $\llbracket c \rrbracket \in U$ for

¹In the case of multi-modal logics, we may have to treat modalities like relation symbols.

every constant $c \in C$, a finite set of relations R , an arity a_r and an interpretation $\llbracket r \rrbracket \in U^{a_r}$ for every relation $r \in R$. As in the case of formulae, objects, constants and relation symbols may be represented by indexed symbols (e.g., o_i for the i -th object in any chosen ordering).

We may decompose the encoding of a structure into two parts: on one hand, the universe, the interpretation of constants and the arity of each relation symbol², and on the other hand, the interpretation of relations. universe may be encoded by the greatest index for objects, the interpretation of constants by a list of numbers of corresponding objects.

For a structure with u objects, c constants and r relations of maximal arity k , the first part of the encoding of the structure is a string of size bound by

$$\underbrace{\log u}_{\text{universe}} + \underbrace{c * \log u}_{\text{constant interpretation}} + \underbrace{r * \log k}_{\text{relation arities}},$$

*hence $\mathcal{O}(c * \log u + r * \log k)$.*

We are presented different alternatives for the encoding of the interpretation of relations:

Standard encodings. The cautious way is to consider that the structure has as much probability of being dense as of being empty (the density of a model refers here to the sum of number of tuples occurring in the interpretation of each relation, i.e., $\sum_{r \in R} |\llbracket r \rrbracket|$). For a given relation r of arity a_r , the $|U|^{a_r}$ possible tuples can be ordered lexicographically and $\llbracket r \rrbracket$ can be represented by a binary word of length $|U|^{a_r}$: a 1 in a position i of the word indicates that the i -th a_r -tuple of objects is in the interpretation of r , and 0 codes the converse. $n = (|C| * \log |U| + \sum_{r \in R} |U|^{a_r})$ is to be taken as the size of an encoding of an input finite model.

*In practise, the number of constant being often much smaller than the number of k -tuples, we can usually take $\mathbf{n} = |\mathbf{R}| * |\mathbf{U}|^k$ as upper bound on the size of a structure where k is the greatest arity of the relations in R .*

2.2.1. EXAMPLE. let $U = \{o_1, o_2, o_3, o_4\}$, $R = \{r_1, r_2\}$, $\llbracket r_1 \rrbracket = \{\langle o_1, o_1 \rangle, \langle o_2, o_2 \rangle\}$ and $\llbracket r_2 \rrbracket = \{\langle o_1, o_3 \rangle, \langle o_1, o_4 \rangle\}$, $\llbracket \cdot \rrbracket$ can be encoded by the following word:

$$\underbrace{100001000000000000}_{\llbracket r_1 \rrbracket} \underbrace{0011000000000000}_{\llbracket r_2 \rrbracket}$$

²Depending on the way used for representing the interpretation of relations, this information could be retrieve from the latest, however, we may safely encode arities separately as this information requires a size polynomial in the size of the remaining parts of the total encoding of a structure.

Adapted encodings. We will encounter special situations in which the density of a model is extreme. Take, for instance, a structure in which every relation $r \in R$ is nearly universal –i.e., verified by almost all a_r -tuples of objects–, the previous encoding would be a waste; it would be more economical to encode only those tuples of objects that are *not* in the interpretation of a given relation; i.e., $\llbracket r \rrbracket$ can be represented by $\{\langle o_{i_1}, \dots, o_{i_{a_r}} \rangle / \langle o_{i_1}, \dots, o_{i_{a_r}} \rangle \notin \llbracket r \rrbracket\}$. In particular, for the complexity proof of model checking for the guarded fragment of FOL in Chapter 2.4.2, a balanced structure with two objects 0 and 1 and a single unary relation r such that $\llbracket r \rrbracket = \{0\}$ will be extended with an n -ary universal relation $\llbracket G_n \rrbracket = \{0, 1\}^n$, n being a variable taking part in the input. Extending a standard encoding of the model would require 2^n extra bits of information, whereas extending a representation of the negative information of the structure is only a matter of adding some information about the existence of the n -ary universal relation: the addition of $\log n$ bits to code the arity.

A model construction, which is recurrent to this thesis, concerns the minimal model of an existential conjunctive FOL-sentence. The minimal model $M(\varphi)$ of an existential conjunctive sentence φ with e existential quantifiers, c constants and l atoms is a structure with $u = e + c$ objects and l relevant tuples of objects –i.e., there are exactly l occurrences of 1's in the standard encoding of the previous paragraph–. Let r be the number of distinct relation symbols in φ and k be their maximal arity, it holds that $l \leq r * u^k$. The interpretation function for relations can this time be encoded in a similar way to the encoding of a formula –i.e., a sequence of relation symbols followed by their arguments–.

*Hence, the size of the formula-style encoding of a minimal model for $\varphi \in \mathbf{FOL}_{\{\exists, \wedge\}}$ is bound by $l * (\log r + k * \log u)$.*

2.2.2. EXAMPLE. The interpretation function in Example 2.2.1 (i.e., $u = 4$, $r = 2$, $k = 2$ and $l = 4$) can be alternatively be encoded by the following word of 20 bits (instead of 32 bits in Example 2.2.1):

$$\underbrace{\underbrace{\overbrace{0}^{r_1} \overbrace{00}^{o_1} \overbrace{00}^{o_1} \overbrace{0}^{r_1} \overbrace{01}^{o_2} \overbrace{01}^{o_2}}_{\llbracket r_1 \rrbracket} \overbrace{1}^{r_2} \overbrace{00}^{o_1} \overbrace{10}^{o_3} \overbrace{1}^{r_2} \overbrace{00}^{o_1} \overbrace{11}^{o_4}}_{\llbracket r_2 \rrbracket}}_{\llbracket r_1 \rrbracket \llbracket r_2 \rrbracket}}$$

For non-dense models –i.e., when l is much smaller than $r * u^k$ – with at least two objects, this encoding seems more economical than the standard encoding. A closer look at the worst case $l = r * u^k$ reveals that for a structure with at least two objects, the sizes of both proposed encodings are polynomially related. Indeed, $k * \log u \leq u^k$ for $u > 1$ and $k > 0$, hence $(l * (\log r + k * \log u)) \leq d * (r * u^k)^2$ for some constant d . In this

thesis, we have chosen to confine ourselves to complexity results that are not sensitive to a finer grain of measure than a polynomial relation between input encodings. Therefore, we will have the freedom to use the encoding which is the most suitable to the problem at stake.

2.2.3.3 Graphs and conceptual graphs.

Graph encodings exemplifies the collapse between existential conjunctive formulae and structures: a graph is a pair $G = (V, E \subseteq V \times V)$ where V is a set of objects, thus a structure with universe V and a single binary relation. Alternatively, a graph may also be represented as a conjunction of atoms³ $\bigwedge_{(v,v') \in E} Rvv'$. In graph terminology, these polynomially related encodings are known under the names *adjacency matrix* (standard encoding) and *edge list* (formula style encoding). Simple conceptual graphs surpass the limitation of a unique binary relation by the use of labelled edges and nodes. They have the expressive power of conjunctive existential predicate logic and therefore also enjoy polynomial translation between formula-style encodings and structure-style encodings.

The main conclusion to draw from these encoding comparisons is that finite structures can equivalently be coded by their k -dimension matrix (where k is the maximal arity of relations) or by a conjunction of atoms, however, the choice may matter if the structure has to be extended with additional information.

Though, it is informative to calculate the exact time and space consumption of a given algorithm on a given input, we are often interested in a larger scale classification of problems. The next section presents some of the usual classes considered in complexity theory.

2.2.4 Complexity classes that will be encountered

With P and PSPACE, we have already met two complexity classes. Figure 2.1 presents an overview of the hierarchy of complexity classes relevant to this chapter.

- Decidable versus undecidable. Problems can be partitioned into two main group of classes: decidable problems –i.e., problems for which we can exhibit a correct solving algorithm that will terminate on any instance– and by opposition, undecidable ones. For instance, satisfiability in FOL –i.e., the problem of determining the existence of a model satisfying a given first-order formula– is undecidable (Theorem 2.3.1). On the other hand, the same problem in the fragment $FOL_{\{\exists, \wedge\}}$ is decidable.

³For non-directed graphs, E is sometimes defined as a set of subsets of V with at most two elements; i.e., $E \subseteq \{X/X \subseteq V \ \& \ |X| \leq 2\}$ where singletons represent reflexive edges. To express the symmetry of non-directed edges, twin-atoms may also be added in $\bigwedge_{(v,v') \in E} Rvv' \wedge Rv'v$.

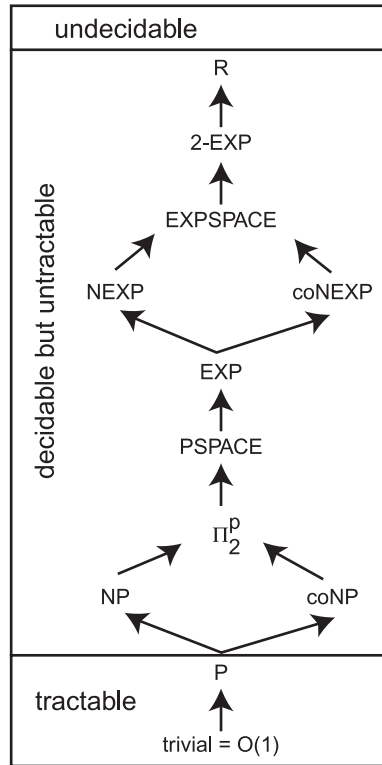


Figure 2.1: Complexity class hierarchy ($A \longrightarrow B$ represents $A \subseteq B$)

2.2.3. THEOREM. *satisfiability in existential conjunctive first-order logic is decidable.*

Proof: formulae of $FOL_{\{\exists, \wedge\}}$ have no mean to express contradictory information, hence the satisfiability problem is trivially decidable. ■

- **Tractable versus untractable.** Decidable problems can themselves be partitioned into tractable problems and decidable but untractable ones. The previous model construction is an example of a tractable task; it can be achieved in a reasonable amount of time (an amount of time linear –hence, polynomial– in the size of the input formula). Polynomial time problems (including trivial problems) are usually considered as tractable: the needed amount of resources grows “moderately” with the growth of the input.

On the other hand, a basic graph problem, *CLIQUE* falls into the difficult category: it is NP-complete.

2.2.4. DEFINITION. CLIQUE:

Input: a finite graph $G = (V, E \subseteq V^2)$ and a natural number $k \geq 2$

Question: does G includes complete subgraph of size k (i.e., a k -clique)?

Thus, a k -clique is a graph $G' = (V', E' \subseteq V'^2)$ such that (i) $V' \subseteq V$, (ii) $E' \subseteq E$, (iii) $|V'| = k$ and (iv) $\forall xy \in V', (x, y) \in E'$.

NP is the class of problems that can be solved in non-deterministic polynomial time; that is, if a potential solution can be non-deterministically guessed, the verification that it is a solution to the problem only takes polynomial time. Furthermore, a problem is complete for a complexity class if it belongs to the class and is as difficult as any other problem of the class: complete problems characterise their class. For instance, the trivial problem *satisfiability*($FOL_{\{\exists, \wedge\}}$) (Theorem 2.2.3), is obviously in NP: if we can guess the canonical model of a $FOL_{\{\exists, \wedge\}}$ -formula, checking the satisfaction of the formula by the model only takes linear time. However, *satisfiability*($FOL_{\{\exists, \wedge\}}$) is not NP-complete as it is much easier than some other problems in NP such as *CLIQUE* or *satisfiability* in propositional logic (*SAT*).

2.2.5. THEOREM. [Kar72] *CLIQUE is NP-complete.*

Symmetrically to NP, coNP is the class of problems that have a succinct disqualification: if we can guess a potential counter-example, then the verification that it is a counter-example only takes polynomial time. For instance, validity –i.e., the satisfaction by all structures– in propositional logic is in coNP: if a propositional formula is not valid, then for a guessed truth assignment that does not satisfy the formula, verifying that this truth assignment does not satisfy the formula takes linear time.

The frontier between tractable and untractable problems is thin. First, the polynomial time characterisation of a problem might not always be as satisfactory in practice as it appears theoretically: an algorithm whose complexity is with a high polynomial degree might not be enough efficient for a robot that requires quick reaction. On the contrary, some untractable problems can be solved reasonably fast for most inputs, while worst cases, that force the belonging to an untractable complexity class, are seldom encountered. Furthermore, the inequality $P \neq NP$ is suspected but not proven. All we can say so far, is that for no NP-complete problem, a deterministic polynomial time algorithm has been exhibited. For these reasons, the limit between modest complexity or not is sometimes set between polynomial classes and exponential ones.

- modest complexity versus high complexity. In this bipartition, all classes included in PSPACE are considered modest. The bottom line of high complexity classes is EXP: the class of problems that can be solved in exponential time (2-EXP being double exponential time). Similarly to the relation between P and NP, NEXP is the class of problems for which a candidate solution can be validated in exponential time. Its complement, coNEXP, is the class for which a candidate counter-example can be validated in exponential time. EXPSPACE is the class of problems that require exponential space, while on top of Figure 2.1, R denotes recursive languages.

The notion of class completeness brings us to our next subject: the transformation of a problem into another problem of known complexity.

2.2.5 Reductions

Reductions are techniques that enable to rely on known classification results for inferring new results: a problem A is at least as hard as a problem B if B reduces to A . We will exclusively deal with problems whose answer are either “yes” or “no” (decision problems). For a decision problem, B reduces to A means that there is a polynomial time transformation f which for any instance b of B provides an instance $f(b)$ of A such that the answer of A on $f(b)$ is the answer of B on b .

As an application of the method, we will prove the undecidability of consequence in positive FOL (i.e., the set of FOL formulae containing no occurrence of a negation symbol). From Church’s thesis [Chu36], we know that the validity problem is undecidable in FOL (Theorem 2.3.1). The consequence problem in positive FOL can be stated as follows: let (φ, ψ) be a pair of positive FOL formulae, is any model of φ a model of ψ ?

2.2.6. THEOREM. *Consequence($FOL_{\{\exists, \forall, \wedge, \vee\}}$) is undecidable.*

Proof: by reduction of *Validity(FOL)* to *Consequence($FOL_{\{\exists, \forall, \wedge, \vee\}}$)*. Let φ be a FOL sentence and ψ be the sentence obtained from the negation normal form of φ by replacing any negative literal $\neg P\vec{t}$ by a positive one $P^-\vec{t}$ such that P^- does not occur in φ .

$$\begin{aligned} & \models_{FOL} \varphi \\ \text{iff } & \bigwedge_{P \in \varphi} \forall \vec{x} (P\vec{x} \vee P^-\vec{x}) \wedge \bigwedge_{P \in \varphi} \neg (\exists \vec{x} (P\vec{x} \wedge P^-\vec{x})) \models_{FOL} \psi \\ \text{iff } & \bigwedge_{P \in \varphi} \forall \vec{x} (P\vec{x} \vee P^-\vec{x}) \models_{FOL} \psi \vee \bigvee_{P \in \varphi} \exists \vec{x} (P\vec{x} \wedge P^-\vec{x}). \end{aligned}$$

Obviously the input $(\bigwedge_{P \in \varphi} \forall \vec{x} [P\vec{x} \vee P^-\vec{x}], \psi \vee \bigvee_{P \in \varphi} \exists \vec{x} [P\vec{x} \wedge P^-\vec{x}])$ of the consequence problem in positive FOL is obtained from the input φ in polynomial time. ■

We can note that for the same reason that satisfiability was trivial in $FOL_{\{\exists, \wedge\}}$, it is also trivial in positive FOL: no contradiction can be expressed without some form of negation. It is also interesting to see that the particular case of *Consequence($FOL_{\{\exists, \forall, \wedge, \vee, functions, =\}}$)* where the premiss is a conjunction of ground literals, is NP-complete [Koz81]. This problem remains NP-complete without equality and without function symbols [Vor99].

Despite their fairly independent characteristic, classical computation models are used to measure the amount of needed resources in terms of the size of their input presented as a string of symbols. Does this mean that we have to abandon many of the graphical features and translate our drawings into strings? Maybe

not, but a paradigm of visual computation models would have to emerge together with real visual computers. In the mean time, actual computers only manipulate strings and therefore, it makes sense to renounce, for a while, some of the visual features such as proximity and compare the computational behaviour of different logics on a solid ground. An extended presentation of complexity theory and reduction techniques may be found in e.g. [Pap94].

With the machinery of complexity theory, we are now nearly equipped for comparing the computational behaviour of various logical fragments. However, we must first decide the kind of logical problems we are interested in.

2.3 The classical decision problem

We have already encountered several problems (e.g., *Consequence*($FOL_{\{\exists, \forall, \wedge, \vee\}}$), *SAT*, *CLIQUE*). The most prominent problem of modern logic is Hilbert's *Entscheidungsproblem*, the classical decision problem. It can be stated in different **equivalent** ways:

satisfiability(FOL): given a sentence of a first-order language, is it first-order satisfiable?

consequence(FOL): given two sentences of a first-order language, does the first one entail the second one?

validity(FOL): given a sentence of a first-order language, is it logically true?

φ entails ψ iff $\neg\varphi \vee \psi$ is valid iff $\varphi \wedge \neg\psi$ is not satisfiable

Church and Turing provided a negative answer to the decidability of this problem in its general form:

2.3.1. THEOREM. [Chu36, Tur37] *satisfiability*(FOL), *consequence*(FOL) and *validity*(FOL) are undecidable.

In the mean time, the restriction of the classical decision problem to certain syntactical forms had already been proven decidable (e.g., the decidability of satisfiability in monadic first-order logic by Löwenheim [Löv15]) and the general problem was restated as a classification of the classical decision problem for first-order fragments.

2.3.1 The classical decision problem in fragments of FOL

Most of the literature on decision problems in first-order logic has focussed on the satisfiability problem in fragments determined by quantifier prefixes, clausal forms (Krom or Horn formulae) and the vocabulary, witness the extensive survey [BGG97]. Examples of complexity results obtained by syntactic restrictions on FOL include:

2.3.1.1 Monadic versus polyadic.

Löwenheim [Löw15] proved the decidability of satisfiability for FOL formulae containing only unary relations and the undecidability of satisfiability for FOL formulae containing binary relations. The later result was refined by Kalmár [Kal36] who proved that one binary relation was sufficient.

2.3.1.2 Quantifier prefixes.

The Bernays-Schönfinkel fragment consists in the FOL formulae in prenex normal form of the form $\exists \vec{x} \forall \vec{y} \varphi$ where φ is quantifier free. Satisfiability in the Bernays-Schönfinkel fragment is decidable [BS28] (and NEXP-complete [Lew80]). By contrast and to highlight the relevance of the order in the interlacing of quantifiers, satisfiability in the fragment consisting of formulae of the form $\forall \vec{x} \exists \vec{y} \varphi$, where φ is quantifier free, is undecidable [Sko20].

2.3.1.3 Finite signature

When only two variables are available for building FOL-formulae, the satisfiability problem becomes NEXP-complete [GKV97]. On the other hand, with three variables or more the satisfiability problem is undecidable (e.g. [BGG97]). It should be noted that for a given *finite* relational vocabulary, there some k such that any formula is equivalent to a formula with at most k variables.

Satisfiability of Bernays-Schönfinkel sentences with a bound number of universal quantifiers (i.e., sentences in prenex normal form $\exists x_1 \dots \exists x_k \forall x_1 \dots \forall x_l \varphi$ where φ is quantifier free and $l \leq q$ for some fixed natural number q) is in NP [BGG97]. It follows a coNP upper bound on the consequence problem in existential FOL –i.e., FOL-formulae with no occurrences of universal quantifier in negation normal form– and sub-fragments, if the conclusion part of the input is restricted to existential sentences with a fixed bound on the number of quantifiers.

More drastically, satisfiability of FOL-sentences with a fixed bound on the amount of quantifiers and built from a finite vocabulary, is in P [BGG97]. It follows that *Consequence* in this fragment is also in P. The idea of the proof is to chose a linear order on the finite vocabulary and to pre-compute a finite table of results for the formulae whose matrix in conjunctive normal form (CNF) is minimal with respect to the order. Then, answering an instance of the problem can be done in logarithmic space as it amounts to a CNF-normalisation of the input and a check into a fixed table.

2.3.1.4 Modal restrictions.

The two variable fragment of FOL has already a modal flavour: it includes the standard translation of propositional modal logic [Gab71]. A closer look at the syntactic structure of the translated modal formulae reveals another property:

existential quantifiers are “guarded” by an atom whose arguments includes all free variables occurring in the scope of the quantifier. For instance $Q \wedge \diamond P$ corresponds to $Qx \wedge \exists y(Rxy \wedge Py)$ where the accessibility atom Rxy is called a guard for the quantifier $\exists y$. By dropping the restriction to use only two variables, Andr eka et al. [ABN98] defined the guarded fragment of FOL which enjoys the finite model property, the decidability of its satisfiability problem (2-EXP-complete [Gr a99]) and diverse model theoretic properties such as the interpolation property or equivalence under bisimulation. [Ben97] further extends the decidability result to the loosely guarded fragment of FOL in which quantifiers are guarded by conjunctions of atoms. We will elaborate on these bounded quantifications in Chapter 2.7 and in the next chapter.

Despite the large focus of the logic literature on the sole satisfiability problem, the three versions of the classical problem –i.e., satisfiability, validity and consequence– are of interest for fragments of FOL: their equivalence in FOL breaks down in fragments of FOL. For instance, we have seen that consequence in positive FOL is just as difficult as in full first-order logic (undecidable), whereas satisfiability in positive FOL is trivial.

Furthermore, the fragments we are interested in –i.e., languages missing some boolean connectives– are not the ones usually studied in logic. To some extent, our application driven route leads above all to the study of tractable fragments and to the point of view of computer science for which other benchmark problems also come to the fore.

2.4 Benchmarks

Although decidability and complexity studies in logic have traditionally been focussed on the satisfiability problem, we should not forget our conceptual graphs and the use we want to make of them: represent knowledge and reason on these representations. A knowledge representation system (KRS) has at least two ingredients: a knowledge base, in which the knowledge represented is stored, and an inference engine that solves problems by manipulating sentences according to a semantics. A third ingredient, which falls outside the scope of this study, concern some methods for revising the knowledge base with new evidences.

Different kinds of questions may fuel an inference engine; for instance, does a given knowledge base contain contradictory information? What is the set of all objects in a knowledge base satisfying some property? Is a given piece of information the logical consequence of a knowledge base (or of another piece of information)? In the eyes of these applications of logic different decision problems are salient and we will select a set of benchmark problems that will serve the comparison between languages.

2.4.1 Four problems

2.4.1.1 Satisfiability.

Proving the satisfiability (or consistency) of a knowledge base guarantees that it does not contain contradictory information and therefore, that it will not classically imply everything. It should be noted that, in practice, satisfiability tests of a knowledge base that is seldom modified, may be performed “offline”. Formally the problem can be formulated as follows:

Satisfiability(\mathcal{L}):

Input: a formula $\varphi \in L$ where L is a language for the logic \mathcal{L}

Question: is there a L -structure M such that $M \models_{\mathcal{L}} \varphi$?

Another problem concerning a single piece of information is *validity* (i.e., is a given formula satisfied by all structures?). Validity is a sign of non-informative content: asking a valid question about the information represented in a knowledge base does not provide any insight on the informational content of the knowledge base. On the other hand, from a valid knowledge base, can only be inferred valid representations: pieces of information whose logical truth is not related to the information contained in the knowledge base. For these reasons, validity will be left aside.

2.4.1.2 Consequence.

Proving the entailment of a piece of information by another piece of information is often considered as the central reasoning task in knowledge representation systems. This consequence problem takes different names depending on whether the premiss is called a knowledge base or not:

(i) *query answering*: given a knowledge base and a query, is it the case that the query follows from the information contained in the knowledge base?

(ii) *subsumption* (also called *query containment* in database theory): is some representation more general than another one? This last formulation is often used as part of a classification scheme.

Consequence(\mathcal{L}):

Input: two formulae $\varphi, \psi \in L$ where L is a language for the logic \mathcal{L}

Question: is it the case that for every L -structure M ,

$$M \models_{\mathcal{L}} \varphi \text{ implies that } M \models_{\mathcal{L}} \psi?$$

A variant where φ and ψ are formulae of two languages over the same vocabulary but that do not share exactly the same logical language (i.e., boolean connectors and quantifiers), will also be useful:

Consequence($\mathcal{L}, \mathcal{L}'$):

Input: $\varphi \in L$ where L is a language for the logic \mathcal{L}
 and $\psi \in L'$ where L' is a language for the logic \mathcal{L}'
Question: is it the case that for every L -structure M ,
 $M \models_{\mathcal{L}} \varphi$ implies that $M \models_{\mathcal{L}'} \psi$?

The consequence problem does not only play a preponderant role because of its association to information comparison tasks which are relevant to KRSs, but as well for the upper bound provided by its complexity on the one of satisfiability and of validity. Indeed, the consequence problem is always at least as difficult as the satisfiability problem as proving that a consequence holds amounts to proving the satisfaction of the conclusion in every model of the premiss. Furthermore, validity is a special case of consequence as a valid expression follows from the logical constant *True*.

Particular cases of consequence problems may arise from specific choices in some knowledge representation systems. For instance, some description logic systems distinguish an intensional knowledge level –i.e. knowledge about patterns in the relational structure represented– from an extensional (or assertional) one –i.e. knowledge about individuals in the relational structure–. In this case (e.g. [DLNS94]), *instance checking* is the problem of verifying that a particular relational pattern is true of a particular individual in a knowledge base, and the term *subsumption* is then reserved for consequence between constant-free sentences.

2.4.1.3 Model checking.

It happens that we are sometimes more interested in one particular model of a knowledge base than in all its satisfying structures. This is a common standpoint in database theory [AHV95, Via97, KV00], where a database instance may be considered as a finite relational structure providing a finite interpretation of the vocabulary. Queries are answered with pieces of information extracted from this finite description. Model checking also offers an approach in computer-aided design and verification [CE81, McM93, Kur94] which is modest enough to be practically applicable (tractable for many query languages). Model checking can be applied to chip designing, software programming, web-site conception or any activity requiring the design of a process that can be represented by a labelled transition system (an automata, simply a structure) and that needs to conform to some properties. These properties are expressed as formulae that must be satisfied by the model.

Model checking(\mathcal{L}):

Input: a formula $\varphi \in L$ and M a finite structure over L ,
 where L is a language for the logic \mathcal{L}
Question: is it the case that $M \models_{\mathcal{L}} \varphi$?

This generic definition of model checking is also called *uniform* as the sizes of both input parts come into account in the complexity analysis. In Database theory, two popular variants of model checking are considered (see e.g., [Var82, KV00]): by fixing the model –i.e., the structure is not part of the input, thus, its size does not take part in the complexity analysis of the problem–, one measures the expression-complexity of a non-uniform model checking problem. Symmetrically, data-complexity is obtained by fixing the query. For instance, in FOL [Var82]: both uniform and expression variants of model checking are PSPACE-complete, while the data-complexity is polynomial time (more precisely in the logarithmic-time class AC^0).

2.4.1.4 Model comparison.

In the line of model checking, another problem emerges: the comparison of two structures. Practical questions are for instance: are two databases equivalent or is one of the two more general than the other one? Are two programs equivalent –a model smaller than the other may then be more useful for model checking–? Variants of the model comparison problem are called *p-morphism* or bisimulation in modal logic [Ben96, BRV01], database conjunctive-query containment [KV00], constraint satisfaction problem [FV99] or automata simulation [Mil90]. An interesting relation to conceptual graph is the efficient bisimulation algorithm proposed in [DPP01] which is based on a graph algorithm in [PTB85]. Indeed, as we will see in Chapter 2.5 and in Chapter 3, for simple conceptual graphs, the equivalence of the consequence problem and the model comparison is exploited in a graph homomorphism consequence calculus, projection.

Model comparison(\mathcal{L}):

Input: two finite L -structures M and N , where L is a language for \mathcal{L}

Question: is there an homomorphism from N to M ($M \preceq N$)?

For these four benchmark problems, we already know some complexity results.

2.4.2 Results that set the scene.

	<i>satisfiability</i>	<i>consequence</i>	<i>model checking</i>	<i>model comparison</i>
<i>FOL</i>	<i>undecidable</i>	<i>undecidable</i>	<i>PSPACE</i>	<i>NP</i>
<i>FOL_{k>2} var</i>	<i>undecidable</i>	<i>undecidable</i>	<i>P</i>	<i>P</i>
<i>ML</i>	<i>PSPACE</i>	<i>PSPACE</i>	<i>P</i>	<i>P</i>

Undecidability results for the classical decision problem in first-order logic [Chu36, Tur37] motivated the foundation of complexity theory. The undecidability of the classical decision problem restricted to predicate logic with 3 variables

follows from the undecidability of satisfiability in the Kahr class [Kah62]: FOL formulae in prenex form $\forall x \exists y \forall z \varphi(x, y, z)$.

The PSPACE-completeness of (local) satisfiability and consequence in modal logic K was proven in [Lad77] and extended to the multi-modal case in [HM92]. Complexity classification has been salient to some applied modal logics in which the behaviour of satisfiability and consequence in various syntactic fragments are put under a microscope, e.g., [SSS91, DLNN97].

Model checking tasks have received a careful attention of finite model theory (e.g., [EF95]) and database theory (e.g., [AHV95, Imm82, HV91]). Chandra and Merlin [CM77] and Vardi [Var82] prove the PSPACE-completeness of uniform model checking in FOL. The polynomial time complexity of uniform model checking in variable bounded fragments of FOL is proven in [Imm82] (see also [Var95] for the complexity of the problem when the model is fixed –i.e., expression complexity– and when the formula is fixed –i.e., data complexity–). The polynomial time complexity of model checking in modal logic K follows from the inclusion of its standard translation in FOL_2 (see also [HV91] for a direct proof of this result). Some recent developments in database theory concern also bounded loosely guarded fragments of FOL. [GLS01] proposes an elegant game theoretic connection between the class of conjunctive queries with bounded hypertree-width and k -loosely guarded existential conjunctive FOL (i.e., subformulae are guarded by a conjunction of at most k atoms). Furthermore, [GLS99] proves the polynomial time complexity of query evaluation in these bounded fragments.

Studies on the complexity of model comparison can be traced back to early results on combinatorial problems on graphs such as CLIQUE [Kar72]. Verifying that a guessed correspondence between the universes of two FOL-structures corresponds to an homomorphism can be done in polynomial time (it is a check of every tuple in the conclusion is a tuple in the premiss under the substitution). On the other hand, the reduction CLIQUE into model embedding provides the NP-hardness. Polynomial results for structure equivalences in variable bounded fragments and modal logic are discussed in [Gro96].

We can also note that at the crossroad of FOL and ML, guarded fragments of FOL have decidable decision problems:

- Satisfiability is 2-EXP-complete for both the guarded fragment and the loosely guarded fragment [Grä99].
- Uniform model checking is P-complete for the restriction of the guarded fragment (GF) to a finite number of variables (or equivalently, to a finite relational vocabulary): indeed uniform model checking in $FOL_{k>2}$ is P-complete and the P-hardness proof in [Var97] uses an encoding inside GF_3 .
- For the whole guarded fragment, Maarten Marx proved the following theorem.

2.4.1. THEOREM. [Personal communication of Maarten Marx, ILLC, 2000.]
Uniform model checking is PSPACE-complete for both the guarded fragment and the loosely guarded fragment of FOL.

Proof: the PSPACE upper bound is provided by uniform model checking in FOL and the PSPACE-hardness by an extension of the reduction from the satisfiability problem of quantified boolean formulae (QBF) [SM73] to model checking in FOL. Let $\phi = \exists p_1 \forall p_2 \dots Q_n p_n \varphi$ be a QBF (i.e., Q_i is \exists if i is odd and \forall otherwise) and $M = (U = \{0, 1\}, \llbracket \cdot \rrbracket)$ with $\llbracket r \rrbracket = \{0\}$ and $\llbracket G \rrbracket = U^n$.

$$\text{Let } \phi' = \exists x_1 (\text{guard} \wedge \forall x_2 (\text{guard} \rightarrow \dots Q_n x_n (\text{guard op } \varphi[p_i/r(x_i)])) \dots)$$

such that $\text{guard} = Gx_1 \dots x_n$ and op is \wedge if it follows an existential quantifier and \rightarrow otherwise.

$$\phi \text{ is satisfiable iff } M \models \phi'$$

ϕ' is guarded and furthermore, the reduction is polynomial if the universal relation in the structure M is economically encoded as described in Chapter 2.2.3. ■

2.4.3 Conclusions

In order to compare logics and select some that might apply to conceptual graph formalisms, we have chosen four benchmark problems that are relevant to knowledge representation systems.

2.4.3.1 A festival of parameters.

The most enthralling fact to retain from the study of decision problem complexities is that there is a wide range of parameters that we may tune to change the rules of the game. The interlacing of different scales offers a variety of specific problems:

1. diverse semantics: we have focussed on first-order and local modal interpretations, however, we could also deal with propositional logic, epistemic logic or modal logic with global satisfiability or combination of logics such as first-order modal logic.
2. other generic problems: even though we have selected four of them, we have been aware of validity problems, instance checking, model equivalences, etc.
3. the vocabulary scale induced by constraints on the variable set, on the relational vocabulary or on the logical vocabulary.
4. the scale of syntactic constraints such as guarded forms and quantifier prefix restrictions.

5. the freedom of choosing premiss and conclusion from different logics over a common vocabulary.
6. restrictions on part of the input of a generic problem. We have mostly considered uniform problems –i.e., in which the sizes of both the premiss and the conclusion take part in the complexity measure– for their fair way of taking into account the size of all the available information. However, non uniform variants of model checking are commonly studied in database theory. Similarly, the freezing of one input side can also apply to consequence and model comparison.

This great amount of possibilities calls for selection –i.e., we will not draw an exhaustive landscape of results– and precision concerning the measured parameters. For instance, subsequently, all results will be about uniform problems except if explicitly stated.

2.4.3.2 Complexity collapses and divergences between problems.

The problems are not as independent as they may look. For instance, in all three (rather rich) logics of the previous table, consequence is just as difficult as satisfiability, but for positive FOL, these two problems are located at opposite extremities of the hierarchy –i.e., satisfiability is trivial, while consequence is undecidable–. We can also notice that the difference between model comparison and model checking in FOL, collapses in modal logic and finite-variable fragments of FOL. The interdependence of the benchmark problems is even more striking as for example, satisfiability may be put at work to compute a model and fuel model checking. Model comparison can be used to simplify a model and ease the model checking task.

With structure encodings we have caught a glimpse of an interdependence between structures and conjunctions of atoms. In the next section, we will observe that this interdependence can be formulated as an equivalence between consequence, model checking and model comparison in existential conjunctive FOL.

2.5 A key fragment: existential conjunctive FOL

Despite its apparent poor language, $FOL_{\{\exists, \wedge\}}$ proves useful in some practical applications where the information represented has the form of a list of positive atomic facts with a slight degree of indeterminacy introduced by the existential quantification (e.g., [GC97] studies document retrieval tasks in a simple conceptual graph knowledge base for libraries or [KV00] examines $FOL_{\{\exists, \wedge\}}$ as a language for database queries). One of the first asset of $FOL_{\{\exists, \wedge\}}$ is a semantic relationship between three of the benchmark problems.

2.5.1 Semantic relationships

Formulae of existential conjunctive logic ($FOL_{\{\exists, \wedge\}}$) enjoy a notion of minimal (or canonical) finite model: a structure containing exactly the positive information conveyed by a formula.

2.5.1. DEFINITION. Let $\varphi \in FOL_{\{\exists, \wedge\}}$ and ϕ be the Skolem form of φ –i.e., existential quantifiers are replaced by new constants or witnesses–. The minimal model of φ is the finite structure $M(\varphi) = (U, \llbracket \cdot \rrbracket_{M(\varphi)})$ defined as follows:

- U is the set of terms occurring in ϕ ,
- $\llbracket \cdot \rrbracket_{M(\varphi)}$ is defined on $R \cup C$ where R is the set of relation symbols occurring in φ and C is the set of constants occurring in φ .
- for a constant $c \in C$, $\llbracket c \rrbracket_{M(\varphi)} = c$ and
- for a relation symbol $r \in R$ of arity a_r , $\llbracket r \rrbracket_{M(\varphi)} = \{\langle u_1, \dots, u_{a_r} \rangle / r \ u_1, \dots, u_{a_r}$ is an atom in $\phi\}$.

2.5.2. FACT. It is easy to prove by induction on the structure of formulae that:

1. $\varphi \in FOL_{\{\exists, \wedge, \neg_{atomic}, \vee\}}$ is equivalent to a formula in prenex form $\exists \vec{x} \varphi'$ where φ' is quantifier free.
2. Minimal models are models: $M(\varphi) \models \varphi$ for $\varphi \in FOL_{\{\exists, \wedge\}}$.
3. The satisfaction of a formula in $\varphi \in FOL_{\{\exists, \wedge\}}$ is preserved under structure homomorphism: if $M \preceq N$ and $M \models \varphi$ then $N \models \varphi$.
4. For $\varphi, \psi \in FOL$, if $M \models \varphi$ and $\varphi \models \psi$ then $M \models \psi$ (by definition of entailment, every model of φ is a model of ψ).

We now have enough material to state a theorem that justifies the use of structure homomorphism algorithms to solve the consequence problem in simple conceptual graphs.

2.5.3. THEOREM. Let $\varphi, \psi \in FOL_{\{\exists, \wedge\}}$ and $M(x)$ be the minimal model of x ,

$$\varphi \models \psi \iff M(\varphi) \models \psi \iff M(\varphi) \preceq M(\psi)$$

Furthermore, these equivalence are complexity preserving.

Proof:

1. $\varphi \models \psi \longrightarrow_1 M(\varphi) \models \psi$

by Fact 2.5.2(2), $M(\varphi) \models \varphi$. Therefore, the hypothesis that $\varphi \models \psi$ and Fact 2.5.2(4) imply that $M(\varphi) \models \psi$.

The reciprocal could be shown from Fact 2.5.2(3) and the property that there is an homomorphism from $M(\varphi)$ to any model of φ .

2. $M \models \psi \longrightarrow_2 M \preceq M(\psi)$

Without loss of generality (by Fact 2.5.2(1)), let ψ be in prenex form $\exists \vec{x} \psi'$. By construction of $M(\psi)$, there is a bijection $F : \text{term}(\psi') \rightarrow \text{domain}(M(\psi))$ such that an atom $R\vec{t} \in \psi'$ iff $F(\vec{t}) \in \llbracket R \rrbracket_{M(\psi)}$.

Assume that $M \models \psi$, we must show that there exists an homomorphism $\pi : M(\psi) \rightarrow M$.

It follows from $M \models \psi$ that there exists a function $a : \text{term}(\psi') \rightarrow \text{domain}(M)$ such that:

- (i) a is conform to $\llbracket \cdot \rrbracket_M$ on constants occurring in ψ and
- (ii) if $R\vec{t} \in \psi'$ then $a(\vec{t}) \in \llbracket R \rrbracket_M$.

Thus $\pi = a(F^{-1})$ is a function from $\text{domain}(M(\psi))$ to $\text{domain}(M)$ such that for any relation R occurring in ψ , $\langle d_1, \dots, d_n \rangle \in \llbracket R \rrbracket_{M(\psi)}$ implies that $\langle \pi(d_1), \dots, \pi(d_n) \rangle \in \llbracket R \rrbracket_M$.

3. $M(\varphi) \preceq M(\psi) \longrightarrow \varphi \models \psi$

Assume that $M(\varphi) \preceq M(\psi)$ and $M \models \varphi$, we have to show that $M \models \psi$.

$M(\psi) \models \psi$, thus, by Fact 2.5.2(3), $M(\varphi) \models \psi$.

Furthermore by \longrightarrow_2 , $M \models \varphi$ implies that there exists an homomorphism from $M(\varphi)$ to M . Hence, by Fact 2.5.2(3), $M \models \psi$.

We have already shown, in Chapter 2.2.3, the polynomial time equivalence between a structure M and a formula $\varphi \in FOL_{\{\exists, \wedge\}}$ such that $M \models \varphi$. The encoding rewriting which was used in Chapter 2.2.3 is precisely the converse of the model construction step from the Skolem form of an existential conjunctive formula to its minimal model described in Definition 2.5.1. Furthermore, as the Skolemisation of an existential conjunctive sentence takes linear time (by a single run through the formula, existential quantifiers can be eliminated and variables be replaced by witnesses), the size of an encoding of φ and the size of an encoding of $M(\varphi)$ are polynomially related. ■

Similar problem equivalences and size preservation do also hold for restricted modal logics such as $ML_{\{\diamond, \wedge\}}$ and its enrichments with nominals and nominal binders; indeed, by standard translation, these fragments of modal logic correspond to fragments of existential conjunctive FOL.

The complexity preserving equivalences suggest the use of an homomorphism calculus on minimal models to decide an instance of the consequence problem. Projection [Sow84] is such a labelled graph homomorphism calculus for simple conceptual graphs (the conceptual graph fragment corresponding to $FOL_{\{\exists, \wedge\}}$). It remains to capture the complexity behaviour of our benchmark problems in this fragment.

2.5.2 Complexity

Excluded from the previous semantic equivalence, satisfiability in existential conjunctive FOL is very easy:

2.5.4. THEOREM. *Satisfiability($FOL_{\{\exists, \wedge\}}$) is trivial.*

Proof: for every positive fragment of FOL (i.e. $\forall X \subseteq \{\exists, \forall, \wedge, \vee\}, FOL_X$), satisfiability is trivially answered by the affirmative as no contradiction can occur. Any positive formula has a model with one element. ■

On the other hand, subsumption is untractable as $FOL_{\{\exists, \wedge\}}$ is a language of cliques.

2.5.5. THEOREM. *Consequence($FOL_{\{\exists, \wedge\}}$) is NP-complete.*

Proof: let $A = \exists x_1 \dots \exists x_k \mu$ where μ is quantifier free and contains l distinct constants. A has a canonical model of size $k + l$ represented by the set of atoms of μ where witnesses are substituted to existentially quantified variables, i.e. $A' = \{Rt_1 \dots t_m[x_1/w_1 \dots x_k/w_k] : Rt_1 \dots t_m \in \mu\}$. Let B' be the set of atoms in B , $A \models_{FOL} B$ iff there exists a substitution β for the free variables of B' such that $B'[\beta] \subseteq A'$. Let n be an upper bound of the length of the strings representing A' and B' . If β is guessed, $B'[\beta] \subseteq A'$ can be verified in $\mathcal{O}(n^2)$, therefore the problem is in NP.

To show the NP-hardness, we can reduce the NP-complete problem *CLIQUE* (see Definition 2.2.4 and Theorem 2.2.5) to *Consequence($FOL_{\{\exists, \wedge\}}$)*.

Let $G = (V, E \subseteq V^2, k \geq 2)$ with $V = \{c_1, \dots, c_m\}$ be an input of *CLIQUE*, we define $f(G) = (A, B)$ where $A = \bigwedge_{(c_i, c_j) \in E \text{ and } i < j} Rc_i c_j$ and

$$B = \exists x_1 \dots \exists x_k \bigwedge_{1 \leq i < j \leq k} Rx_i x_j.$$

An atom $Rt_1 t_2$ stands for an edge, A represents the set of edges V and B a complete graph with k vertices. It is easy to show that $G \in \text{CLIQUE}$ iff $A \models_{FOL} B$.

■

By language correspondence, the problem is equivalent to the NP-complete problem of proving the existence of an homomorphism between two labelled graphs (e.g., simple conceptual graphs) [Mug95].

As a corollary of Theorem 2.5.3 and Theorem 2.5.5, the remaining two benchmark problems are also untractable (concerning model comparison, after all, structures of $FOL_{\{\exists, \wedge\}}$ are just structures of FOL and the problem inherits the complexity of its richer parent).

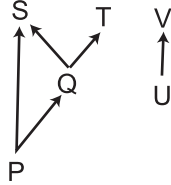
2.5.6. COROLLARY. *Model checking($FOL_{\{\exists, \wedge\}}$) and Model comparison($FOL_{\{\exists, \wedge\}}$) are NP-complete.*

Coming back to our conceptual graph systems, we may wonder what are the consequences of the information conveyed by a support on the behaviour of reasoning in the fragment.

2.5.3 Harmless meaning postulates

Simple conceptual graph systems add a supplementary touch of enrichment to existential conjunctive FOL: a finite subset R of the relational vocabulary is governed by a partial order \leq_R such that two relations with different arities are incomparable. The ordered set (R, \leq_R) is represented as an *acyclic directed* graph, called a relational support, in which the set of nodes corresponds to R and a directed edge (P, Q) can occur iff $P \leq_R Q$ and there is no S such that $P \leq_R S \leq_R Q$.

2.5.7. EXAMPLE. Let $\{P, Q, S, T, U, V\}$ be a set of k -ary relations for some $k \in \mathbb{N}^+$.



A directed edge represents a universal sentence; e.g., the edge from P to Q corresponds to $\forall \vec{x}(P\vec{x} \rightarrow Q\vec{x})$ where $\vec{x} = \langle x_1 \dots x_k \rangle$.

The whole graph corresponds to a conjunction of universal formulae, e.g.,

$$\sigma = \forall \vec{x}(P\vec{x} \rightarrow Q\vec{x}) \wedge \forall \vec{x}(Q\vec{x} \rightarrow S\vec{x}) \wedge \forall \vec{x}(P\vec{x} \rightarrow S\vec{x}) \wedge \forall \vec{x}(Q\vec{x} \rightarrow T\vec{x}) \wedge \forall \vec{x}(U\vec{x} \rightarrow V\vec{x})$$

Applying σ to an existentially quantified conjunction of atoms

$$\varphi = \exists \vec{y}(P\vec{y} \wedge Q\vec{b} \wedge S\vec{c} \wedge T\vec{d})$$

results in an extension of φ with new k -ary atoms:

$$\sigma(\varphi) = \exists \vec{y}(\underbrace{P\vec{y} \wedge Q\vec{y} \wedge S\vec{y} \wedge T\vec{y}}_{\sigma(P\vec{y})} \wedge \underbrace{P\vec{b} \wedge Q\vec{b} \wedge S\vec{b} \wedge T\vec{b}}_{\sigma(P\vec{b})} \wedge \underbrace{Q\vec{c} \wedge S\vec{c} \wedge T\vec{c}}_{\sigma(Q\vec{c})} \wedge \underbrace{U\vec{d}}_{\sigma(U\vec{d})})$$

We can note that for each atom of the original formula, each meaning postulate (MP) can apply at most once and that a meaning postulate cannot be triggered without a fitting atom. Indeed, a meaning postulate is a guarded formula; i.e., of the form $\neg(\exists \vec{x}(P\vec{x} \wedge \neg Q\vec{x}))$ where P and Q are relation symbols and every variable in the conclusion $Q\vec{x}$ occurs in the guard $P\vec{x}$.

Meaning postulates can be seen as part of the premiss of a consequence problem and influence the minimal model of a knowledge base. We note $FOL_{\{\exists, \wedge\}+MP}$ the class of FOL languages whose formulae are conjunctions of an existential conjunctive sentence and a conjunction of meaning postulates corresponding to a relational support.

A simple counting argument is sufficient to prove that the addition of meaning postulates to a knowledge base does not change the complexity of the benchmark problems:

2.5.8. THEOREM. *Satisfiability($FOL_{\{\exists, \wedge\}+MP}$) is trivial and Consequence($FOL_{\{\exists, \wedge\}+MP}, FOL_{\{\exists, \wedge\}}$) is NP-complete.*

Proof: For satisfiability, in the absence of negation and constant *FALSE*, meaning postulates cannot bring in contradictions.

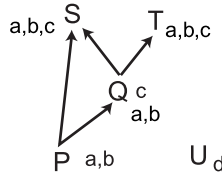
For consequence, we could fear an explosion of the size for the minimal model of the premiss. However, a closer look at the needed expansion of the minimal model can rest us. How many atoms would be sufficient to replace the meaning postulates in an equivalent “meaning-postulate-free” premiss?

For each atom of the premiss minimal model, each MP can only be triggered once as the guarded form of the MP makes it eventually introduce new relational occurrences, but no new object and no more than one new atom per application of an MP. Hence, a simple model saturation algorithm would make the minimal model of the premiss graph grow polynomially in the combined size of the model and the set of MP’s, in a polynomial number of expansion steps. ■

2.5.9. EXAMPLE. For instance, the minimal model of the sentence:

$$Pa \wedge Pb \wedge Qc \wedge Ud \wedge \forall \vec{x}(P\vec{x} \rightarrow Q\vec{x}) \wedge \forall \vec{x}(Q\vec{x} \rightarrow S\vec{x}) \wedge \forall \vec{x}(P\vec{x} \rightarrow S\vec{x}) \wedge \forall \vec{x}(Q\vec{x} \rightarrow T\vec{x})$$

represented by the graph:



is the minimal model of:

$$Pa \wedge Pb \wedge Qc \wedge Ud \wedge Qa \wedge Qb \wedge Sa \wedge Sb \wedge Sc \wedge Ta \wedge Tb \wedge Tc$$

The harmless role of meaning postulates in the complexity of the problems makes them often be considered as part of the input which is fixed beforehand; just like the vocabulary. Furthermore, the notion of guarded occurrences of quantifiers in MP's suggests several possible extensions:

1. the “acyclicity” property of the meaning postulate set may be dropped.
2. meaning postulates in the transitive closure of the implication set may freely be added.
3. the conclusion may not have the same arity as its guard.
4. the guard of a meaning postulate may be replaced by a loose guard; i.e. a conjunction of atoms.

2.5.4 Conclusion

The logic of simple conceptual graphs, existential conjunctive FOL, is not tractable for consequence, the prominent problem of KRSs. Fortunately, enriching the language with meaning postulates leaves the complexity of the benchmark problems unchanged:

	<i>satisfiability</i>	<i>consequence</i>	<i>model checking</i>	<i>model comparison</i>
$FOL_{\{\exists, \wedge\}}$	<i>trivial</i>	<i>NP – complete</i>	<i>NP – complete</i>	<i>NP – complete</i>

Two notions have been essential in this fragment. Firstly, the one of a minimal model of a $FOL_{\{\exists, \wedge\}}$ -sentence. The close relation between the size of an encoding of such a sentence and the one of its minimal model has repercussions under the form of problem equivalences for the fragment. These equivalences can serve the elaboration of calculi: e.g., model matching by homomorphism to compute consequences.

Secondly, a notion of guarded sentences, borrowed from modal formalisms, has already proved its attractive computational behaviour. Other applications of guards will later be examined.

2.5.4.1 A road map.

Some of fragments that gravitate around our central reference, existential conjunctive first-order logic, will be studied in the following two sections. The relatively low, but not safe enough, NP-complete complexity result suggests two directions to explore:

- fragments richer than $FOL_{\{\exists, \wedge\}}$ where decidability (or even better, modest complexity) can be preserved and
- tractable restrictions of $FOL_{\{\exists, \wedge\}}$.

In a first part, we consider expressivity extensions by addition of boolean connectors.

2.6 Fragments richer than $FOL_{\{\exists, \wedge\}}$

2.6.1 Including atomic negation

A first sensible extension of existential conjunctive first-order logic consists in the enrichment with atomic negation, the possibility of stating that some objects are not in a given relation. The question related to these enrichments is whether the already untractable, but modest, complexity results remain “reasonable”.

2.6.1.1 Benchmark problems on models

The complexities of model checking and model comparison do not increase:

2.6.1. PROPOSITION. *Model checking($FOL_{\{\exists, \wedge, \neg_a\}}$) and model comparison($FOL_{\{\exists, \wedge, \neg_a\}}$) are NP-complete*

Proof: for model comparison, upper and lower bounds are respectively provided by FOL and ($FOL_{\{\exists, \wedge\}}$).

For model checking, the NP lower bound of *model checking*($FOL_{\{\exists, \wedge\}}$) also remains an upper bound: if we can guess an assignment for the existentially quantified variables, then the verification can be done in a single (polynomial time) run through the formula and the structure. ■

2.6.1.2 Satisfiability problem

Even though the language is rich enough to express inconsistencies, satisfiability remains tractable.

2.6.2. PROPOSITION. *Satisfiability($FOL_{\{\exists, \wedge, \neg_a\}}$) is in P.*

Proof: a polynomial time algorithm goes through the matrix and checks that for every positive atom, its negation is not present. ■

In the further addition of universal quantification, satisfiability remains in polynomial time: *Satisfiability*($FOL_{\{\exists, \forall, \wedge, \neg_a\}}$) is in P. A quantified conjunction of literals is called an Herbrand formula and [BGG97](Chapter 8.2.2) proves that the unification problem, which is a known polynomial time problem, can be reduced to the satisfiability of Herbrand formulae.

Despite these encouraging results, difficulties arise with the consequence problem.

2.6.1.3 Consequence problem

In $FOL_{\{\exists, \wedge\}}$, proving the entailment between two formulae was only a matter of finding the right substitution for the variables in the conclusion into the Herbrand domain of the premiss. However, in the presence of atomic negation, this single substitution criterion fails. For instance, in the following instance of the problem, there is no substitution γ for x and y in $\{a, b, c\}$ such that $(Px \wedge Rxy \wedge \neg_a Py)[\gamma]$ follows from the premiss formula.

$$Pa \wedge Rab \wedge Rbc \wedge \neg_a Pc \models_{FOL} \exists x \exists y (Px \wedge Rxy \wedge \neg_a Py)$$

Atomic negation is a source of disjunction in disguise and many alternative models of the premiss must be taken in consideration: in the example, the information that “either Pb or not Pb ” is implicit, and in both cases a substitution that satisfies the conclusion can be found, but an exponential rewriting of the premiss might be required to make this implicit information become explicit.

The complexity of $Consequence(FOL_{\{\exists, \wedge, \neg_a\}})$ remains an open question.

In Chapter 2.7, we will examine an intermediate fragment between $FOL_{\{\exists, \wedge\}}$ and $FOL_{\{\exists, \wedge, \neg_a\}}$ for which consequence is non-deterministic polynomial time complete: it consists in formulae for which positive and negative atoms do not share variables. A further tree constraint on the syntax will provide a tractable fragment.

In theoretical database research, conjunctive queries including atomic negation have also drawn considerable attention. In particular, query containment check algorithms have been proposed for the containment problem in different subcases of conjunctive query languages with safe (i.e., any variable occurring in a negative literal must also occur in a positive literal) negation; see e.g. [LS93], [LMSS93], [LS95] or [FTU99].

Back to the drawings, the difficulty of reasoning implied by the representation of negated information can be related to the semantical difficulty of negations in pictures. Indeed, we may simply wonder what a negated picture means. It certainly loses its characteristic of direct mapping to the represented world, which we called *the faithfulness of drawings* in the previous Chapter. Despite this rising level of complexity, parallel to the rising of abstraction in language, it is interesting to observe how fast this process grows towards undecidability. A natural subsequent step is the study of existential FOL.

2.6.2 Existential first-order logic

Existential FOL is the FOL fragment including existential quantification and the three boolean operators: conjunction, disjunction and atomic negation. Compared to the situation in the previous fragment, satisfiability in existential FOL leaves the tractable classes and consequence, the modest complexity classes.

2.6.3. THEOREM. *Satisfiability($FOL_{\{\exists, \wedge, \vee, \neg_a\}}$) is NP-complete.*

Proof: a satisfiable existential sentence is satisfiable over the same domain as its Skolem form, and thus has a canonical model of size $k \leq n$, the number of quantifiers plus the number of constants. The guessed model is represented as a consistent list of atoms and constants values of size polynomial in n and the verification takes a time polynomial in the size of the list, therefore the problem is in NP. A reduction of *SAT* proves the NP-hardness, for instance a propositional formula $\phi(X_1, \dots, X_k)$ is satisfiable if and only if $\exists x_1 \dots \exists x_k \phi[X_i/Px_i]$ is satisfiable. ■

In fact, deciding the satisfiability of existential formulae with functions and equality is a NP-complete problem [BGG97].

2.6.4. THEOREM. *Consequence($FOL_{\{\exists, \wedge, \vee, \neg_a\}}$) is coNEXP-complete.*

A reduction of a tiling problem (e.g. [BGG97]) will be applied to prove the theorem. A tile is a 1×1 square. A tiling of a square X^2 is a covering of $X \times X$ with a given finite set of tiles $T = \{T_1, \dots, T_k\}$ in such a way that it respects two given binary compatibility relations H and V on T and a given initial condition w . $(t_i, t_j) \in H$ (respectively, V) means that t_j is allowed to be juxtaposed at the right (respectively, on top) of t_i . $w = \langle w_0, \dots, w_{n-1} \rangle \in T^n$ is a sequence of length n of tiles such that the word w covers the n leftmost places at the bottom of the square (i.e. $w_{i/0 \leq i \leq n-1}$ is required to be placed at the position $(i, 0)$). A tiling of the torus $Z(2^n)$ is a tiling of Z^2 that can be decomposed into juxtaposed copies of the tiling of a square with side 2^n (note that the definition of a tiling requires that a tile occurring on the border of the $2^n \times 2^n$ square is compatible with its juxtaposed neighbour).

2.6.5. LEMMA. *(Theorem 6.1.8 in [BGG97]) If any instance of the tiling of the torus $Z(2^n)$ is reducible to an instance φ of a problem P such that $|\varphi| \in \mathcal{O}(n \log n)$, then there exists $c > 0$ such that P cannot be decided in non-deterministic time $2^{cn/\log n}$*

Proof of Theorem 2.6.4: for a given vocabulary, let $FOL(\exists^*)$ be the set of sentences that are the existential closure of a quantifier free formula and $FOL(\forall^*)$ be the set of sentences that are the universal closure of a quantifier free formula. $FOL(\exists^* \wedge \forall^*)$ denotes the satisfiability problem of sentences of the form $A \wedge B$ where $A \in FOL(\exists^*)$ and $B \in FOL(\forall^*)$. $A \wedge B$ is unsatisfiable iff $A \models_{FOL} \neg B$. If $A \in FOL(\exists^*)$ and $B \in FOL(\forall^*)$, both A and $\neg B$ can be rewritten in negation normal form as sentences in $FOL_{\{\exists, \wedge, \vee, \neg_a\}}$. Hence, the complexity of $Consequence(FOL_{\{\exists, \wedge, \vee, \neg_a\}})$ is the complement of the complexity of $Satisfiability(FOL(\exists^* \wedge \forall^*))$.

For an upper bound, $Satisfiability(FOL(\exists^* \wedge \forall^*)) \in \text{NEXP}$ as any sentence in $FOL(\exists^* \wedge \forall^*)$ is in the Bernays-Schönfinkel fragment (FOL sentences with prefix $\exists^* \forall^*$) whose satisfiability is NEXP-complete [Lew80, BGG97].

To prove the NEXP-hardness, we reduce the tiling of the torus $Z(2^n)$ to *Satisfiability*($FOL(\exists^* \wedge \forall^*)$). It is sufficient to construct for any tiling instance (T, H, V, w) , a formula φ of the requested size and form –i.e., in $FOL(\exists^* \wedge \forall^*)_-$, such that φ is satisfiable iff (T, H, V, w) tiles $Z(2^n)$.

Let $T = \{T_1, \dots, T_k\}$ and $n = |w|$, a point $(x, y) \in Z(2^n)$ is encoded in binary by a $2n$ -tuple $(\bar{x}, \bar{y}) = (x_{n-1}, \dots, x_0, y_{n-1}, \dots, y_0) \in \{0, 1\}^{2n}$. There are $(k+1)$ $2n$ -ary predicates: S, t_1, \dots, t_k . S encodes the successor relation (i.e. addition of 1 to a number in binary), and $t_i(\bar{x}, \bar{y})$ stands for “a tile T_i is at the position (x, y) ”. The intended model has two elements 0 and 1. We first define the successor relation S such that $S(\bar{x}, \bar{y})$ iff $y = x + 1$, by the following conjunction $Succ = \forall \bar{x} \bar{y} (Succ_1 \wedge Succ_2 \wedge Succ_3)$. For convenience, we use implication and equivalence symbols inside quantifier free formulae, but they can be removed by applying the De Morgan’s laws.

The last bit alternates:

$$Succ_1 = S(x_{n-1}, \dots, x_1, 0, y_{n-1}, \dots, y_1, 1) \vee S(x_{n-1}, \dots, x_1, 1, y_{n-1}, \dots, y_1, 0)$$

If the last bit is 0 then it is replaced by 1 and the rest remains identical:

$$Succ_2 = S(x_{n-1}, \dots, x_1, 0, y_{n-1}, \dots, y_1, 1) \leftrightarrow S(x_{n-1}, \dots, x_1, 0, x_{n-1}, \dots, x_1, 1)$$

Otherwise, addition of 1 is shifting or going back to 0 if 2^n has been reached:

$$Succ_3 = S(x_{n-1}, \dots, x_1, 1, y_{n-1}, \dots, y_1, 0) \leftrightarrow [S(0, x_{n-1}, \dots, x_1, 0, y_{n-1}, \dots, y_1) \vee S(x_{n-1}, \dots, x_2, 1, 1, y_{n-1}, \dots, y_2, 0, 0)]$$

We now encode the conditions on the tiles: $Tiling = Tiling_1 \wedge Tiling_2 \wedge Tiling_3$.

There is exactly one tile at each point:

$$Tiling_1 = \forall \bar{x} \bar{y} (\bigvee_{i=1}^k t_i(\bar{x}, \bar{y}) \wedge \bigwedge_{1 \leq i < j \leq k} (\neg_a t_i(\bar{x}, \bar{y}) \vee \neg_a t_j(\bar{x}, \bar{y})))$$

Two juxtaposed tiles on a row have a H -match:

$$Tiling_2 = \forall \bar{x} \bar{y} \bar{z} (S(\bar{x}, \bar{z}) \rightarrow \bigvee_{(i,j) \in H} (t_i(\bar{x}, \bar{y}) \wedge t_j(\bar{z}, \bar{y})))$$

Two juxtaposed tiles on a column have a V -match:

$$Tiling_3 = \forall \bar{x} \bar{y} \bar{z} (S(\bar{y}, \bar{z}) \rightarrow \bigvee_{(i,j) \in V} (t_i(\bar{x}, \bar{y}) \wedge t_j(\bar{x}, \bar{z})))$$

Finally the initial condition w is encoded:

$$Initial = \bigwedge_{i=0}^{n-1} t_{w_i}(\bar{i}, \bar{0}).$$

$\varphi = Succ \wedge Tiling \wedge Initial$ is a universal formulae composed of atomic negations, disjunctions and conjunctions, thus of the proper form. Furthermore, φ is satisfiable iff there exists a tiling $f : Z(2^n) \rightarrow T$. The “if-direction” is easy as the formulae encode their intended meaning. Conversely, let M be a model of φ with universe $\{0, 1\}$. We associate to each point (x, y) of the square, the unique $2n$ -tuple of objects $g(x, y) = (x_{n-1}, \dots, x_0, y_{n-1}, \dots, y_0)$ such that (x_{n-1}, \dots, x_0) encodes x in binary and (y_{n-1}, \dots, y_0) encodes y in binary. At every point (x, y) , we put the unique tile T_i such that $M \models_{FOL} t_i(g(x, y))$. φ ensures that this defines a correct tiling of $Z(2^n)$ by T with initial condition w . Finally, φ is of size $\mathcal{O}(n \log n)$ and thus *Satisfiability*($FOL(\exists^* \wedge \forall^*)$) is NEXP-hard. Hence, *Consequence*($FOL_{\{\exists, \wedge, \vee, \neg_a\}}$) is coNEXP-complete. ■

We have now achieved a general view of complexity for first-order logic and some of its fragments. In particular, we found that even *small* fragments by the standards of a classical logician can still be quite complex. Therefore, we now turn to more *tractable* fragments, which drop to polynomial-time complexity, and which correspond more closely to the *positive information* aspect of conceptual graphs.

2.7 Two steps backwards into tractable worlds

We have seen that existential conjunctive FOL enjoys an equivalence between consequence and model comparison, which is complexity preserving. We can use the intrinsic structure of the models for delimiting tractable fragments. Acyclic simple conceptual graphs can be seen as representations of models with a tree structure.

2.7.1 Simple conceptual trees

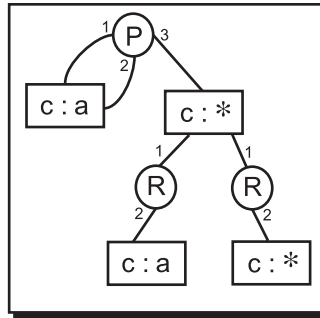


Figure 2.2: A simple conceptual tree

Mugnier [Mug95] proposes a polynomial time algorithm for deciding the existence of an homomorphism from a simple conceptual tree to a simple conceptual graph. The notion of tree is very natural in the graph language and corresponds in the textual language to the absence of “term cycles”. For instance, in Figure 2.2, there is no cyclic path of edges passing by least two relation nodes.

Baader et al. [BMT99] generalises Mugnier’s result to the absence of “variable cycles”. Indeed, cycles containing constants can be eliminated as the interpretation of a constant in the conclusion (and therefore, its image by homomorphism) is determined by its occurrence in the premiss. For instance, the cyclic graph represented in Figure 2.3 can be transformed into the equivalent tree in Figure 2.2 by splitting the node labelled with the term a .

The tree property (the absence of cycles) is highlighted in an inductive definition for the syntax.

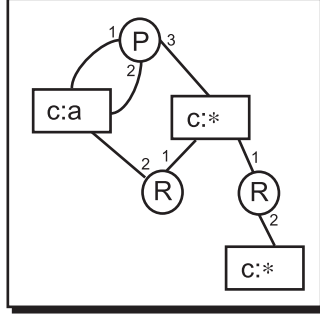


Figure 2.3: Eliminable cycles

2.7.1. DEFINITION. The existential conjunctive tree fragment (ECTF) is defined inductively by:

- Every atomic formula belongs to ECTF.
- ECTF is closed under conjunction.
- If (i) α is an atom and (ii) $\varphi = \varphi_1 \wedge \dots \wedge \varphi_n$ is a (possibly empty) conjunction of formulae in ECTF and (iii) for every $i/1 \leq i \leq n$, both $free(\varphi_i) \subseteq free(\alpha)$ and $|free(\varphi_i)| \leq 1$ then for every $\vec{x} \subseteq free(\alpha)$, $\exists \vec{x}(\alpha \wedge \varphi)$ is in ECTF.

If \vec{x} is the empty sequence, then $\exists \vec{x}(\alpha \wedge \varphi)$ is understood as $(\alpha \wedge \varphi)$. An ECTF sentence is a formula $\varphi \in ECTF$ such that $free(\varphi) = \emptyset$.

The atom α , which relativises the existential quantifier, is called a guard. The only possible variable shared by two sub-formulae φ_i and φ_j in $\exists \vec{x}(\alpha \wedge \varphi_1 \wedge \dots \wedge \varphi_n)$ must also occur in their guard. For instance, the simple conceptual tree in Figure 2.2 can be translated in an ECTF sentence:

$$\exists x(Paax \wedge Ca \wedge Cx \wedge Rxa \wedge \exists y(Rxy \wedge Cy))$$

2.7.2. PROPOSITION. Let $A \in FOL_{\{\exists, \wedge\}}$ and $B \in ECTF$ be two sentences, $A \models B$ can be decided in polynomial time.

Proof: The result directly follows from equivalent expressive power of simple conceptual trees and ECTF and from the complexity of the consequence problem with a simple conceptual graph premiss and a simple conceptual tree conclusion: Theorem 3.2.9[CM92, Mug95]. ■

2.7.3. COROLLARY. Let M be a first-order structure and $B \in ECTF$, $M \models B$ and $M \preceq M(B)$ can be decided in polynomial time.

Proof: the complexity of model checking follows from proposition 2.7.2 and the formula-style encoding of a model Chapter 2.2.3. The particular instance of model comparison is a direct consequence of the complexity preserving equivalences in $FOL_{\{\exists, \wedge\}}$ (Theorem 2.5.3). ■

The tree property of ECTF will later be generalised to a wider fragment of $FOL_{\{\exists, \wedge\}}$: the existential conjunctive guarded fragment. For the time being, we can examine how the “tree idea” can be applied to negative information.

2.7.2 A bit of negation: non-interlaced positive and negative information

There is a fragment of $FOL_{\{\exists, \wedge, \neg_{atomic}\}}$ which inherits the complexity behaviour of $FOL_{\{\exists, \wedge\}}$: the set of existential conjunctive formulae with atomic negation in which no pair of literals of opposite sign share some variable. For instance, $\exists x \exists y (Px \wedge Rxa \wedge \neg Pa)$ is acceptable in the fragment, while $\exists x \exists y (Px \wedge Rxy \wedge \neg Py)$ is not (Rxy and $\neg Py$ are of opposite sign and share the variable y).

2.7.4. DEFINITION. [Rewriting negative literals into positive ones] For a formula $\varphi \in FOL_{\{\exists, \wedge, \neg_{atomic}\}}$, the positive formula $\varphi^+ \in FOL_{\{\exists, \wedge\}}$ is obtained from φ by replacing every negative literal $\neg R(\vec{t})$ by an atom $R^-(\vec{t})$ such that R^- does not occur in φ . We note \mathcal{R}^- the set of all these new relation symbols.

This rewriting of negative literals into positive ones takes linear time. Furthermore, it preserves the semantics of these particular negated representations:

2.7.5. THEOREM. *Let $\varphi, \psi \in FOL_{\{\exists, \wedge, \neg_{atomic}\}}$ such that $\varphi \not\equiv \perp$ and ψ is of the form $\exists \vec{x} \alpha \wedge \exists \vec{y} \beta$ where α is a conjunction of positive literals and β is a conjunction of negative literals.*

$$\varphi \models \psi \iff \varphi^+ \models \psi^+$$

Proof: Lemma: $M(\varphi^+) \models \varphi$ (by construction of φ^+).

We can use Theorem 2.5.3 and show that $\varphi \models \psi \iff M(\varphi^+) \models \psi^+$.

• \longleftarrow : assume $M(\varphi^+) \models \psi^+$ and $M \models \varphi$.

We first extend M to the extra vocabulary: M' has the same domain and same interpretation function for constants as M and $\llbracket R \rrbracket_{M'} = \llbracket R \rrbracket_M$ and $\llbracket R^- \rrbracket_{M'} = \{\langle d_1, \dots, dn \rangle / \langle d_1, \dots, dn \rangle \notin \llbracket R \rrbracket_M\}$.

There is an homomorphism from $M(\varphi^+)$ to M' which preserves the satisfaction of positive existential conjunctive formulae, thus $M' \models \psi^+$.

By construction of M' , $M' \models \psi^+$ implies that $M' \models \psi$. Indeed, $\llbracket r^- \rrbracket_{M'} \cap \llbracket r \rrbracket_{M'} = \emptyset$.

Finally, M and M' only differ on the interpretation of the \mathcal{R}^- relations (which do not occur in ψ), hence, $M \models \psi$.

- \longrightarrow : assume that $\forall M, M \models \varphi \rightarrow M \models \psi$.

It is easy to verify that $M(\varphi^+) \models \varphi$. Therefore, $M(\varphi^+) \models \psi$.

We must show that (i) $M(\varphi^+) \models \exists \vec{x}\alpha$ and (ii) $M(\varphi^+) \models \exists \vec{y}\beta^+$ (the special form of ψ makes possible to divide the work in two parts, one for positive information and one for negative information).

(i) $M(\varphi^+) \models \exists \vec{x}\alpha$ directly follows from $M(\varphi^+) \models \psi$.

(ii) We can extend $M(\varphi^+)$ as follows: M' has the same domain and same interpretation function for constants as $M(\varphi^+)$ and $\llbracket R^- \rrbracket_{M'} = \llbracket R^- \rrbracket_{M(\varphi^+)}$ and $\llbracket R \rrbracket_{M'} = \{\langle d_1, \dots, dn \rangle / \langle d_1, \dots, dn \rangle \notin \llbracket R^- \rrbracket_{M(\varphi^+)}\}$.

M' extends $M(\varphi^+)$ with some positive facts which are not in conflict with the negative information conveyed by φ , thus $M' \models \varphi$ and so $M' \models \psi$. Hence $M' \models \exists \vec{y}\beta$. By construction, $\langle d_1, \dots, dn \rangle \notin \llbracket R \rrbracket_{M'}$ iff $\langle d_1, \dots, dn \rangle \in \llbracket R^- \rrbracket_{M'}$, thus $M' \models \exists \vec{y}\beta^+$. As $\forall R^- \in \mathcal{R}^-$, $\llbracket R^- \rrbracket_{M'} = \llbracket R^- \rrbracket_{M(\varphi^+)}$ and the satisfaction of $\vec{y}\beta^+$ in M' only depends on the denotation of relations in \mathcal{R}^- , it follows that $M(\varphi^+) \models \vec{y}\beta^+$. ■

To highlight the importance of a strict separation between positive and negative pieces of information, follows a counter-example⁴ of the theorem in the general case of $FOL_{\{\exists, \wedge, \neg_{atomic}\}}$ is for instance,

$$Pa \wedge Rab \wedge Rbc \wedge \neg Pc \models \exists x \exists y (Px \wedge Rxy \wedge \neg Py)$$

Indeed

$$Pa \wedge Rab \wedge Rbc \wedge P^-c \not\models \exists x \exists y (Px \wedge Rxy \wedge P^-y)$$

As a direct consequence of the previous theorem (Theorem 2.7.5) and of Proposition 2.7.2 (i.e., the tractability of consequence for tree structures), follows the following tractability result for a tree fragment of $FOL_{\{\exists, \wedge, \neg_{atomic}\}}$.

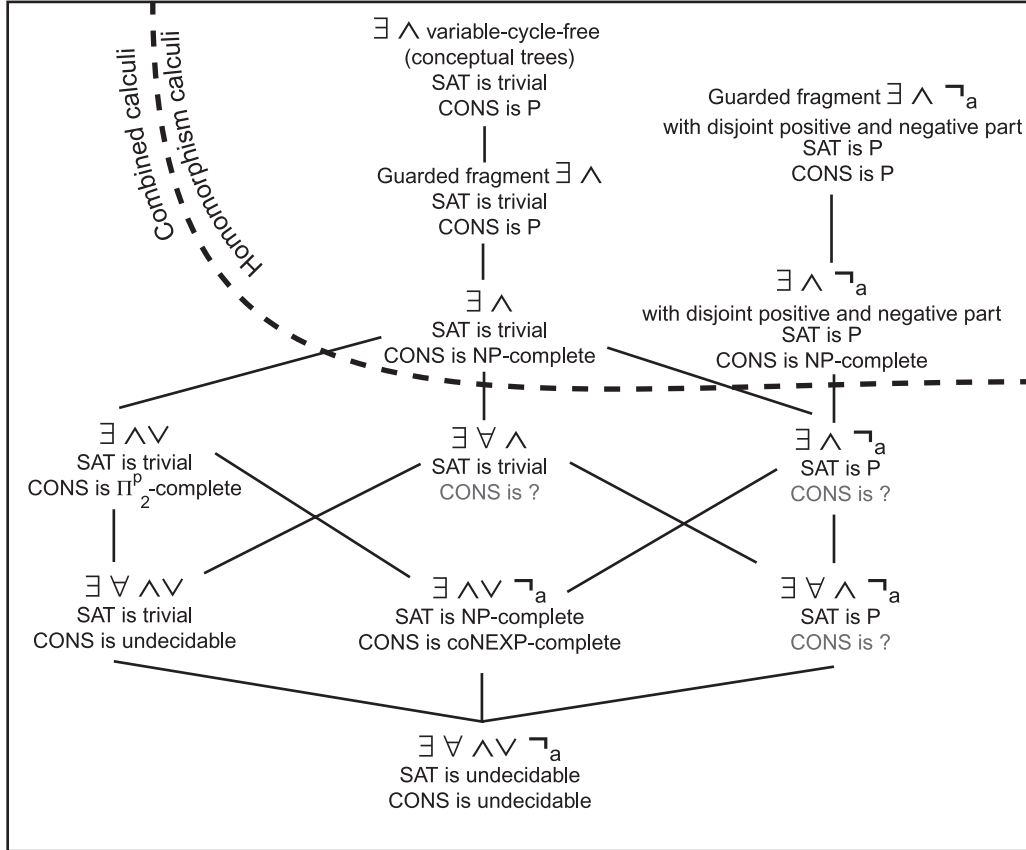
2.7.6. COROLLARY. *Let $\varphi, \psi \in FOL_{\{\exists, \wedge, \neg_{atomic}\}}$ such that (i) $\varphi \not\models \perp$ and (ii) ψ is of the form $\exists \vec{x}\alpha \wedge \exists \vec{y}\beta$ where α is a conjunction of positive literals and β is a conjunction of negative literals.*

$\psi^+ \in ECTF$ implies that $\phi \models \psi$ is decidable in polynomial time.

We may now gather the learned results on the computational behaviour of various predicate logic fragments.

⁴In personal communications, Geneviève Simonet proposed me related counter-examples to the completeness of projection for simple conceptual graphs in presence of negated concepts.

2.8 The travelled FOL landscape



From the complexity results, we can note the following points:

1. With language impoverishment, the benchmark problems do not evolve in a uniform direction. For instance, the step from FOL to positive FOL leaves consequence undecidable whereas satisfiability goes from undecidable to trivial.
2. Some low fragments enjoy a polynomial time equivalence between consequence and minimal-model checking (e.g., $FOL_{\{\exists, \wedge\}}$ and the fragment of $FOL_{\{\exists, \wedge, \neg_{atomic}\}}$ with non interlaced positive and negative information). It is this property which will be exploited in graph-homomorphism calculi for corresponding conceptual graph languages.
3. The frontier between tractability and untractability is subtle. Some tree characterisation is present in all examined fragments which are tractable for consequence.

This last property, which is central to modal logics, motivates a change of perspective. Tree properties are central to *modal languages*, whose raison d'être

is precisely their good balance between expressive power and low complexity (cf. [BRV01] and [Are00] for extended discussions of modal formalisms and their advantages). Therefore, we also take a look at what is known about the complexity of modal fragments, which brings us to an extension of the earlier tractability results.

2.9 The modal perspective: expressivity at low costs

In modal logics, formulae are not evaluated by an observer having an overview on a structure like in first-order logic, but by focussing on a particular point of a structure and moving the focus to accessible locations along paths of relational edges. This local internal view-point is acknowledged as a main strength of modal logics applied to artificial intelligence problems. Indeed, this locality principle not only leads to advantageous complexity results, but it also meets the needs of many applications, witness the use of modal formalisms in, among others, program verification (PDL, e.g., [Mas98]), multi-agent systems (epistemic logic, e.g., [FHMV95]), linguistic structures (feature logic, e.g. [Rou97]) and our present subject, knowledge representation (description logic, e.g., [DLNS94]).

A language of modal logic is a language of propositional logic extended with unary operators, modalities. The semantic message of such operators is an indication to move the local point of interpretation to other accessible locations. It is a peculiarity of modal languages to talk about local phenomena without explicitly naming the places where they occur. Hybrid languages naturally emerged as modal languages extended with some of the apparatus of first-order logic: the ability of naming the objects inhabiting the structures, of representing them by place holders (variables) and associated machinery (classical quantifiers or more specific binders). Such extensions of orthodox modal logics are present in hybrid logic, tense logic, feature logic or description logic.

2.9.1. DEFINITION. A vocabulary is defined by four disjoint countable sets: $\mathcal{P} = \{P, Q, \dots\}$ of propositional variables, $N = \{i, j, \dots\}$ of nominals, $VAR = \{x, y, \dots\}$ of nominal variables and $MOD = \{1, 2, \dots\}$ of modality indexes.

For a vocabulary, the well-formed formulae of a language of modal logic (ML) are defined by:

$WFF := True | A | \neg_a A | \neg \varphi | \varphi \wedge \psi | \varphi \vee \psi | \diamond_m \varphi | \square_m \varphi | @_l \varphi | \downarrow x(\varphi)$
 where $A \in \mathcal{P} \cup N \cup VAR$, $m \in MOD$, $l \in N \cup VAR$ and $x \in VAR$.

A class of modal languages is denoted by a subset of $\{\diamond, \square, \wedge, \vee, \neg_a, \neg, N, @, \downarrow\}$. For example, $L \in ML_{\{\diamond, \wedge, \neg\}}$ is a classical language of multi-modal logic. Note that, in well-formed formulae, the presence of @ requires the one of N , but not

vice-versa, and that the absence of \downarrow makes nominal variables useless (as inputs are required to be sentences). Furthermore, all languages include the logical constant *True*.

The variable binder \downarrow bookmarks the actual point of evaluation by storing it in a variable (for this reason, it is also called the *Here and Now* binder [BT99]): $M, w \models_{ML} \downarrow x(\varphi)$ if $M, w \models_{ML} \varphi[x/w]$. The jump operator enables to move the evaluation point to the world denoted by a nominal or a bound variable: $M, w \models_{ML} @_i \varphi$ if $M, V(i) \models_{ML} \varphi$.

The literature on the complexity of modal formalisms is broad and diversified. For in-depth studies, we refer the interested reader to the following non-exhaustive list of publications:

[Lad77], [BRV01], [Hem01] (modal logics), [ABM99a], [ABM99b], [ABM00] and [Are00] (hybrid logics), [DLN⁺92], [DLNS94] and [DLNN97] (description logics). Concerning the later knowledge representation formalism, the search of efficient reasoning techniques in description logics has led to an extensive study of fragments of modal logics and to a recent focus on graph homomorphism methods in e.g., [BKM99] and [BMT99]).

Before exploring the behaviour of our benchmark problems in modal fragments, we will first describe a direct application of modal ideas to predicate logic.

2.9.1 The existential conjunctive guarded fragment

The guarded fragment of FOL (GF) arose as a generalisation of the standard translation of modal logics into classical logic and the search for a fragment preserving some modal model-theoretic properties (see for instance [Ben85] and [ABN98]). GF includes the translation of the corresponding modal formalisms, while having an expressivity advantage for knowledge representation: no predicate arity limitation. In the guarded fragment, satisfiability can be decided in deterministic double exponential time (2-EXP-complete) [Grä99] and model checking is PSPACE-complete⁵. We will prove in Chapter 3 that ECGF, the intersection of the guarded fragment with $FOL_{\{\exists, \wedge\}}$, has a deterministic polynomial time consequence problem.

2.9.2. DEFINITION. The existential conjunctive guarded fragment (ECGF) is defined inductively by:

- Every atomic formula belongs to ECGF.
- ECGF is closed under conjunction.
- If (i) α is an atom (called a guard) and
(ii) $\varphi = \varphi_1 \wedge \dots \wedge \varphi_n$ is a (possibly empty) conjunction of formulae in ECGF

⁵Maarten Marx, Model checking in the guarded and packed fragments. Unpublished note, ILLC, 2000.

- and
- (iii) for every $i/1 \leq i \leq n$, $free(\varphi_i) \subseteq free(\alpha)$ and
 - (iv) $\vec{x} \subseteq free(\alpha)$,
- then $\exists \vec{x}(\alpha \wedge \varphi)$ is in ECGF.

The atom α relativises the quantifier sequence: it guards the sub-formulae φ_i 's by sharing their free variables.

The similarity of the tree fragment of simple conceptual graphs with ECGF is striking. Indeed, the later is a generalisation of the notion of guarding used in the former.

2.9.3. FACT. ECTF is strictly included in ECGF.

The only difference between the two fragments resides in the constraint on the cardinality of the set of free variable occurring in a guarded sub-formula φ_i . Furthermore $\exists x \exists y (P(x, y) \wedge R(x, y))$ is in ECGF but not in ECTF.

A particular case of ECGF has been studied in the description logic \mathcal{ELIRO}^1 . An \mathcal{ELIRO}^1 -sentence φ translates into an ECGF-formula $ST(\varphi)$ with only unary and binary predicates. [BMT99] proves that $Consequence(\mathcal{ELIRO}^1)$ is in P. In Chapter 3, we will prove the following generalisation of both the complexity of consequence in \mathcal{ELIRO}^1 and of the complexity of projection in simple conceptual trees.

2.9.4. PROPOSITION. *If A is a sentence in $FOL_{\{\exists, \wedge\}}$, B is an ECGF-sentence and M is a FOL-structure, then $A \models B$ can be decided in polynomial time.*

The result directly follows from the equivalence between the fragment of guarded simple conceptual graphs and ECGF and from the polynomial time algorithm for projecting a guarded simple graph onto a simple conceptual graph; See Chapter 3.3 in Chapter 3).

2.9.5. COROLLARY. *Let M be a first-order structure and $B \in ECGF$, $M \models B$ and $M \preceq M(B)$ can be decided in polynomial time.*

Proof: similarly to the case of ECTF, the complexity of model checking follows from proposition 2.9.4 and the formula-style encoding of a model (Chapter 2.2.3). The particular instance of model comparison is a direct consequence of the complexity preserving equivalences in $FOL_{\{\exists, \wedge\}}$ (Theorem 2.5.3). ■

For the time being, the main ideas of the algorithm will be presented in an example.

2.9.6. EXAMPLE.

Let $\varphi = Paa \wedge Raaa \wedge \exists x(Pax \wedge Rxaa \wedge Rrax) \in FOL_{\{\exists, \wedge\}}$

and $\psi = \exists x \exists y (Pxy \wedge Rxyy \wedge \exists z (Rzxy)) \in ECGF$

From Theorem 2.5.3, we know that $\varphi \models \psi$ iff $M(\varphi) \models \psi$ and that the minimal model of φ can be represented by the set of ground atoms occurring in the Skolemised form of φ .

E.g., $M(\varphi) = \{Paa, Raaa, Pab, Rbaa, Rbab\}$

Without loss of generality, ψ is considered to be pure –i.e., no variable is quantified twice–. A guarded existential conjunctive formula is represented by a tree with nodes labelled by atoms and such that a guard occurs as predecessor of the subformulae in its scope.

For instance, ψ is represented by 

Actually, to be precise, the graph representation of an ECGF-sentence is a forest as the conjunction of two ECGF-sentences is an ECGF-sentence (Definition 2.9.2-ii). However, we can easily eliminate non-guarded conjunctions of ECGF-sentences by a linear rewriting: let Δ_1 and $\Delta_2 = \exists \vec{x}(Guard \wedge \delta)$ be two ECGF-sentences such that $\Delta = \Delta_1 \wedge \Delta_2$ is pure, Δ is equivalent to $\exists \vec{x}(Guard \wedge \Delta_1 \wedge \delta)$.

$M(\varphi) \models \psi$ holds iff there exists a substitution s from the variables in ψ to the domain of $M(\varphi)$: i.e., $s : \{x, y, z\} \rightarrow \{a, b\}$ such that, under s , the set of atoms in ψ is included in $M(\varphi)$ (i.e., $\{Pxy, Rxyy, Rzxy\}[s] \subseteq M(\varphi)$).

The algorithm uses the tree-structure of the guarded conclusion sentence to verify recursively the existence of such a substitution:

substitution(M, ψ):

Input: M and $\psi = \exists \vec{x}(\alpha \wedge \varphi_1 \wedge \dots \wedge \varphi_n)$

Output: a set S of substitutions from X , the set of variables in α , to D , the domain of M .

$S = \{s/s : X \rightarrow D \text{ and candidate}(s) \text{ and neighbour-agreement}(s)\}$ where

(i) s is a candidate if the model satisfies the guard α under the substitution s : i.e., $\alpha[s] \in M$.

(ii) s agrees with the neighbourhood of the guard if for every successor $\varphi_{i/1 \leq i \leq n}$ of α , s agrees with some substitution s_i chosen for φ_i on their shared domain: i.e., $\forall 1 \leq i \leq n, \exists s_i \in \text{substitution}(M, \varphi_i) / x \in (\text{domain}(s) \cap \text{domain}(s_i))$ implies that $s(x) = s_i(x)$.

$M(\varphi) = \{Paa, Raaa, Pab, Rbaa, Rbab\} \models^? \psi$. The recursive launching of the function *substitution* stops at the leaves:

- $\text{substitution}(M(\varphi), Rxyy) = \{(x/a, y/a), (x/b, y/a)\}$ as $Rxyy$ can be evaluated at both –and only– $Raaa$ and $Rbaa$.

- $\text{substitution}(M(\varphi), Rzxy) = \{(x/a, y/a, z/a), (x/a, y/a, z/b), (x/a, y/b, z/b)\}$.

The two leaves are treated independently despite their shared variables, however, the resulting substitutions are confronted at the level of their guard as follows.

For the root, Pxy , candidate substitutions correspond to the atoms Paa and Pab . Hence, $(x/a, y/a)$ and $(x/a, y/b)$ are pre-selected and confronted to the neighbourhood-agreement criterion:

- $(x/a, y/b)$ agrees with none of the substitutions selected for the successors of the guard and therefore is rejected.

- on the other hand, $(x/a, y/a)$ agrees with at least one substitution for each successor and therefore, it can be selected at the level of the guard.

$\text{substitution}(M(\varphi), \psi = \{(x/a, y/a)\})$ is not empty, thus, it holds that $M(\varphi) \models \psi$. Indeed, the atoms of the conclusion are satisfied by the minimal model of the premiss under two substitutions extending $(x/a, y/a)$ that could have been recorded along the recursive steps:

$$\{Pxy, Rxyy, Rzxy\}[x/a, y/a, z/a] \subseteq \{Paa, Raaa, Pab, Rbaa, Rbab\} \text{ and}$$

$$\{Pxy, Rxyy, Rzxy\}[x/a, y/a, z/b] \subseteq \{Paa, Raaa, Pab, Rbaa, Rbab\}.$$

2.9.1.1 Note on the complexity of the algorithm

the minimal model of the premiss can be encoded (see Chapter 2.2.3) as a binary string of size polynomial in $n \in \mathcal{O}(r * d^k)$ where d is the size of the domain, r the number of relation symbols and k their maximal arity. We can assume that every relation symbol occurring in the conclusion formula has an interpretation in the model (a simple polynomial time check can rest us) and therefore, the maximal arity of relation symbols in the conclusion is bound by k .

Applying the algorithm as it is may require exponential time. However, in the algorithm proposed in Chapter 3, the recursive calls of the function *substitution* are replaced by a single run through the syntactic tree from the leaves to the root. A set of substitutions is recorded for each atom of the conclusion formula. The number of substitutions in these sets is bound by the number of tuples occurring in the interpretation of relation symbols –i.e., the number of 1's in the encoding of the model or seen from the point of view of the consequence problem, the number of atoms in the premiss formula–; hence, bound by n . If an order is chosen for the v variables of the conclusion, each substitution can be encoded as a binary string of size $k * \log v * \log d$, a size polynomial in the size of the input. Let m be the number of atoms occurring in the conclusion formula, for each of the m nodes in the conclusion tree, at most n polynomial size substitutions are pre-selected in time $\mathcal{O}(n * k)$. Each of them is compared to the selection made for the (at most) m successors of the node at stake. For each pair of substitutions, their

agreement depends on at most k shared variables. Hence, the total time of the algorithm is polynomial in the combined size of the (encoding) of the model and the conclusion formula.

The salience of the “single guard property” to the polynomial time complexity of consequence in ECGF is revealed by examining a wider fragment: the loosely guarded fragment of FOL (LGF) [Ben97] generalises GF by allowing conjunctions of atoms of a special form as guard. LGF is also decidable for satisfiability [Ben97] (2-EXP-complete [Grä99]) and PSPACE-complete for model checking (Chapter 2.4.2). ECLGF, the restriction of LGF to existential conjunctive FOL is defined inductively by:

2.9.7. DEFINITION. (ECLGF)

- Every atomic formula belongs to ECLGF.
- ECLGF is closed under conjunction.
- If (i) $\alpha = \alpha_1 \wedge \dots \wedge \alpha_m$ is a conjunction of atoms and
 - (ii) $\varphi = \varphi_1 \wedge \dots \wedge \varphi_n$ is a (possibly empty) conjunction of formulae in ECLGF and
 - (iii) for every $i/1 \leq i \leq n$, $free(\varphi_i) \subseteq free(\alpha)$ and
 - (iv) $\vec{x} \subseteq free(\alpha)$ and
 - (v) for every $x \in \vec{x}$ and every other variable $y \in free(\alpha)$, there is at least one atom $\alpha_{i_1 \leq i \leq m}$ that contains both x and y ,
 then $\exists \vec{x}(\alpha \wedge \varphi)$ is in ECLGF.

ECLGF is a typical language for cliques. For instance,

$$\exists x_1 \exists x_2 \exists x_3 \underbrace{((Ex_1x_2 \wedge Ex_1x_3 \wedge Ex_2x_3))}_{\text{loose-guards}} \wedge \underbrace{(Ex_2x_1 \wedge Ex_3x_1 \wedge Ex_3x_2)}_{\text{symmetry of edges}}$$

expresses a clique with three nodes (E represents the edge relation between nodes).

2.9.8. COROLLARY. *If A is a sentence in $FOL_{\{\exists, \wedge\}}$ and B is an ECLGF-sentence, then $A \models B$ is NP-complete.*

Proof: The proposition is a direct consequence of the NP-completeness of the CLIQUE problem (Theorem 2.2.5) and the non-deterministic polynomial time algorithm for consequence in $FOL_{\{\exists, \wedge\}}$ (Theorem 2.5.5). The recursive function *substitution* can easily be adapted to ECLGF by representing a formula as a graph in which two nodes (atoms) are connected if they share a variable. However, if this graph is cyclic –e.g., the previous clique–, the recursive calls of the function lead to a deterministic exponential time algorithm. ■

This application of modal ideas to the structure of first-order formulae demonstrates several points. Firstly, as already noticed, tree like structures can often be associated to efficient computation. Secondly, how thin is the frontier between tractability and untractability is highlighted by the subtle difference between guards and loose guards. Finally, predicate logic, modal logic and graph formalisms can benefit from a multicultural symbiosis. In particular, the application of a guarded syntax to simple conceptual graphs combined with graph homomorphism techniques will be one of our main result in the next chapter. Now, to learn more about this symbiosis, it is interesting to consider further correspondences between classical and modal languages.

2.9.2 The modal cousin of $FOL_{\{\exists, \wedge\}}$

The modal operator corresponding to existential quantification in first-order logic is the diamond: $\diamond\varphi$ is true at a world w of a model M if there exists some world w' in M such that w' is accessible from w and φ is true at w' .

2.9.9. THEOREM. *For $ML_{\{\diamond, \wedge\}}$, satisfiability is trivial.*

Model checking, model equivalence and consequence are in polynomial time.

Proof: In the absence of negation, any modal formula is satisfiable (in a model with a single world). Complexity results for model checking and model equivalence inherit the upper bound of the richer logic $K = ML_{\{\diamond, \wedge, \neg\}}$ [HV91, Gro96]. For consequence, we can use the fact that a sentence in $ML_{\{\diamond, \wedge\}}$ translates into a formula (with at most two variables and at most one free variable) of the existential conjunctive guarded fragment which has a polynomial time consequence problem (Proposition 2.9.4):

$$\begin{aligned} ST_x(\diamond\varphi) &:= \exists y(Rxy \wedge ST_y(\varphi)), \\ ST_x(\varphi \wedge \psi) &:= ST_x(\varphi) \wedge ST_x(\psi) \text{ and} \\ ST_x(P) &:= Px. \blacksquare \end{aligned}$$

We can also note that it follows from the standard translation and Theorem 2.5.3 that complexity preserving equivalences between consequence, model checking and model comparison, also hold in $ML_{\{\diamond, \wedge\}}$.

By relying on a guarded use of existential quantification, consequence in $ML_{\{\diamond, \wedge\}}$ is tractable. What would be the effect of adding atomic negation to the language?

2.9.3 Atomic negation at no cost

2.9.10. PROPOSITION. *For $ML_{\{\diamond, \wedge, \neg_a\}}$, consequence is in P .*

Proof: We can use the tractability result of Theorem 2.9.9 by translating negated propositions to new positive ones. Let φ and ψ be formulae in $ML_{\{\diamond, \wedge, \neg_a\}}$. Without loss of generality, assume that φ is satisfiable (the case of an unsatisfiable formula can be solved by a pretreatment checking for contradictory literals). Let ϕ^+ denote the formula ϕ in which every negated proposition $\neg_a P$ has been replaced by a proposition P^- not occurring in ϕ . By induction, it is easy to prove that $\varphi \models \psi$ iff $\varphi^+ \models \psi^+$ and the later is an instance of the polynomial time problem, $Consequence(ML_{\{\diamond, \wedge\}})$. ■

Contrary to existential conjunctive FOL with atomic negation, $ML_{\{\exists, \wedge, \neg_a\}}$ enjoys together the guarded criterion and a minimal model (of reasonable size) property, two recurrent characteristics of tractable fragments.

A step further in combining classical and modal logics, is the introduction of the first-order machinery for manipulating variables into a modal language. The result is an hybrid language.

2.9.4 Back to untractability

The price to pay for the introduction of first-order logic freedom with variable manipulation is no surprise:

2.9.11. PROPOSITION. *Consequence($ML_{\{\diamond, \wedge, N, @, \downarrow\}}$) is NP-complete.*

Proof: a formula in $ML_{\{\diamond, \wedge, N, @, \downarrow\}}$ enjoys a minimal model whose size is polynomially related to the formula: a first-order structure with a designated object in which each diamond generates a single accessible new world. If we guess a substitution for the nominal variable of the conclusion into the universe of the minimal model of the premiss, the verification that the conclusion holds in this model takes polynomial time. For the NP-hardness, we can again reduce CLIQUE in polynomial time: $A \models \downarrow x_1 \dots x_k (\bigwedge_{1 \leq i < j \leq k} @_{x_i} \diamond x_j)$ iff A represents a graph containing a clique of size k . ■

Standard translation provides another perspective on the consequence problem: it boils down to $Consequence(FOL_{\{\exists, \wedge\}})$ as the \downarrow binder enables to introduce non guarded variable by marking worlds to which we may later come back. For instance, $ST_x(\downarrow z(\diamond_1 \diamond_1 z)) \equiv \exists y(R_1(z, y) \wedge \exists x(R_1(y, x) \wedge R_1(x, z)))$ is not guarded.

In contrast to FOL, the full language of modal logic remains decidable and of modest complexity:

2.9.12. THEOREM. *Satisfiability($ML_{\{\diamond, \wedge, \neg\}}$) and Consequence($ML_{\{\diamond, \wedge, \neg\}}$) are PSPACE-complete.*

Proof by reduction of validity of quantified boolean formulae and a PSPACE tableau algorithm to explore economically the possibly exponential model can be found in [Lad77] (modal logic K), [HM92] (multi-modal logic K) and [SSS91] (description logic \mathcal{ALC}).

[ABM00] shows that the further addition of nominals and the jump operator @ does not increase the difficulty of satisfiability (i.e., $Satisfiability(ML_{\{\diamond, \wedge, \neg, N, @\}})$ is PSPACE-complete) whereas the \downarrow -binder for nominal variables can have a dangerous effect: $Satisfiability(ML_{\{\wedge, \neg, \diamond, \downarrow\}})$ is undecidable [ABM99a]; see also [ABM99b] and [Are00] for a complexity study of several hybrid fragments.

2.9.5 Finite bounds

By analogy to the first-order logic case, finite bounds on the modal languages may also ease decision problems. For instance, [Hal95] shows that a fixed bound on the modal depth (the maximal nesting of modalities) makes satisfiability be NP-complete. If this modal depth bound is cumulated with a finite bound on the number of propositional variables, then the problem can be decided in linear time. It is however surprising to note that, when a finite bound on the number of propositional variables is fixed, the complexity of satisfiability in K (multi-modal or not) remains PSPACE-complete (one propositional symbol is sufficient to obtain the polynomial space lower bound) [Hal95].

2.9.6 Description logics

Modal languages are not the only way of carving up simple fragments of first-order logic. Another well-established tradition are *description logics*. These occur implicitly in many parts of this dissertation, but we record a few salient facts here.

Description logics arose in the KL-ONE system [BS85] from the study of the representation of structured knowledge in semantic networks (see e.g., the synthesis in [Leh92]) and frame systems (e.g. [Min74]). Even though the semantics of description logics is often provided as an interpretation in first-order models or as a translation to first-order logic, Schild [Sch91] identified strong semantical connections between modal logics and the core language of description logics. Indeed, the applications of description logics to specific knowledge representation problems have given rise to constructors which are not typically considered in (core language of the) modal logics: e.g., so-called assertional knowledge about individuals, complex relational constructors or counting operators. Bridging these expressive, application driven, extensions has been obtained by considering hybrid modal formalisms [Are00].

Nonetheless, it remains that, in their seek for expressive, while efficient, reasoning systems, description logics have adopted the locality principle of modal interpretation.

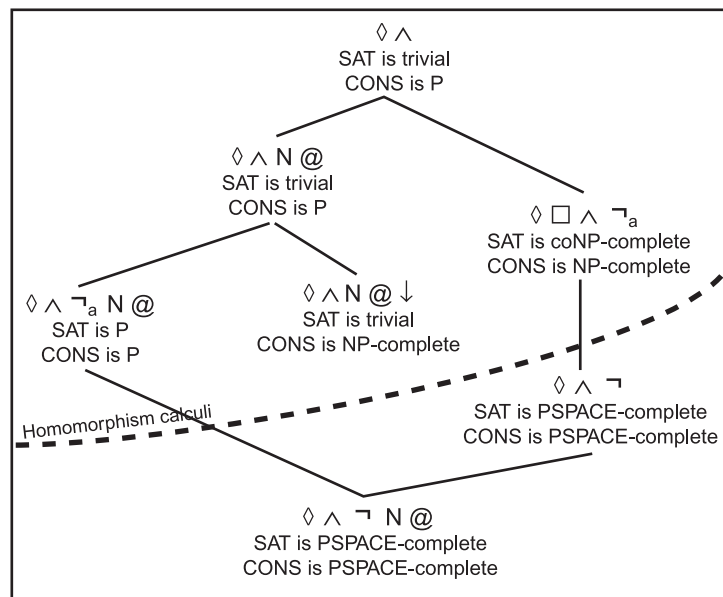
Another interesting feature is the modularity of description logic languages. The extensive study of different syntactic fragments adapted to specific applications and of associated (often modal tableau-based) optimal calculi, provides a general map of the complexity landscape for reasoning with description logics; we refer the interested reader to the detailed synthesis in [BBH⁺90],[DLNS94] and [DLNN97].

Following the route traced by KL-ONE for describing rules and “universal knowledge”, many description logics include universal quantification or its modal counter-part, box-modalities. However, a recent bifurcation to a focus on “factual knowledge” expressed by existential quantification (and diamond-modalities) has brought to the fore the use of the underlying tree structures of models for efficient homomorphism-based calculi: see e.g., [DLN⁺92] and [BKM99].

This use of graph representation of knowledge and graph-based deduction methods to efficiently compute logical reasoning, has naturally lead to bridges between conceptual graph and description logic formalisms. [BMT99] proposes a first step in this direction, by constraining simple conceptual graphs to tree descriptions of the existential conjunctive \mathcal{ELIRO}^1 description logic. We will further explore this interaction in our modal treatment of simple and nested conceptual graphs in Chapter 4.3.

We refrain from giving an extensive complexity analysis here, referring the reader to [Are00] for a bridge to modal languages of various sorts, which makes many of our earlier points applicable mutatis mutandis.

2.9.7 A landscape of modal complexity results



The main message of this quick overview of complexity results in modal logics is that the locality principle of modal semantics has a benefic effect on the chosen benchmark problems: first, model checking and model comparison are tractable. But also, on the basis of the correspondence between \diamond and existential quantification, satisfiability and consequence are often easier (and never more difficult) in modal fragments than in the corresponding first-order fragments.

Finally, the study of the modal paradigm is twofold. First, some strengths of modal formalisms can directly be exploited in first-order formalisms, witness the notion of guards that we will further develop in the next chapter. But of course, the modal view on interpretation can directly be applied to graph-based systems. This will be the chosen path in Chapter 4.3. We will also take up the question of combining these two perspectives in predicate modal formalisms.

2.10 Conclusions

With the development of automatised reasoning systems, the search for tractable problems has become a crucial point of the agenda identified in [LB87] as the tradeoff between expressivity and tractability in knowledge representation.

By “projecting” graph-like reasoning systems on usual formal logical systems, we have achieved a finely-structured view of complexity for the former ones. We are now equipped to exploit, in a controlled way, their analogies to modal and first-order fragments.

Of course, this bridge is useful as such, as we can apply it to get new tractable conceptual graph systems. But also, vice versa, it suggests graphical aspects to existing weak linguistic formalisms that are often neglected by logical studies.

Also, by bringing to the fore several benchmark problems that have a role to play in the comparison of logics, we have attempted to escape the yoke of the traditional (in logic) and unidirectional focus on satisfiability. Indeed, the chosen remaining problems are not only salient in practical applications of logics, but also are interesting for their evolution with language changes; i.e., the problems appear to follow quite independent complexity paths with identified crossroads. However, such choices are not the only relevant ones. For instance, to be fair to applications of logics to automated reasoning, we could as well consider average complexity analysis. Leaving such further explorations to perspectives, we nevertheless regard our present “worst-case” analysis as a necessary first step to understanding logical and computational grounds in graph-based reasoning.

The eventual issue is not a list of separate complexity results, but understanding the general *parameters* that affect complexity and in particular, the overstepping of low complexity bounds such as tractability. Whether the parameters that we found here (e.g., tree-like structures, direct mapping between formulae and their minimal model, etc.) are *visually relevant* is something we leave for discussion in our concluding chapter.

Strong of our recent experience with fragments of formal languages, we can now undertake the study of specific conceptual graph systems and observe how the symbiosis of logics and techniques can pay off to devise expressive and efficient fragments of the graph formalism. A natural starting point is the central language of simple conceptual graphs.

Chapter 3

Positive information

In representing positive factual information, simple conceptual graphs achieve a symbiosis of the previously discussed cognitive and computational aspects.

Among the cognitive properties of drawings, we have seen that their faithfulness to what they represent is one of their preponderant positive characteristics. This aspect is salient in simple conceptual graphs. Not only the interpretation of a simple graph drawing can be described by the slogan “find the direct mapping of the picture in the formal model”, but also simple conceptual graphs enjoy a one-to-one correspondence between a graph and its canonical structure. From the previous chapter, we know that such a correspondence propagates to complexity results under the form of a collapse between three of our benchmark computation problems. Hence, the interlaced effect of computational and cognitive aspects for simple graph drawings.

In order to provide a careful complexity analysis of logical reasoning with simple conceptual graphs, we first need to expose the formal syntax and semantics of these graphs. This will form the first part of the chapter.

In a second part, projection will be defined. It is a consequence calculus based on labelled graph homomorphism, which is proved complete with respect to the semantics. But, consequence, a central problem in knowledge base managing and querying, is known untractable in the equivalent fragment of FOL. Therefore, it becomes vital for the tractability of automated reasoning to look at more constrained fragments. A way to do so is to consider syntactical constraints on the language, without changing the semantics. We present the known tractable restriction of projection to graphs having a tree structure. By language correspondence, another perspective on the simple conceptual tree fragment is offered: the one, imported from modal logic, of guarded sentences.

Finally, in the last part, the tractability result is extended to a wider fragment by careful enlarging of the power of guards. The proved polynomial complexity of a projection algorithm on this new fragment will constitute the main result of the chapter.

This chapter partly overlaps with the previous discussion on tractability in Chapter 2. What follows is the extensive version of our lighter treatment of tree-properties and guards for existential conjunctive FOL.

3.1 The cornerstone: simple conceptual graphs

Departing from formal logical languages and adopting features of graph theory, simple conceptual graphs (SCG), introduced by Sowa in [Sow84], have captured the attention of computer scientists more than of logicians, which often prefer more expressive formalisms such as the one including Peirce's negation boxes.

Nonetheless, SCGs are the basic and inescapable components in the puzzle of the different languages considered in this thesis. Hence, they deserve a detailed presentation and an analysis of their complexity behaviour on logical problems.

Just as any other formal language, the syntax of the graphs rests on a predefined choice of vocabulary. For their concern to the classification of knowledge, conceptual graph formalisms have adopted the notion of hierarchical vocabulary, the so-called ontologies in artificial intelligence.

3.1.1 Language signature

The signature of a conceptual graph language represents an ontology of a specific application domain. Classification is the recurring message as the different language items of an alphabet are organised in a set of hierarchies forming, as a whole, a support.

3.1.1. DEFINITION. [Alphabet] An alphabet is a quadruple $(\mathcal{I}, \mathcal{C}, \mathcal{R}, \text{arity})$ where \mathcal{I} , \mathcal{C} and \mathcal{R} are pairwise disjoint enumerable non-empty sets. \mathcal{I} is a set of object names, \mathcal{C} is a set of concept names and \mathcal{R} is a set of relation names. $\text{arity} : \mathcal{R} \rightarrow \mathbf{IN}^+$ is an arity function for relation names which partitions the set \mathcal{R} into a family $(\mathcal{R}_n)_{n \in \text{arity}(\mathcal{R})}$ such that $\forall r \in \mathcal{R}, r \in \mathcal{R}_n$ iff $\text{arity}(r) = n$. Furthermore, there are two distinguished concept names, the *unit concept* $c^\top \in \mathcal{C}$ and the *zero concept* $c^\perp \in \mathcal{C}$, and for every $\mathcal{R}_n \subseteq \mathcal{R}$, there are two distinguished relation names, the *unit n -ary relation* $r_n^\top \in \mathcal{R}_n$ and the *zero n -ary relation* $r_n^\perp \in \mathcal{R}_n$.

A support describes some hierarchical knowledge on (part of) an alphabet.

3.1.2. DEFINITION. [Support] A support over an alphabet $\mathcal{A} = (\mathcal{I}, \mathcal{C}, \mathcal{R}, \text{arity})$ is a pair $((\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}))$ where $(\mathcal{C}, \leq_{\mathcal{C}})$ is a \wedge -semilattice of concept names and $(\mathcal{R}, \leq_{\mathcal{R}})$ is an ordered set of relation names partitioned by arity such that:

1. $\mathcal{C} \subseteq \mathcal{C}$ and $(\mathcal{C}, \leq_{\mathcal{C}})$ is \wedge -semilattice with supremum c^\top (i.e. $\forall c \in \mathcal{C}, c \leq_{\mathcal{C}} c^\top$) and zero c^\perp (i.e. $\bigwedge_{\mathcal{C}} \mathcal{C} = c^\perp$)

2. $R \subseteq \mathcal{R}$ and $(R, \leq_R) = \bigcup_{i \in \text{arity}(R)} (R_i, \leq_{R_i})$ where for all $i \in \text{arity}(R)$, $R_i = \mathcal{R}_i \cap R$ and (R_i, \leq_{R_i}) is an ordered set of relation names of arity i with supremum r_i^\top and infimum r_i^\perp (i.e. $\forall r \in R_i, r_i^\perp \leq_R r \leq_R r_i^\top$)

A support $((C, \leq_C), (R, \leq_R))$ is called *finite* if both C and R are finite.

It follows from the definition that \leq_R is an order on R such that two relations of different arity are incomparable; i.e., $\forall i, j \in \text{arity}(R), \forall r \in R_i, \forall r' \in R_j, (i \neq j$ implies that neither $r \leq_R r'$ nor $r' \leq_R r$). The intuitive meaning of $c_1 \leq_C c_2$ is that every object belonging to the extension of the concept $c_1 \in C$ also belongs to the extension of the concept $c_2 \in C$. Similarly, for two n -ary relations in R , r_1 and r_2 , the intuitive meaning of $r_1 \leq_R r_2$ is $\forall x_1, \dots, x_n [r_1(x_1, \dots, x_n) \rightarrow r_2(x_1, \dots, x_n)]$.

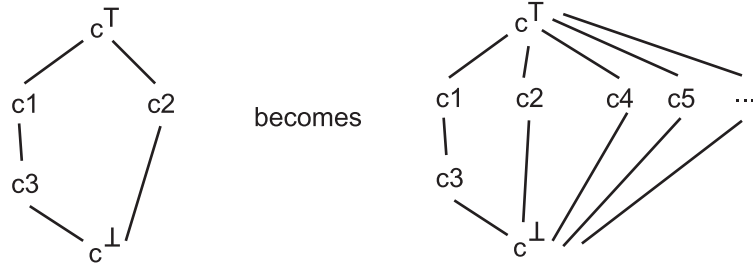
The signature of a conceptual graph language is an alphabet ordered by a support.

3.1.3. DEFINITION. [Signature] The signature of a conceptual graph language is a tuple $(\mathcal{I}, (\mathcal{C}, \leq_C), (\mathcal{R}, \leq_R), \text{arity})$ such that $\mathcal{A} = (\mathcal{I}, \mathcal{C}, \mathcal{R}, \text{arity})$ is an alphabet and $((\mathcal{C}, \leq_C), (\mathcal{R}, \leq_R))$ is a support over \mathcal{A} .

From a practical point of view, the hierarchical information conveyed by a signature is often specified and stored as a finite support which is extended into a signature in the following way. A signature is obtained from a support by a closure over the vocabulary of the concerned alphabet. Elements which are not contained in the original support become pairwise incomparable in the resulting signature.

Given a support $\sigma_{\mathcal{A}} = ((C, \leq_C), (R, \leq_R))$ over an alphabet $\mathcal{A} = (\mathcal{I}, \mathcal{C}, \mathcal{R}, \text{arity})$, $\Sigma_{\sigma_{\mathcal{A}}}$ is the tuple $(\mathcal{I}, (\mathcal{C}, \leq_C), (\mathcal{R}, \leq_R), \text{arity})$ such that $((\mathcal{C}, \leq_C), (\mathcal{R}, \leq_R))$ uniquely extends the support $((C, \leq_C), (R, \leq_R))$ by:

1. $\forall (c, c') \in \mathcal{C} \times \mathcal{C}, c \leq_C c'$ iff (i) $(c, c') \in C \times C$ and $c \leq_C c'$ or (ii) $c = c'$ or (iii) $c' = c^\top$ or (iv) $c = c^\perp$,



For example,

2. $\forall (r, r') \in \mathcal{R} \times \mathcal{R}, r \leq_R r'$ iff (i) $(r, r') \in R \times R$ and $r \leq_R r'$ or (ii) $r = r'$ or (iii) $r' = r_{\text{arity}(r)}^\top$ or (iv) $r = r_{\text{arity}(r')}^\perp$.

3.1.4. PROPOSITION. Given a support $\sigma_{\mathcal{A}} = ((C, \leq_C), (R, \leq_R))$ over an alphabet $\mathcal{A} = (\mathcal{I}, \mathcal{C}, \mathcal{R}, \text{arity})$, $\Sigma_{\sigma_{\mathcal{A}}}$ is a signature.

Proof: We need to verify that a \wedge -semilattice of concept names with supremum C^\top is constructed. Obviously, \leq_C is an order on C (it is the order \leq_C). For two concept names $c \in (C \setminus C)$ and $c' \in C$,

- reflexive: $c = c'$ implies that $c \leq_C c'$ by construction,
- antisymmetric: $c \neq c'$ and $c \leq_C c'$ implies that $c' \not\leq_C c$. Indeed, $c^\top \not\leq_C c$ and $c \not\leq_C c^\perp$ and c is \leq_C -incomparable with other concept names than the zero concept and the unit concept.
- transitive: c is only comparable to c^\perp and c^\top , and $c^\perp \leq_C c^\top$.

Hence, \leq_C is an order on C with infimum c^\perp and supremum c^\top . For the semilattice condition, it holds that for two concept names $c \in (C \setminus C)$ and $c' \in C$, the non-empty set $\{x \in C/x \leq_C c \text{ and } x \leq_C c'\} \subseteq \{c, c^\perp\}$ and $c^\perp \leq_C c$.

Similarly for each set of relations $\mathcal{R}_i \subseteq \mathcal{R}$, $\leq_{\mathcal{R}_i}$ is an order with infimum r_i^\perp and supremum r_i^\top . ■

The semilattice condition on concept names is sometimes relaxed to an order (e.g. [CM92]) or strengthened to a complete lattice [GW99]. Our intermediate choice is semantically motivated: the assertion that an object belongs to the extension of two distinct concept names will be equivalent to the assertion that the same object belongs to the extension of the meet of those concept names. It will therefore prove convenient to guarantee the existence of the meet of two concept names, which denotes the intersection of the extensions of those two concept names. It should be noted that an ordered set (C, \leq_C) can easily be embedded into a \wedge -semilattice by insertion of eventual “missing meets”. On the other hand, contrary to Formal Concept Analysis [GW99] which is devoted to the extraction (and manipulation) of concept name lattices from structures, the join operation on concept names will find no use in the conceptual graph systems defined in this thesis.

Another difference with some other notions of support in the literature resides in the absence of predefined association of a concept name to every object name. We will see with the homomorphism calculus that such a constraint would imply special care for the isolated terms (i.e, terms that occur in a graph but not as argument of a relation).

Now that we have defined how a vocabulary is organised into an hierarchy, we can proceed to the assembling of symbols into simple conceptual graphs. In order to provide a systematic description of the drawings that can later be formally exploited for semantical and deductive purposes, a sensible style is to use the mathematical way of representing the abstract objects, graphs. Of course, these symbolic and textual notations may seem cumbersome when compared to the related drawings, but this is a necessary step for our subsequent complexity analysis in the framework of symbolic complexity theory. Furthermore, by defining simple conceptual graphs as abstract symbolic objects, we enrich our ordnance with an other symbolic language, which comes in addition to the previous translations to formal languages of logic.

3.1.2 Syntax

3.1.5. DEFINITION. [Simple conceptual graph] A simple conceptual graph (SCG) over a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_R), \text{arity})$ is a finite, directed and bipartite multigraph $G = (R, C, E, \text{label}, \text{co})$ where R denotes the set of relation vertices, C the set of concept vertices and E the set of edges.

It is a *multigraph* as more than one edge may connect two vertices, *bipartite* as concept and relation vertices alternate and *directed* as $E \subseteq (R \times C \times \mathbb{N}^+)$; the third parameter provides a total order on the edges incident to a relation vertex, enabling to distinguish two distinct edges connecting the same two vertices.

label is a mapping from vertices to names such that:

- (i) $\forall c \in C, \text{label}(c) = (\text{type}(c), \text{marker}(c)) \in \mathcal{C} \times (\mathcal{I} \cup \{*\})$ and
- (ii) $\forall r \in R, \exists k \in \mathbb{N}^+$ such that $\text{label}(r) \in \mathcal{R}_k, |\{(r, c, i)/(r, c, i) \in E\}| = k$ and $\{i/(r, c, i) \in E\} = \{1, \dots, i\}$ (i.e., there are k edges incident to r).

C_* denotes the set of concept nodes labelled with the marker $*$ and co is an equivalence relation on C_* .

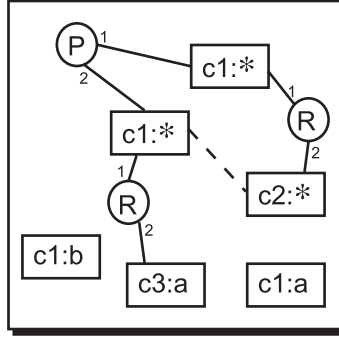


Figure 3.1: A simple conceptual graph

For instance, the graph in Figure 3.1 represents the information “ a is of type $c1$ and $c3$ (i.e., “the denotation of a belongs to both the extension of the concept $c1$ and the extension of the concept $c3$ ”) and there is something of type $c1$ and $c2$ which is in relation R with a and also in both relations R and P with something of type $c1$ ”. As already seen, a possible translation of the graph into a language of existential conjunctive FOL is “ $c1(b) \wedge c3(a) \wedge c1(a) \wedge \exists x(c1(x) \wedge c2(x) \wedge R(x, a) \wedge \exists y(c1(y) \wedge R(y, x) \wedge P(y, x)))$ ”.

3.1.2.1 Notations

- We note $r(i)$ the i^{th} neighbour of r ; i.e. $r(i) = c$ such that $(r, c, i) \in E$.
- The size $|G|$ of a simple graph G is equal to $|C| + |R| + |E|$.

- Furthermore, the special simple graph $G_\emptyset = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ is called the *empty graph* and $SCG(\Sigma)$ denotes the set of all (finite) simple graphs with respect to a signature Σ .
- In order to lighten the drawings, an equivalence relation on a set of concept nodes is represented by an undirected graph covering the relation: $c \equiv_{co} c'$ iff there is a dashed path between c and c' . For later manipulations of equivalence relations, it is useful to fix some related notations: given an equivalence relation R on a set X and $x \in X$, $class_R(x)$ is the R -equivalence class of x (i.e. $class_R(x) = \{y \in X / (x, y) \in R\}$) and $class_R(X)$ is the set of R -equivalence classes (i.e. $class_R(X) = \{class_R(x) / x \in X\}$).
- It should be emphasised that the adopted set notation of graphs presupposes that if $G^1 = (R, C, E, label, co)$ and $G^2 = (R', C', E', label', co')$ are two disjoint graphs, then $R \cap R' = C \cap C' = \emptyset$.

The graphs would make no sense if we would not be able to understand them. So far, in the many previous examples, we have seen two ways of interpreting the simple conceptual graph drawings: an intuitive description in natural language of the represented information and a translation into the usual language of first-order logic. It is now time to formalise the former. We will proceed by interpreting the graphs into structures of the later; i.e., first-order structures.

3.1.3 Semantics

Following the work of Sowa [Sow84], in most of the conceptual graph literature, SCGs are given a meaning by translating them to existential conjunctive first-order formulae. We propose, in the line of [Ker96], [Pre98] or [Min00], to stride FOL semantics and directly interpret our graphs into classical structures. By this way, the structural connections between graphs and models are better rendered.

3.1.3.1 Structures

The hierarchical knowledge associated to a conceptual graph vocabulary is not part of the graphs, but an a priori information we have when the representations are built, communicated or simply, perceived. This classification should therefore be contained in the underlying structures used for interpretation.

3.1.6. DEFINITION. [Σ -structure]

For a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$, a Σ -structure is a pair (D, F) where the domain D is a set of objects and the interpretation function F is partial on the set of object names (or equivalently, there is an “undefined object”, $\odot \notin D$)

$$\text{and } F : \begin{cases} \mathcal{I} \rightarrow D \cup \{\odot\} \\ \mathcal{C} \rightarrow \mathcal{P}(D) \\ \forall \mathcal{R}_n \subseteq \mathcal{R}, \mathcal{R}_n \rightarrow 2^{D^n} \end{cases}$$

such that the ordering conveyed by the signature is respected:

- $F(c^\top) = D$ and $\forall c, c' \in \mathcal{C}, F(c) \cap F(c') = F(c \wedge_{\mathcal{C}} c')$
- $\forall r, r' \in \mathcal{R}_n, r \leq_{\mathcal{R}} r'$ implies that $F(r) \subseteq F(r')$.

For a concept c , $F(c)$ is called the extension of the concept c or the c -subdomain. Note that a subdomain or even the whole domain can be empty.

A first step in relating a graph to a structure is to assign object nodes of the former to objects of the later. Coming back to our embedding slogan, assignments would correspond to “find the anchoring points of the picture in the model”.

3.1.3.2 Assignments

An assignment is a mapping from the concept nodes of a simple graph to the domain of a structure. It extends the interpretation function and maps two coreferent nodes to the same domain object.

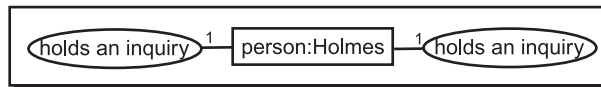
3.1.7. DEFINITION. [Assignment]

For a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}), \text{arity})$, a SCG $G = (C, R, E, \text{label}, \text{co})$ over Σ and a Σ -structure $M = (D, F)$, an assignment is a function $f : C \rightarrow D$ such that

- $\forall c \in C, f(c) \in F(\text{type}(c))$ and if $\text{marker}(c) \in \mathcal{I}$ then $f(c) = F(\text{marker}(c))$ and
- $\forall c, c' \in C_*$, if $c \equiv_{\text{co}} c'$ then $f(c) = f(c')$.

As a consequence, if $\exists c \in C$ such that $F(\text{marker}(c)) = \odot$ then there is no assignment of the graph in M .

As for the classical interpretation of a textual first-order language, two occurrences of a constant (two concept nodes labelled with the same object name) are mapped on the same object of a structure. It should however be noted that this choice is not the only natural one in the case of conceptual graphs: a textual language is usually not equipped for distinguishing a single occurrence of a constant which is an argument of different relation symbols, from distinct occurrences of the same constant. For instance, the two different graphs:



and



have a single textual counterpart:

$$Person(Holmes) \wedge HoldsInquiry(Holmes) \wedge Person(Holmes) \wedge HoldsInquiry(Holmes)$$

With a more liberal interpretation of proper names, the first graph could convey some redundant information about the activity of a single person named Holmes, whereas the second graph could represent a gathering of information about the detective Sherlock Holmes and about the US Supreme Court Justice, Holmes. In order to obtain a straight forward semantical correspondence between graphs and textual formulae, the choice of forcing an object name to denote at most¹ one object in the domain of a structure is made.

Finally, we come to the last ingredient of the interpretation: the verification that the relational network connecting the anchoring points (i.e., the concept nodes) coincides with the one linking the corresponding objects in the structure.

3.1.3.3 Truth definition for SCGs

3.1.8. DEFINITION. [Truth of a SCG] Let $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$ be a signature, $G = (C, R, E, label, co)$ be a SCG over Σ and $M = (D, F)$ be a Σ -structure.

- G is true in M under an assignment f (noted $M, f \models G$)
iff $\begin{cases} \forall c \in C, f(c) \in F(type(c)) \text{ and} \\ \forall r \in R, label(r) \in \mathcal{R}_n \Rightarrow \langle f(r(1)), \dots, f(r(n)) \rangle \in F(label(r)) \end{cases}$
- G is true in M , noted $M \models G$, iff there exists an assignment f such that $M, f \models G$.
- H is a consequence of G (or H subsumes G), noted $G \sqsubseteq H$, iff H is true in every Σ -structure in which G is true.
- H is equivalent to G , noted $H \equiv G$, iff $G \sqsubseteq H$ and $H \sqsubseteq G$.

The empty graph corresponds to the logical constant *True* of a textual language as it is verified by any structure; it is in fact the only valid simple graph.

In conceptual graphs, terms are proper formulae: an isolated concept node is a simple graph denoting the non-emptiness of a concept. Indeed, if every object name was interpreted as an object in the domain of a structure, then any graph consisting of an isolated concept node labelled with the unit concept (i.e., $\boxed{c^{\top} : m}$)

¹The importance of the undefined object as possible denotation for a name will become clear from the interpretation of isolated concept nodes and their role in homomorphism calculi.

for $m \in \mathcal{I} \cup \{*\}$) would be equivalent to the empty graph as $\mathcal{I} \neq \emptyset$ and $F(c^\top) = D$, whereas the two graphs are intended to represent different pieces of information: the empty graph represents the absence of information and $\boxed{c^\top : m}$ captures the non-emptiness of the depicted domain.

We have decomposed embeddings of graphs into structures into two separate phases: a mapping from concept nodes to objects and a test of the relational links. The whole interpretation process emphasises the correspondence between the simple graph edge structure and the relational structure in the represented model. This correspondence allies simplicity of semantics with resemblance of the representation to the represented (i.e., Hammer’s homomorphism thesis for diagrams [Ham95]).

From the interpretation of simple graphs, we remark that, for a given piece of information, there is an infinity of different (but semantically equivalent) graph representations. Indeed, if isomorphism takes implicitly care of the commutativity of conjunction, the duplication of atomic pieces of information is not limited. In particular, the multiple occurrences of concept nodes representing a single object are redundant. We now examine two different (and somehow opposite) graph transformations that uses these redundancies to defined canonical forms of a given graph.

3.1.4 Canonical forms

3.1.4.1 Normal forms

The first transformation eliminates duplicates of concept nodes which, a priori, represent a single object in any model of a graph.

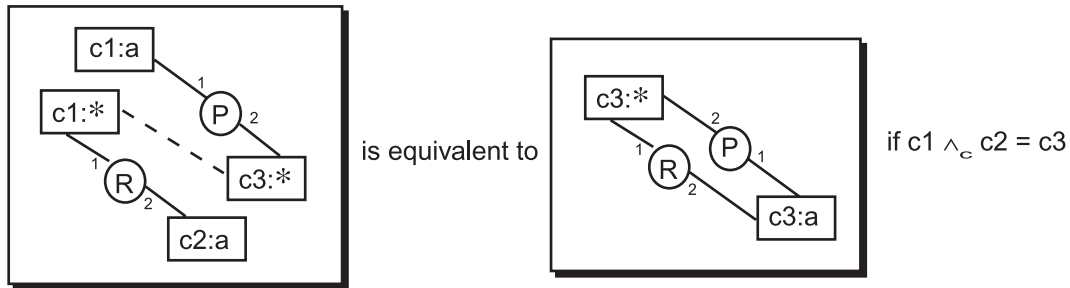


Figure 3.2: Normalisation step

The normal form of a simple graph is obtained by merging all occurrences of concept nodes having the same individual marker and all elements of each coreference equivalence class as depicted in Figure 3.2.

3.1.9. DEFINITION. [$co+$: an extended coreference equivalence relation] $G = (C, R, E, label, co)$ be a simple graph, we note $co+$ the equivalence relation on

concept nodes that extends co and capture the information that two nodes labelled with the same object name necessarily denote the same object:

$$\begin{aligned} \forall c, c' \in C_*, \quad (c, c') \in co+ & \text{ iff } (c, c') \in co \\ \text{and } \forall c, c' \in (C \setminus C_*), \quad (c, c') \in co+ & \text{ iff } marker(c) = marker(c') \end{aligned}$$

3.1.10. DEFINITION. [Normalisation of SCGs] A SCG $H = (C', R', E', label', co')$ is the normal form of a SCG $G = (C, R, E, label, co)$, noted $Norm(G) = H$, iff there exists a bijection $b_R : R \rightarrow R'$ and a bijection $b_C : class_{co+}(C) \rightarrow C'$ such that the following holds:

- R' is a duplicate of R : $\forall r \in R, label(r) = label'(b_R(r))$
- an equivalence class in C corresponds to a single concept node of the normal form, labelled with the meet of the concept labels: $\forall X \in class_{co+}(C)$, $type'(b_C(X)) = \bigwedge_C \{type(x)/x \in X\}$ and $marker'(b_C(X)) = marker(c)$ where $c \in X$
- co' is the identity on C' and the neighbourhood is preserved: $\forall (r, c, i) \in E$, $(b_R(r), b_C(class_{co+}(c)), i) \in E'$

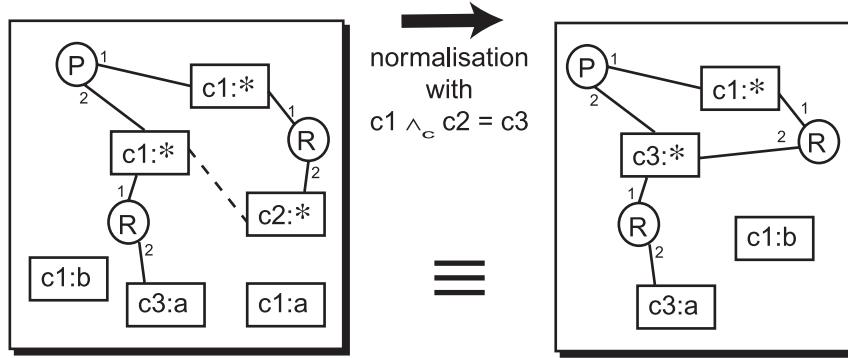


Figure 3.3: The transformation of the SCG in Figure 3.1 into its normal form

The normal form of a graph can be obtained during a single run through the sets of concept nodes and edges, thus in linear time in the size of the graph. Furthermore, normalisation is meaning preserving:

3.1.11. PROPOSITION. G and $Norm(G)$ are equivalent

Proof: The proof is immediate as each of the nodes in a $co+$ equivalence class and their unique representant in the normal form, denote the same object in a structure and as the extension of the meet of two concepts is equal to the intersection of the extensions of these concepts. ■

The conceptual graph formalism represents terms (concept nodes) and relations (relation nodes) on an egalitarian level: as nodes connected by edges. By opposition, formal languages of logic ascribe a somehow preponderant role to relations and atoms; after all, we say that a term is an argument of a predicate symbol. To ease the correspondence between graphs and formulae, it seems natural to let the relation nodes lead the dance.

3.1.4.2 Crazy forms

A crazy graph is a simple conceptual graph partitioned into atomic relational networks. There are two kinds of atomic subgraphs: isolated concept nodes and star graphs.

3.1.12. DEFINITION. [Isolated concept nodes and star graphs] • An isolated concept node is a concept node which is not connected to a relation node: for a SCG $G = (R, C, E, label, co)$, the node $c \in C$ is isolated iff $\forall (r, x, i) \in E$ it holds that $x \neq c$. Let $C_{isolated}$ denote the set of isolated concept nodes in G .

• A star graph is an atomic simple conceptual graph: i.e., a SCG composed of a single relation node connected to its concept node arguments.

For instance, in the rightmost drawing in Figure 3.4, the component indexed by (4) is an isolated concept node, whereas the remaining three components are star graphs. To transform a graph into its crazy form, we duplicate concept nodes that are linked to more than one relation node in the normal form of the graph.

3.1.13. DEFINITION. [Crazy form of a SCG] The crazy form of a simple conceptual graph G is the simple conceptual graph $Crazy(G) = (R, C', E', label', co')$ obtained from $Norm(G) = (R, C, E, label, co)$ as follows:

for any concept node $c \in C$ directly linked to $n > 1$ distinct relation nodes (i.e., $\{r_1, \dots, r_n\} = \{r \in R / \exists i, (r, c, i) \in E\}$ and $n > 1$), we replace c in $Norm(G)$ by n copies $\{c_1, \dots, c_n\} \subseteq C'$ in $Crazy(G)$ such that

1. $\forall 1 \leq j \leq n, label'(c_j) = label(c)$ and
2. for every relation node $r_j \in \{r_1, \dots, r_n\}$, any edge $(r_j, c, i) \in E$ is replaced by an edge $(r_j, c_j, i) \in E'$ and
3. if $marker(c) = *$ then $\{c_1, \dots, c_n\}$ are co' -equivalent.

3.1.14. FACT. For a SCG G , (i) $Crazy(G)$ is unique, (ii) the transformation from G to $Crazy(G)$ takes polynomial time and (iii) $Crazy(G)$ is equivalent to G .

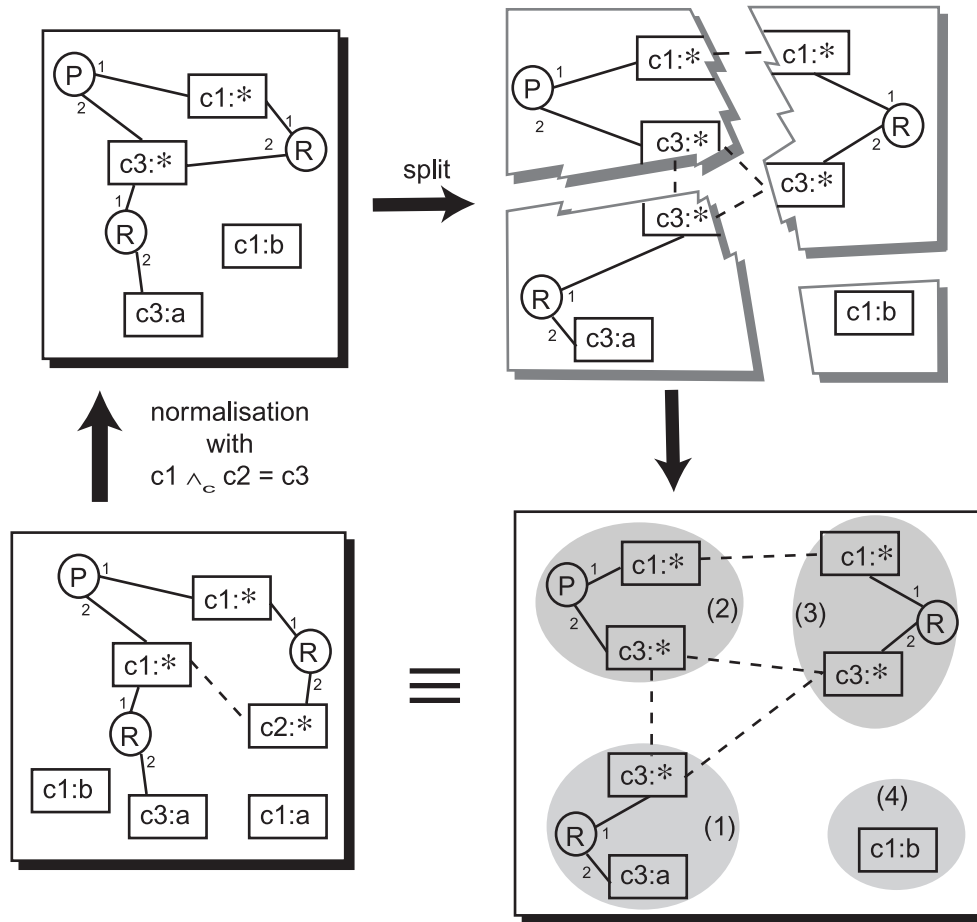


Figure 3.4: The transformation of the SCG in Figure 3.1 into its crazed form

Proof: (i) The normal form of the original graph is unique and furthermore, up to an isomorphism induced by the chosen indexing on the relational neighbourhood of a concept node, the splitting of a shared concept node is deterministic, unique and independent from any other splitting.

(ii) Normalisation takes polynomial time and each splitting consists in visiting all edges connected to the concept node at stake and modifying them. In the whole transformation, each concept node and each edge of the original graph is only visited once.

(iii) Obviously, the splitting of a node preserve the meaning of a graph as copies of an instantiated concept node refer to a single object in a structure and copies of an existential concept node (i.e., a node labelled with the marker $*$) are forced to refer to a single object by their assignment to a common coreference equivalence class. ■

Once partitioned into atoms and isolated concept nodes, a graph can be ascribed an arbitrary ordering of its components (e.g., the indices indicated between brackets in the rightmost graph in Figure 3.4), which will determine the outcome of its translation to a language of FOL.

3.1.15. DEFINITION. [Order on the components of a crazed graph] An ordered partition of the E -connected compounds of a non-empty crazed graph $G = (R, C, E, label, co)$ is defined as a function $comp : (R \cup C) \rightarrow \{1, \dots, n\}$ where $n = |C_{isolated}| + |R|$ and such that

1. $\forall x, x' \in (R \cup C_{isolated}), x \neq x'$ implies that $comp(x) \neq comp(x')$ and
2. $\forall c \in (C \setminus C_{isolated}), comp(c) = i$ if $\exists (r, c, j) \in E$ and $comp(r) = i$.

We note $(G, comp, n)$ such an ordered partition of the crazed graph G and $G_i = (C_i, R_i, E_i, label_i, co_i)$, the restriction of G to the nodes indexed by $i \in \{1, \dots, n\}$.

Fortified by these formal definitions of simple graphs and their meaning, we can now return to the correspondence with the usual textual language of existential conjunctive FOL and verify that our, so far only intuitively described, translation is conform to the intended semantics.

3.1.5 Textual correspondences

A language of conceptual graph is an order-sorted language (e.g. [BHR90], [SS89]) in which not only terms are associated to sorts organised into a predefined hierarchy, but relation names are ordered as well.

3.1.5.1 Translation of the support

A signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$ translates into a formula $\Phi(\Sigma)$ of a first-order language with constants in \mathcal{I} , relation symbols in $\mathcal{C} \cup \mathcal{R}$ and variables in an enumerable set VAR such that:

- $\forall c, c', c'' \in \mathcal{C}, c \wedge_{\mathcal{C}} c' = c''$ iff $\Phi(\Sigma) \models_{FOL} \forall x (c(x) \wedge c'(x) \rightarrow c''(x))$
- $\forall r, r' \in \mathcal{R}_n, r \leq_{\mathcal{R}} r'$ iff $\Phi(\Sigma) \models_{FOL} \forall x_1, \dots, x_n (r(x_1, \dots, x_n) \rightarrow r'(x_1, \dots, x_n))$

In other words, if an object belongs to the extension of two concepts, then it also belongs to the extension of their meet (thus, if an object belongs to the extension of a concept, then it also belongs to the extension of super-concepts). The order on relation names is similarly preserved.

A simple conceptual graph translates into an existentially closed conjunction of atoms. We propose an inductive translation based on the crazed form of a SCG.

3.1.5.2 Translation to $\text{FOL}_{\{\exists, \wedge\}}$, component by component

3.1.16. DEFINITION. $[\Phi]$ Let H be a SCG, if H is the empty graph then we define $\Phi(H) = \text{True}$. Otherwise, let (G, comp, n) be an ordered partition of the crazed form of H . The translation of the n components of $G = (R, C, E, \text{label}, \text{co})$ is defined inductively as follows.

• term is a function which associates to each concept node in G , a term such that:

- (i) $\forall c \in C, \text{term}(c) = m \in \mathcal{I}$ iff $\text{marker}(c) = m$ and
- (ii) $\forall c, c' \in C_*, \text{term}(c) = \text{term}(c') \in \text{VAR}$ iff $c \equiv_{\text{co}} c'$.

• For $i \in \{1, \dots, n\}$, $\text{Var}(G_i)$ denotes the set of variables which are associated to nodes in G_i and which are not associated to nodes in components with higher indices than i :

$$\text{Var}(G_i) = \bigcup_{c \in C_i} \left\{ \text{term}(c) / \begin{array}{l} \text{marker}(c) = * \\ \text{and } \forall c' \in C (c \equiv_{\text{co}} c' \rightarrow \text{comp}(c') \leq \text{comp}(c)) \end{array} \right\}$$

• The translation of a component may be decomposed in three parts:

1. A sequence of quantifiers as prefix (a function that transforms a formula into a formula):

$$\text{Quant}(G_i) : \varphi \rightarrow \begin{cases} \varphi & \text{if } \text{Var}(G_i) = \emptyset \\ \exists x_1 \dots \exists x_l (\varphi) & \text{if } \text{Var}(G_i) = \{x_1, \dots, x_l\} \end{cases}$$

2. The translation of the concept nodes:

$$\text{Conc}(G_i) = \bigwedge_{c \in C_i} \text{type}(c)(\text{term}(c))$$

3. The translation of an eventual (unique if it exists) relation node:

$$\text{Rel}(G_i) = \begin{cases} \text{True} & \text{if } R_i = \emptyset \\ P(\text{term}(r_i(1)), \dots, \text{term}(r_i(m))) & \text{if } \begin{cases} R_i = \{r_i\}, \\ \text{label}(r_i) = P \in \mathcal{R}_m, \\ \text{and } (r_i, r_i(j), j) \in E \end{cases} \end{cases}$$

We may now compose the translation from its parts and inductively obtain $\Phi(G) = \Phi_n(G)$:

base : if $\text{Rel}(G_1) = \text{True}$, then $\Phi_1(G) = \text{Quant}(G_1)(\text{Conc}(G_1))$ else

$$\Phi_1(G) = \text{Quant}(G_1)(\text{Rel}(G_1) \wedge \text{Conc}(G_1))$$

step $(k-1) \rightarrow k$ where $1 < k \leq n$:

if $Rel(G_k) = True$, then $\Phi_k(G) = Quant(G_k)(Conc(G_k) \wedge \Phi_{k-1}(G))$ else

$$\Phi_k(G) = Quant(G_k)(Rel(G_k) \wedge Conc(G_k) \wedge \Phi_{k-1}(G))$$

Of course, we have abused notations by defining Φ as a function from simple graphs to formulae. Indeed, the outcome of the translation relies on the chosen ordering on the components of the underlying crazed graph. However, we can notice that in the target language of existential conjunctive FOL, the ordering does only influence the relative position of atoms and quantifiers. Furthermore, as the translations are pure sentences (i.e., formulae with no free variables and variables quantified at most once), we can safely commute conjuncts and shift quantifiers, provided that the sentence characteristics is preserved, and obtain equivalent sentences including the translations that would be obtained with different orderings of the components.

3.1.17. EXAMPLE. A translation of the partitioned graph in Figure 3.4 is

$$\begin{array}{c}
 \overbrace{\hspace{15em}}^{\text{component(4)}} \\
 \underbrace{c1(b) \wedge \exists x \exists y (R(y, x) \wedge c1(y) \wedge c3(x) \wedge \overbrace{P(y, x) \wedge c1(y) \wedge c3(x) \wedge \underbrace{R(x, a) \wedge c3(x) \wedge c3(a)}^{\text{component(1)}}})}_{\text{component(3)}} \\
 \overbrace{\hspace{15em}}^{\text{component(2)}}
 \end{array}$$

We can now verify that the Φ -translation is conform to the interpretation of simple graphs.

3.1.18. PROPOSITION. For G a simple graph over a signature Σ and $M = (D, F)$ a Σ -structure,

$$M \models G \text{ iff } M \models_{FOL} \Phi(G)$$

Proof: Suppose that for some assignment f , $M, f \models G$, we want to prove that $M \models_{FOL} \Phi(G)$. We proceed by induction on the structure of the graph: the number of components in the crazed form.

If G is the empty graph, then $\Phi(G) = True$ and $\forall M, M \models G$ and $M \models_{FOL} True$.

Without loss of generality, let G be under its crazed form which is composed of n components. For $1 \leq k < n$, suppose that $M, f' \models_{FOL} \Phi_k(G)$ where f' which coincides with f : i.e., $\forall c \in C, f(c) = f'(term(c))$. We have to prove that (M, f') also satisfies the translation of the restriction of G to G_1, \dots, G_{k+1} ; viz. $M, f' \models_{FOL} \Phi_{k+1}(G)$.

- $\Phi_{k+1}(G) = Quant(G_{k+1})(Rel(G_{k+1}) \wedge Conc(G_{k+1}) \wedge \Phi_k(G))$

- By definition of an assignment (Definition 3.1.7), it holds that the set of concept nodes in G_{k+1} is satisfied by (M, f) , thus, by construction of f' (i.e., $\forall c \in C, f(c) = f'(term(c))$) it holds that $M, f' \models_{FOL} Conc(G_{k+1})$.
- Furthermore $M, f \models G$, thus, by Definition 3.1.8, the eventual relation occurrence between the concept nodes of G_{k+1} is verified in (M, f) . This implies that $M, f' \models_{FOL} Rel(G_{k+1})$.
- Therefore $M, f' \models_{FOL} Rel(G_{k+1}) \wedge Conc(G_{k+1}) \wedge \Phi_k(G)$. It follows that for any variable x , by definition of the interpretation of an existential quantifier, $M, f' \models_{FOL} \exists x(Rel(G_{k+1}) \wedge Conc(G_{k+1}) \wedge \Phi_k(G))$ thus $M, f' \models_{FOL} \Phi_{k+1}(G)$. The particular case where $Rel(G_{k+1}) = True$ is similar.

Reciprocally, the induction on the structure of the sentence runs as smoothly, by considering subformulae corresponding to the translation of components.

■

In this opening part on the representation of positive information, we have described the syntax of simple conceptual graphs using the formal mathematical language of sets and graphs. Graphs are built over an hierarchical vocabulary captured by the notion of support.

We have diverged from the traditional treatment of the semantics of simple conceptual graphs, which consists in translating the graphs into sentences of existential conjunctive FOL (see e.g., the extensive surveys in [Sow84] or [CM92] and the proceedings of ICCS conferences). Instead, we have chosen a set theoretical extensional semantics and directly interpreted the graphs by embedding into classical structures. This view renders the close connection between represented structures and graph representations, thus the facility of understanding the meaning of the graphs.

Of course, the translation into the textual language of FOL is also beneficial to conceptual graph theory. For instance, we can directly adapt the complexity results of the precious chapter by such correspondences. We have proposed an inductive translation based on the crazed form of a graph, i.e., its transformation to an equivalent graph decomposed into atomic subgraphs.

The idea of atomic decomposition behind the notion of crazed form is also used in the *anti-normal form* proposed by Ghosh and Wuwongse [GW95, Gho96] for logic programming with conceptual graphs.

There are some tiny differences due to our specific choices: contrary to the general trend in the conceptual graph literature, we have chosen not to fix in advance (i.e., in the support) the type of individual markers. Furthermore, we have allowed distinct concept nodes representing a single individual (either labelled

with the same constant or being coreferent) to be labelled with different concept types.

Our specific choices are, on one hand, justified by our ontological commitment to graph artefacts (e.g., the presence in a graph of a single instantiated concept node $\boxed{t : m}$ conveys the information of the existence of the individual represented by m and its belonging to the type t , whereas this information would be redundant with the support-information if this one should also predefine the same information). Therefore, our choices do, on the other hand, justify the previous lengthy and detailed treatment of basic definitions.

In the subsequent sections, crazed graphs will prove useful for the study of efficient reasoning in simple conceptual graphs enjoying tree-like properties. Reasoning on the graph representations will be the subject of the coming section.

3.2 Consequence proofs by homomorphism

If simple conceptual graphs share many syntactic similarities with Peirce's existential graphs or even, with the classical language of FOL, there is a facet on which they differ: the use for deductive purposes of projection, a labelled graph homomorphism method. This mapping idea which was already present in the embedding semantics is not only visual and intuitive, but it emphasises the role of the edge structure.

We will first present the projection calculus, prove its completeness with respect to the semantics of simple graphs and examine known complexity results.

3.2.1 Projection

A proof of a consequence relation between two graphs, called a projection [Sow84], takes the form of a labelled graph homomorphism: a mapping from the nodes of the conclusion to the nodes of the premiss which preserves the neighbourhood relation and respects the hierarchical information of the language signature.

3.2.1. DEFINITION. [Projection]

Let $G = (C, R, E, label, co)$ and $G' = (C', R', E', label', co')$ be two simple conceptual graphs over a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$. A projection from the source G' to the target G is a mapping $\pi : C' \cup R' \rightarrow C \cup R$ such that:

- $\forall c \in C', \pi(c) \in C$ and $type(\pi(c)) \leq_c type'(c)$
- $\forall c \in C',$ if $marker'(c) \in \mathcal{I}$ then $marker(\pi(c)) = marker'(c)$
- $\forall r \in R', \pi(r) \in R$ and $label(\pi(r)) \leq_{\mathcal{R}} label'(r)$
- $\forall (r, c, i) \in E', (\pi(r), \pi(c), i) \in E$

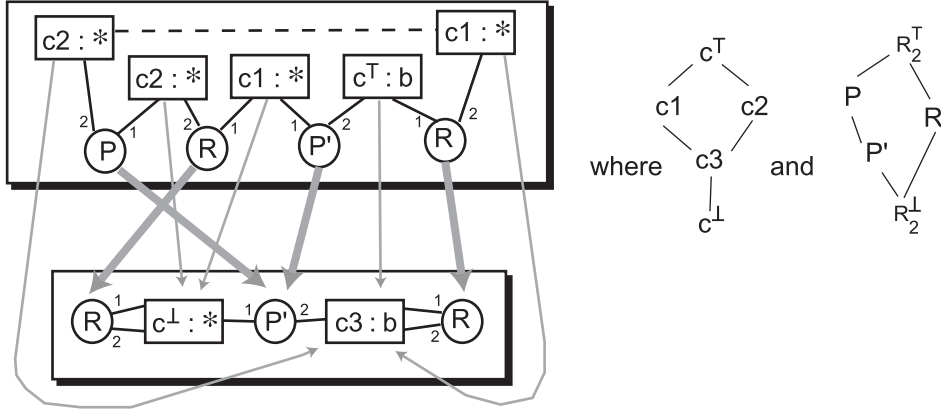


Figure 3.5: A projection

- $\forall (c_1, c_2) \in cO', \pi(c_1) = \pi(c_2)$

The mapping is a specialisation relation: a concept node is mapped onto another concept node labelled with a subconcept name, and if the source is instantiated with a proper name, then the target is labelled with the same name, otherwise (i.e., the source is labelled with a “*” representing an indefinite object) the target may be labelled with any marker. Similarly, a relation node is projected onto another relation node labelled with a subrelation name. Finally, the neighbourhood of a relation node is preserved by the mapping and coreferent concept nodes are associated to a single target.

When it is clear from the context which order on concept names is considered, we often use the derived order \leq : for $t, t' \in \mathcal{C}$, $m, m' \in \mathcal{I} \cup \{*\}$ and $r, r' \in \mathcal{R}$

$$\left\{ \begin{array}{l} (t, m) \leq (t', m') \text{ iff } t \leq_{\mathcal{C}} t' \text{ and } m' \in \mathcal{I} \rightarrow m = m' \\ r \leq r' \text{ iff } r \leq_{\mathcal{R}} r' \end{array} \right.$$

3.2.1.1 Independent sub-projections

A decomposition principle directly follows from the definition of a projection. It exemplifies the meaning of juxtaposition in the graphical language: juxtaposition represents conjunction.

3.2.2. FACT. For three simple graphs, G , H_1 and H_2 , there exists a projection from the juxtaposition of H_1 and H_2 (this simple graph is noted $H_1 \oplus H_2$) to G iff there exists a projection from H_1 to G and a projection from H_2 to G .

Proof: For the “only if”, it is obvious that the domain restriction of a projection $\pi : H_1 \oplus H_2 \rightarrow G$ to H_1 (and also to H_2) is a projection, as neither an edge nor a coreference link bridge the two compounds in the juxtaposition. Reciprocally, for the same reason, the union of a two projection (one for each of the two compounds) provides a projection of their juxtaposition. ■

This fact will later prove useful in different applications. For instance, the possibility of rewriting a cyclic graph into an equivalent juxtaposition of acyclic compounds guarantees the tractability of the projection reasoning. In the fragment of simple graphs including atomic negation, the possibility of splitting positive and negative pieces of information will be the key of a complete projection calculus.

A calculus is reliable if it is complete with respect to the defined semantics. In order to prove this property for projection, we need to define an intermediate notion: the canonical model of a simple graph.

3.2.2 Canonical model

We have already seen in the previous chapter that a sentence of the fragment corresponding to SCGs, $FOL_{\{\exists, \wedge\}}$, enjoys a minimal model. We will now define how such a model is built for a simple conceptual graph in normal form.

The canonical model of a simple graph can be read off its normal form:

3.2.3. DEFINITION. [Canonical model of a SCG] Let G be a simple conceptual graph in normal form $(C, R, E, label, co)$ over a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_C), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$, the canonical model of G , $M_G = (D, F)$, is built as follows:

1. $D = C$
2. $\forall m \in \mathcal{I}$, if $\exists c \in C$ such that $marker(c) = m$ then $F(m) = c$ else $F(m) = \odot$
3. for $concept \in \mathcal{C}$, $F(concept) = \{x/x \in C \text{ and } type(x) \leq_C concept\}$
4. for $P \in \mathcal{R}_n$, $F(p) = \{\langle r(1), \dots, r(n) \rangle / r \in R \text{ and } label(r) \leq_{\mathcal{R}} P\}$

We must now verify that the canonical model of a simple graph is indeed a structure which satisfies the graph.

3.2.4. FACT. Let G be a SCG in normal form over a signature Σ and $M_G = (D, F)$ be the canonical model of G , it holds that:

1. M_G is a Σ -structure.
2. The canonical model of a graph captures at least all the information contained in the graph:
 G is true in M_G under the identity-assignment.
3. Conversely, M_G conveys no more information than together G and the information that can be derived from G by applying meaning postulates of the language signature.

4. $\forall a, b \in \mathcal{I}, F(a) = F(b) \neq \odot$ implies that $a = b$.

Proof:

1. On \mathcal{I} , F is a function as the graph is in normal form. We need to verify that the ordering conveyed by the signature is preserved in the structure:
 - (a) $F(c^\top) = D$ follows from the fact that $\forall x \in \mathcal{C}, x \leq_c c^\top$.
 - (b) $\forall c, c' \in \mathcal{C}, c \leq_c c'$ implies that $F(c) \subseteq F(c')$ as by construction, for any concept node x , $x \in F(c)$ iff $\text{type}(x) \leq_c c$, thus $\text{type}(x) \leq_c c'$ by transitivity of \leq_c . Hence, $x \in F(c')$ by construction.
 - (d) $\forall c, c' \in \mathcal{C}, F(c \wedge_c c') \subseteq F(c) \cap F(c')$ directly follows from (b) and the definition of the meet. $\forall c, c' \in \mathcal{C}, F(c) \cap F(c') \subseteq F(c \wedge_c c')$: indeed, by construction, $x \in F(c) \cap F(c')$ implies that $\text{type}(x) \leq_c c$ and $\text{type}(x) \leq_c c'$. Thus, $\text{type}(x) \leq_c (c \wedge_c c')$ and $x \in F(c \wedge_c c')$.

Similarly for relation names (without the meet-constraint).

2. The canonical model captures all the positive information contained in G and thus the concept nodes and the positive relation nodes can be successfully interpreted under the identity assignment Id : Id is an assignment as for any concept node c with label (t, m) , by construction (Definition 3.2.3-3), it holds that $Id(c) = c \in F(t)$ and furthermore, if m is an object name (i.e., $m \in \mathcal{I}$) then $F(m) = Id(c) = c$. It remains to verify that relation nodes can successfully be interpreted: For any relation node r with arguments $\langle r(1), \dots, r(n) \rangle$, it holds that $\langle r(1), \dots, r(n) \rangle \in F(\text{label}(r))$ (by Definition 3.2.3-4).
3. By construction:
 - (a) $\forall P \in \mathcal{R}_n \forall \langle d_1, \dots, d_n \rangle \in D^n$, if $\langle d_1, \dots, d_n \rangle \in F(P)$, then $\langle d_1, \dots, d_n \rangle$ are the ordered arguments of a relation node $r \in G$ such that $\text{label}(r) \leq_{\mathcal{R}} P$ and
 - (b) $\forall d \in D, d$ is a concept node in G and
 - (c) $\forall \text{concept} \in \mathcal{C}$, if an object $d \in F(\text{concept})$, then $\text{label}_G(d) \leq_c \text{concept}$
4. G is normalised, thus an object name can appear in at most one concept node. Hence, F is a bijection between the set of object names occurring in G and the set of instantiated concept nodes.

■

As in the case of existential conjunctive FOL, we can state an equivalence between consequence and canonical model satisfaction:

3.2.5. THEOREM. *For G and H two simple graphs, $G \sqsubseteq H$ iff $M_{\text{Norm}(G)} \models H$*

Proof: the "only if direction" is immediate as by Fact 3.2.4-2 M_G is a model of G .

For the "if direction", assume that $M_{norm(G)}, f \models H$ and $N, g \models Norm(G)$. G and $Norm(G)$ are equivalent (Proposition 3.1.11), thus $N \models G$. We will show that $h = g(f)$ is such that $N, h \models H$.

$$N = (D_N, F_N) \xleftarrow{g} Norm(G) \xleftarrow{Fact\ 3.2.4-3} M_{Norm(G)} = (D, F) \xleftarrow{f} H$$

- instantiated concept nodes: let c be any concept node in H such that its label is (t, a) for $t \in \mathcal{C}$ and $a \in \mathcal{I}$. f is an assignment of H in $M_{Norm(G)}$, thus $f(c) = F(a) \neq \odot$ and $f(c)$ is the unique concept node labelled with 'a' in $Norm(G)$ (by Fact 3.2.4-3 and 4). $N, g \models Norm(G)$ thus, by definition of an assignment, $g(f(c)) = F_N(a)$.
- coreferent concept nodes: let c and c' be two distinct coreferent concept nodes in H . By definition of an assignment, $f(c) = f(c')$. Hence $g(f(c)) = g(f(c'))$.
- concept types: let c be any concept node in H and $t \in \mathcal{C}$ be its concept type label. $f(c) \in F(t)$ by definition of an assignment. Furthermore, $F(t) = \{x/x \in Norm(G) \text{ and } type(x) \leq_c t\}$ (by construction of the canonical model) and $\forall x \in F(t), g(x) \in F_N(type(x))$ (as g is an assignment of $Norm(G)$ in N). N is a Σ -structure, thus $\forall x \in F(t), g(x) \in F_N(t)$. Hence, $g(f(c)) \in F_N(t)$.
- atoms: let r be a relation node in H with label $P \in \mathcal{R}$ and ordered arguments $\langle r(1), \dots, r(n) \rangle$, we must verify that $\langle h(r(1)), \dots, h(r(n)) \rangle \in F_N(P)$. $\langle f(r(1)), \dots, f(r(n)) \rangle \in F(P)$ as $M_{Norm(G)}, f \models H$. Hence, by Fact 3.2.4-3, there exists a concept node s in $Norm(G)$ such that $label(s) \leq_{\mathcal{R}} P$ and $\forall 1 \leq i \leq n, s(i) = f(r(i))$. Furthermore, $N, g \models Norm(G)$, thus $\langle g(s(1)), \dots, g(s(n)) \rangle \in F_N(label(s))$ and $F_N(label(s)) \subseteq F_N(P)$ by definition of a Σ -structure.

■

Going one step further, by relating canonical model satisfaction and graph homomorphism, we will now prove that projection is consistent and complete with the defined semantics. We can note that the relevance of normalisation emerge in the proof of Theorem 3.2.5: the mapping from the canonical model to its original graph is only possible because the graph is in normal form. This normal form constraint will also be essential for the completeness of projection.

3.2.3 Completeness theorem

The completeness result of projection with respect to a FOL-translation was proved by Sowa [Sow84] (Soundness) and Chein and Mugnier [CM92, MC96] (Completeness). We prove here the completeness with respect to the direct interpretation of the graphs into structures.

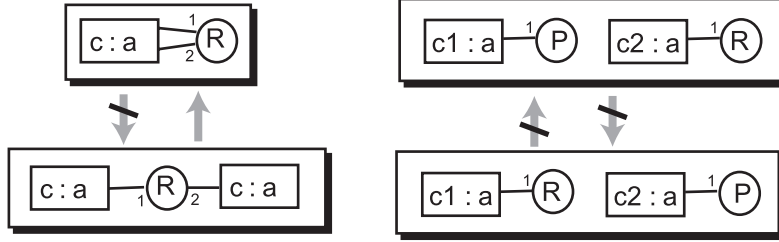


Figure 3.6: No projection between equivalent graphs: needed normalisation.

As noted by Preller in a personal communication to Chein and Mugnier (see Chapter 3.4 in [MC96]), some graph rewriting of the target graph of a projection is required for completeness: normalisation. Indeed, coreferent concept nodes or concept nodes instantiated with the same proper name necessarily denote a single object. Hence, they should find a mapping onto the same target. However, some equivalent simple graphs cannot always be associated to a projection on each other. For instance, in Figure 3.6, the two graphs on the left-hand side are equivalent but there is no projection from the top one to the bottom one as the conservation of the neighbourhood of the relation node cannot be satisfied. The remaining two graphs are also equivalent but if P and R are not comparable, then there is no projection between the graphs as \leq_c is antisymmetric.

The target graph must explicitly state that the object denoted by the proper name a belongs to the extension of the concept which is the meet of $c1$ and $c2$. To solve this problem, we can require the target graph to be in normal form.

3.2.6. THEOREM (COMPLETENESS OF PROJECTION). *For G and H two simple graphs, $G \sqsubseteq H$ iff there exists a projection from H to $Norm(G)$*

Proof: By Theorem 3.2.5, we must show that $M_{Norm(G)} \models H$ iff there is a projection from H to $Norm(G)$. Assume without loss of generality that G is already in normal form.

- Let π be a projection from $H = (C_H, R_H, E_H, label_H, co_H)$ to $G = (C_G, R_G, E_G, label_G, co_G)$, we must verify that there exists an assignment f from H to $M_G = (C_G, F)$ such that $M_G, f \models H$. Using the fact that G is satisfied by M_G under the identity function (Fact 3.2.4-2), we can chose the restriction of π to concept nodes in H as assignment f :
 - (i) f is a function from C_H to C_G

- (ii) $\forall c \in C_H$, $type_G(\pi(c)) \leq_c type_H(c)$ and $f(c) \in F(type_G(\pi(c)))$, thus $f(c) \in F(type_H(c))$ as by definition 3.1.6, $F(type_G(\pi(c))) \subseteq F(type_H(c))$
 - (iii) $\forall c \in C_H$ such that $marker_H(c) \in \mathcal{I}$, $marker_H(c) = marker_G(\pi(c))$ and $f(\pi'(c)) = F(marker_G(\pi(c)))$, thus $f(c) = F(marker_H(c))$
 - (iv) $\forall (c, c') \in co_H$, $\pi(c) = \pi(c')$ and therefore $f(c) = f(c')$.
 - (v) For any relation node r in R_H , $\langle r(1) \dots r(n) \rangle$ are the n ordered neighbours of r in H . By definition of a projection $label_G(\pi(r)) \leq_{\mathcal{R}} label_H(r)$ and the n ordered neighbours of $\pi(r)$ in G are $\langle \pi(r(1)) \dots \pi(r(n)) \rangle$. G is true in M under f thus $\langle f'(r(1)), \dots, f'(r(n)) \rangle \in F(label_G(\pi(r)))$ and by definition 3.1.6, $\langle f'(r(1)), \dots, f'(r(n)) \rangle \in F(label_H(r))$.
- Assume that for every structure in which G is true, H is also true, then there exists an assignment f of H in the canonical model $M_G = (D, F)$ under which H is true in M_G . We have to show that there exists a projection from H to G ; We first prove that the mapping f on concept nodes is conform to the definition of a projection: $\forall c \in C_H$, $f(c) \in C_G = D$ and
 - if $marker_H(c) \in \mathcal{I}$ then $f(c) = F(marker_G(f(c)))$ by definition of a canonical model and $f(c) = F(marker_H(c)) \neq \odot$ by definition of an assignment, so $F(marker_G(f(c))) = F(marker_H(c))$ and therefore, by Fact 3.2.4(3), $marker_G(f(c)) = marker_H(c)$
 - $f(c) \in F(type_H(c)) = \{c'/c' \in C_G \text{ and } type_G(c') \leq_c type_H(c)\}$. Thus, $type_G(f(c)) \leq_c type_H(c)$
 - $\forall c' \in C_H$, $(c, c') \in co_H$ implies that $f(c) = f(c')$.

We now have to verify that for every relation node in H , we can find a relation node in G such that the mapping preserve adjacency and label hierarchy: Let $r \in R_H$, if $label_H(r) = p \in \mathcal{R}_n$: H is true in M_G under f , thus $\langle f(r(1)), \dots, f(r(n)) \rangle \in F(p)$. By construction of M_G , $\langle f(r(1)), \dots, f(r(n)) \rangle \in F(p)$ iff $\exists r' \in R_G$ such that $label_G(r') \leq_{\mathcal{R}} p$ and $\forall 1 \leq i \leq n$, $r'(i) = f(r(i))$.

Therefore, $\exists r' \in R_G$ such that $label_G(r') \leq_{\mathcal{R}} label_H(r)$ and $\forall (r, c, i) \in E_H$, $(r', f(c), i) \in E_G$.

■

The problem of finding a projection is known decidable but untractable: it is an NP-complete problem (see, for instance, the complexity of the consequence problem in the existential conjunctive fragment of first-order logic in Chapter 2.5).

However, it is also known that structural constraints on the data can ease the complexity of reasoning. For instance, the problem of finding an homomorphism

from a labelled tree to a labelled graph is a polynomial problem: the search of a mapping from the source to the target, starting at the leaves towards the root, is an elimination process of possible projection candidates which runs at most once through each node of the source.

In Chapter 3.3, a larger tractable fragment of simple graphs will be proposed: the guarded simple conceptual graphs. Guarded simple graphs rely on a notion of tree-like structures. To arrive at this notion, it is instructive to examine the more usual notion of simple conceptual tree.

3.2.4 Tractable simple conceptual trees

In the simple conceptual graph fragment, we face a computational difficulty: consequence is untractable.

3.2.7. THEOREM. [CM92] *For two simple conceptual graphs G and H , proving the existence of a projection from H to G is NP-complete.*

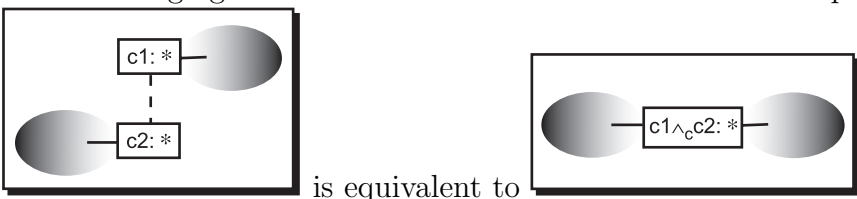
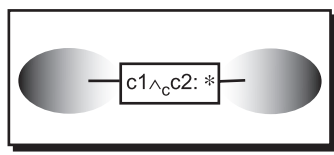
Automated reasoning being one of the goals for the conceptual graph formalism, it becomes relevant to exhibit how much constraint on the problem is sufficient for obtaining a tractable version.

A first answer was provided by Chein and Mugnier [CM92, MC92, Mug92]: when the source graph is a tree, the complexity of the projection problem decreases to polynomial. By transferring results from description logics to simple conceptual graphs, Baader et al. [BMT99] extended the tractable fragment to simple graphs that can be transformed into equivalent trees.

We recall these results and characterise these fragments in terms of tree covering of the coreference equivalence relation in the crazed form of a SCG.

3.2.4.1 Simple conceptual trees

The tree structure of a simple conceptual tree is more salient in a simple graph without coreferent nodes. We already know from the normal form of a graph that coreference can safely be eliminated. Indeed, any graph is equivalent to the graph resulting from the merging of all distinct elements in each coreference equivalence

class: i.e.,  is equivalent to .

3.2.8. DEFINITION.

[Edge-path, cycles and SCT] In a graph $G = (R, C, E, label, Id)$ where Id is the identity relation on C_* , an edge-path between two concept nodes $c, c' \in C$ is a sequence of distinct edges connecting the two concept nodes in the incidence undirected graph; in other

words, abstraction is made over multiple edges connecting a concept and a relation node:

$edge-path(c, c') = \langle e_1 = (c, f_1), \dots, e_j = (d_j, f_j), \dots, e_n = (d_n, c') \rangle$ such that

- (i) $n > 1$ and
- (ii) $\forall 1 \leq x \leq n, \exists i_x \in \mathbf{IN}^+ / (d_x, f_x, i_x) \in E$ or $(f_x, d_x, i_x) \in E$ and
- (iii) $\forall 1 \leq x < n, f_x = d_{x+1}$ and
- (iv) $\forall 1 \leq x \neq y \leq n, f_x = d_y$ implies that $d_x \neq f_y$.

A cycle is an edge-path from a concept node to itself and a simple conceptual tree (SCT) is a simple graph that does not contain any cycle.

For instance, the graph in Figure 3.7 is a simple conceptual tree.

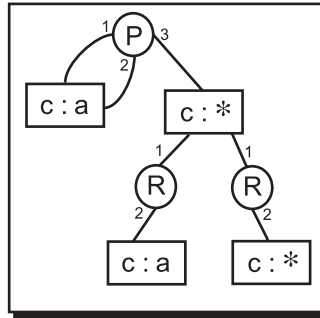


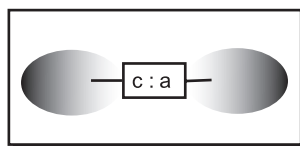
Figure 3.7: A simple conceptual graph with a tree structure

3.2.9. THEOREM. [CM92, Mug95] For a SCG G and a SCT T , proving the existence of a projection from T to G is polynomial.

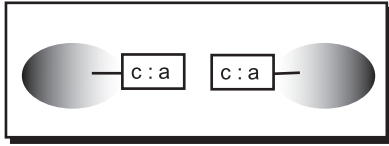
[Mug95] proposes a polynomial algorithm which builds a projection starting from the leaves of the tree towards the root and successively eliminating mapping candidates.

3.2.4.2 SCGs transformable into SCTs

In [BMT99], Theorem 3.2.9 is extended to source simple graphs that can be polynomially transformed into equivalent trees by splitting instantiated concept nodes.



Indeed, the graph  is equivalent to the graph



Hence, any cycle containing a concept node labelled with a proper name can be eliminated. For instance, the simple graph in Figure 3.8, which is not a simple conceptual tree, is equivalent to the tree in Figure 3.7.

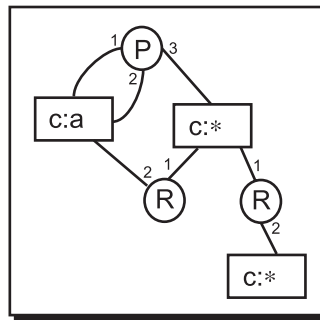


Figure 3.8: Eliminable cycles

3.2.4.3 Tree coverings of crazed graphs

Similarly to the Gaifman graph of a structure², the conceptual graph formalism puts forward the role of terms in representations. We have seen an alternative equivalent representation based on atomic sub-graphs: the crazed form of a SCG (Definition 3.1.13).

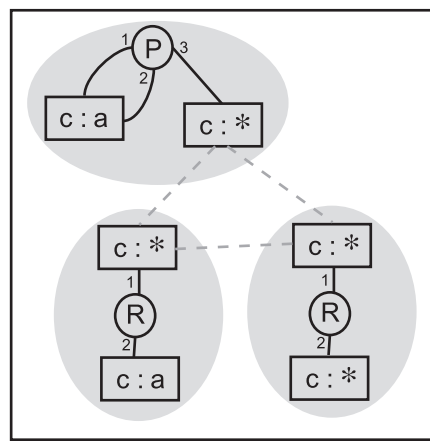


Figure 3.9: The crazed form of the graph in Figure 3.8

²The Gaifman graph [Gai82] of a structure has the elements of the universe as nodes and an edge connects two nodes iff the corresponding objects appear in the extension of a relation.

A tree covering of a crazed simple graph is an undirected acyclic graph, which covers the coreference equivalence relation in such a way that two components (star-graph or isolated concept node) are connected by at most one path of coreference-edges.

3.2.10. DEFINITION. [Covering of coreference edges]

Let $(G = (R, C, E, label, co), comp, m)$ be the crazed form of a SCG.

- A covering of G is a graph, (C_*, U_{co}) , such that two distinct coreferent concept nodes, c and c' , are linked by at least one path of coreference-edges in U_{co} .

Coreference edges are uplifted to the level of atomic components:

for $1 \leq k, l \leq m$ and two existential concept nodes $c \in G_k$ and $c' \in G_l$, a coreference-edge $\{c, c'\} \in U_{co}$ is also an edge connecting G_k and G_l .

An extended path between two components G_k and G_l is a sequence of components $\langle G_k = g_0, \dots, g_n = G_l \rangle$ such that $\forall 1 \leq i \leq n, g_{i-1} \neq g_i$ and there is a **unique** coreference-edge in U_{co} between g_{i-1} and g_i .

- A tree covering of G is an acyclic covering of G such that there is at most one extended path between two distinct atomic components of G .

3.2.11. EXAMPLE. A tree covering of the graph in Figure 3.9 is highlighted in Figure 3.10.

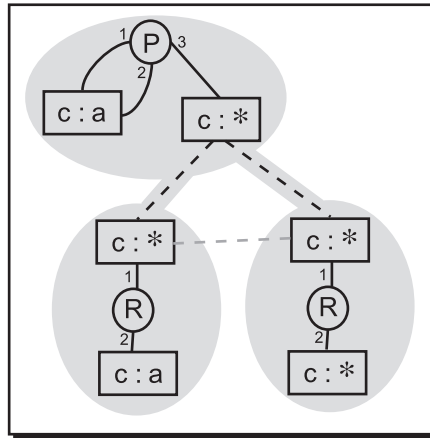


Figure 3.10: A tree covered crazed graph

The graph in Figure 3.11 has no tree covering. Indeed, its unique covering includes two distinct paths (a cycle) between the two components.

3.2.12. FACT. A simple graph G is transformable into an equivalent simple conceptual tree if and only if the crazed form of G has a tree covering.

Proof: on one hand, from a simple conceptual tree we can extract a tree covering of its crazed form. In the crazed form transformation, a concept node is split if

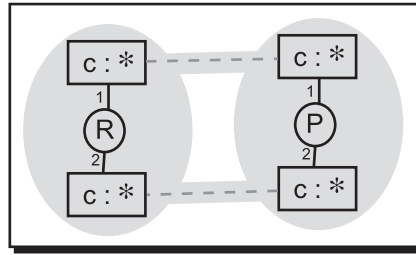


Figure 3.11: A cyclic graph

and only if it is originally shared by more than one relation nodes. For the case of an instantiated concept, no cycle can be introduced as the nodes is only split into copies. For the case of an existential concept node, we must chose locally a set of coreference-edges which is a tree and links every resulting copies; we simply chose an arbitrary copy and linked it to every other resulting copies. It then directly follows from the acyclicity of the original SCT, that these “local tree-coverings” combine together in a tree covering of the crazed form.

On the other hand, merging all elements of each coreference equivalence class transforms a tree covered graph into an equivalent simple tree (if there was a cycle in the resulting graph, it would correspond to a cyclic extended path in the crazed graph). ■

We should note that the previous proposition is not sensitive to redundant information taking the form of multiple copies of an atom. For example, the graph

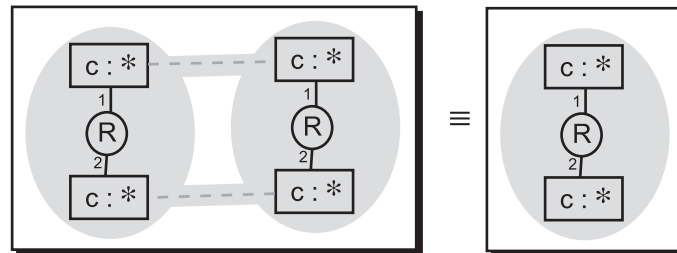


Figure 3.12: A cyclic path which could be eliminated

on the left-hand side in Figure 3.12 has no tree covering, however, it is equivalent to the irredundant graph pictured on the right-hand side which is acyclic.

In this section, we have reviewed the complete homomorphism calculus for consequence in simple conceptual graphs: projection. We have also recalled the intractability of the method, a complexity result which we already knew from the complexity of consequence in the corresponding fragment $FOL_{\{\exists, \wedge\}}$.

With implications for the safety of computational applications, a tree constraint on the source graph of a projection proves sufficient to enable a shift of the consequence problem into the tractable zone.

The next natural step consists in relaxing this constraint without going back beyond the untractability frontier. Our proposition will be a widening of the notion of tree covering, in order to accept cyclic paths such as the one in Figure 3.11.

3.3 Guarded simple conceptual graphs

From our previous experience with simple conceptual trees and the exploration of modal formalisms in Chapter 2, we learned that tree structures are a key to efficiency. How can we import in conceptual graphs, some modal tree features? Guarded syntactic forms are an answer; see, for instance, [Ben85], [ABN98] or [Grä99] for detailed studies of these embeddings of modal logics into classical logic.

The notion of guarded syntax, that we presented for $FOL_{\{\exists, \wedge\}}$ in Chapter 2.9.1, finds a natural formulation in terms of constraints on coverings of the coreference relation.

3.3.1. DEFINITION. [Guarded SCGs] Let $(G = (R, C, E, label, co), comp, m)$ be an ordered crazed form of a SCG and (C_*, U_{co}) be a covering of G .

A guarded path between two components G_k and G_l is a sequence of components $\langle G_k = g_0, \dots, g_n = G_l \rangle$ such that $\forall 1 \leq i \leq n, g_{i-1} \neq g_i$ and there is a (not necessarily unique) coreference-edge in U_{co} between g_{i-1} and g_i .

- A guarded covering of G is an acyclic covering of G such that there is at most one guarded path between two distinct atomic components of G .
- A simple graph is guarded if its crazed form admits a guarded covering.

3.3.2. EXAMPLE. In Figure 3.13, the simple graph is guarded by the highlighted covering.

The difference between a tree covering and a guarded covering is that, in the later, multiple coreference-edges are allowed to connect two components of the crazed graph.

3.3.3. FACT. The fragment of guarded simple conceptual graphs strictly includes the one of simple conceptual trees.

Proof: It directly follows from Definition 3.2.10 and Definition 3.3.1 that an extended path is a guarded path. Conversely, the graph in Figure 3.13 is guarded, but is not a tree. ■

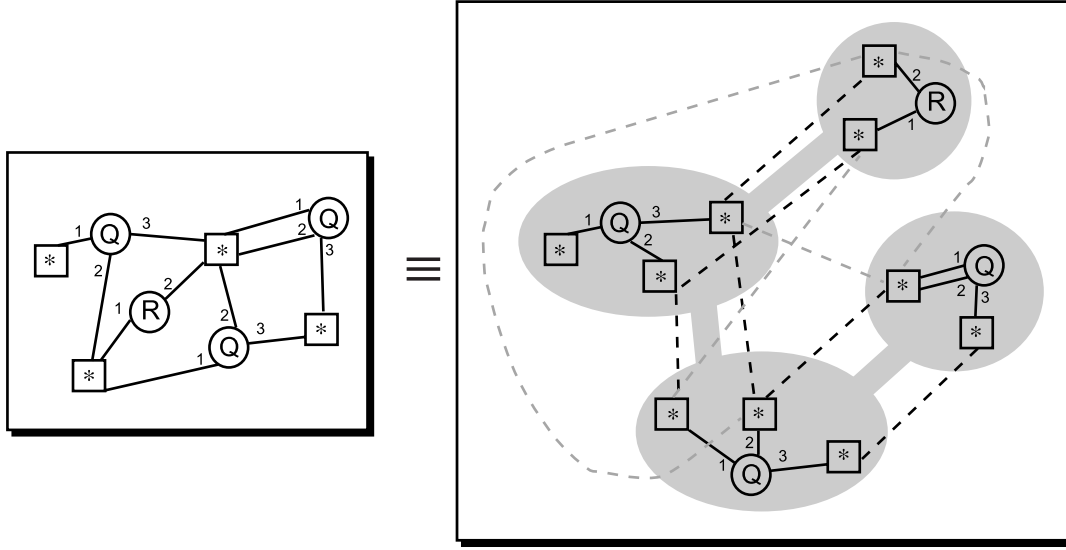


Figure 3.13: A guarded simple graph and one of its guarded coverings

3.3.0.4 Ordering derived from a guarded covering

To crazed graphs in general, we have associated an arbitrary ordering of the atomic components. Now that we have a tree structure linking these components in a guarded crazed graph, we can associate to the graph an ordering which respects the tree structure. To obtain the ordering, we first define a directed version of guarded covering.

3.3.4. DEFINITION. [Directed guarded covering]

Let $G = ((R, C, E, label, co), comp, m)$ be a guarded crazed simple graph.

• A directed guarded covering of G is a directed graph, $T' = (C_*, U'_{co})$, derived from a guarded covering $T = (C_*, U_{co})$ of G :

1. $U'_{co} \subseteq C_* \times C_*$ is irreflexive and asymmetric.
2. A direction is chosen for every undirected coreference edge in T :
 $\forall \{x, y\} \in U_{co}$, either $(x, y) \in U'_{co}$ or $(y, x) \in U'_{co}$.
3. All coreference edges between two components have the same orientation:
 $\forall \{x, y\}, \{x', y'\} \in U_{co}$ if $comp(x) = comp(x')$ and $comp(y) = comp(y')$ then $(x, y) \in U'_{co}$ iff $(x', y') \in U'_{co}$
4. Connected compounds are rooted:
 If there is a guarded path in T from a component G_k to a component G_l , then there is a guarded directed path in T' either from G_k to G_l or from G_l to G_k , where a guarded directed path from G_k and G_l is a sequence of components $\langle G_k = g_0, \dots, g_n = G_l \rangle$ such that $\forall 1 \leq i \leq n, g_{i-1} \neq g_i$ and there is a coreference-edge in U'_{co} from g_{i-1} to g_i .

- From a directed guarded covering T' , a successor function is defined. $succ : \{G_1, \dots, G_m\} \rightarrow \mathcal{P}(\{G_1, \dots, G_m\})$ such that $G_j \in succ(G_i)$ iff there is a directed coreference-edge in U'_{co} from G_i to G_j .

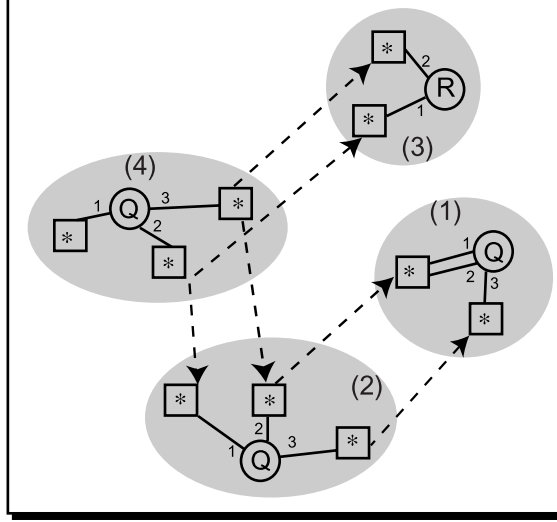


Figure 3.14: A directed guarded covering

A directed guarded covering can always be derived from a guarded covering of a SCG by an exhaustive run through the undirected structure. Such a run through the underlying tree structure provides an order of the components.

3.3.5. DEFINITION. [Post-order sequence of the components]

Let $(G = (R, C, E, label, co), comp, m)$ be a guarded crazed simple graph and $succ$ be the successor function derived from a directed guarded covering of G .

$comp$ is a post-order sequence of the components in G iff $\forall 1 \leq i, j \leq m, G_j \in succ(G_i)$ implies that $j < i$.

3.3.6. EXAMPLE. A possible directed guarded covering of the graph in Figure 3.13 is represented in Figure 3.14. A post-order sequence of the components, which respects the directed covering, is indicated by indexes between brackets.

3.3.0.5 On the FOL-translation of guarded simple graphs

The translation Φ (Definition 3.1.16) does not translate a guarded simple graph into a guarded sentence of $FOL_{\{\exists, \wedge\}}$, but in a sentence which is equivalent to a guarded one.

Locally, each component translates to a guarded formula: if $G_i \in G$ is an isolated concept node, it translates to an atom Pt eventually preceded by a quantifier $\exists t$. Hence, a guarded formula.

If G_i is a star graph, the translation of its relation node, $Rel(G_i)$, is a guard for the remaining conjunction of unary atoms (i.e., atoms reflecting the concept type of terms). Indeed, all existentially quantified variables in $Quant(G_i)$ and all terms in $Conc(G_i)$ occur as argument in $Rel(G_i)$.

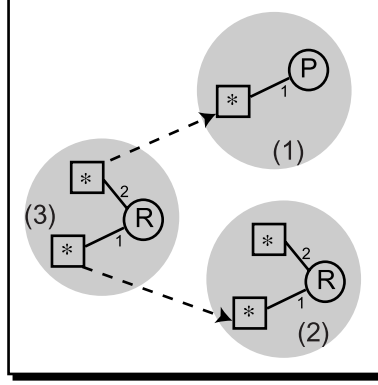


Figure 3.15: Translation problem

However, the translation of components in a linear fashion does not capture the general tree structure of a guarded covering. For instance, using the pictured ordering of components, the guarded graph G in Figure 3.15, would translate

to the non-guarded sentence $\Phi(G) = \exists xz(Rzx \wedge \underbrace{\exists y(Rzy \wedge Px)}_{\text{component}(2)})$, whereas

the equivalent sentence $\underbrace{\exists xz(Rzx \wedge \underbrace{\exists y(Rzy)}_{\text{component}(2)})}_{\text{component}(3)} \wedge \underbrace{Px}_{\text{component}(1)}$ is guarded.

In order to translate guarded graphs into ECGF-sentences, we can use the successor function on components:

$$\Phi_k(G) = Quant(G_k)(Rel(G_k) \wedge Conc(G_k) \wedge \bigwedge_{l/G_l \in succ(G_k)} \Phi_l(G))$$

By induction hypothesis, for $G_l \in succ(G_k)$ (thus, $l < k$), $\Phi_l(G)$ is guarded and every variable occurring free in G_l also occurs in G_k . Hence $\Phi_k(G)$ is guarded.

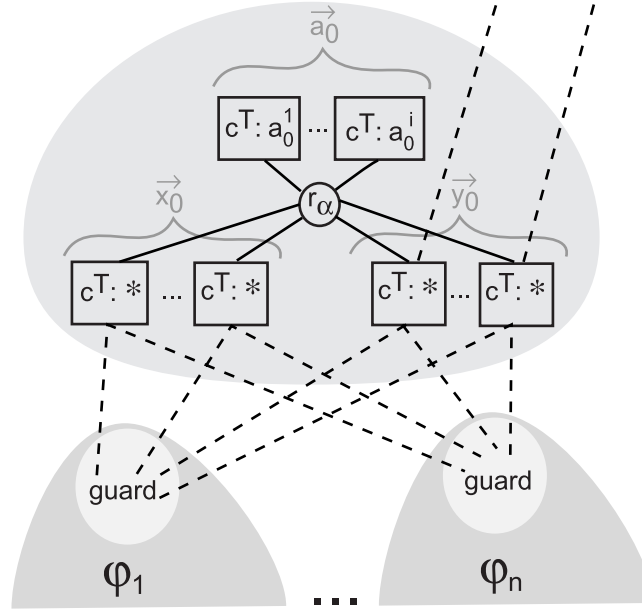
Reciprocally, any sentence in ECGF can be represented as a guarded simple graph.

Without loss of generality, we can consider a pure ECGF-sentence (i.e., a sentence in which no variable is quantified twice).

- An atom $P(t_1, \dots, t_n)$ corresponds to a star graph whose relation node is labelled with P . The i^{th} neighbour of the relation node (for $1 \leq i \leq n$) is a concept node

labelled with (c^\top, t_1) . This single star-graph is necessarily guarded.

- A conjunction of ECGF-formulae corresponds to the juxtaposition of graphs. Furthermore, if this conjunction is a sentence, then each conjunct is an ECGF-sentence and the graphical correspondent is a guarded graph partitioned into guarded connected compounds, by induction hypothesis.
- The main case of the induction step is illustrated as follows:



an ECGF-formula $\exists \vec{x}(\alpha(\vec{x}_0, \vec{y}_0, \vec{a}_0) \wedge \varphi_1(\vec{x}_1, \vec{y}_1, \vec{a}_1) \wedge \dots \wedge \varphi_n(\vec{x}_n, \vec{y}_n, \vec{a}_n))$ such that for $0 \leq i \leq n$, φ_i is not a conjunction and $\vec{x}_i \subseteq \vec{x}_0$, $\vec{y}_i \subseteq \vec{y}_0$ and \vec{a}_i is composed of constants. The guard α translates into a star-graph root whose successors are, by induction, roots of guarded graphs. The result is therefore a guarded graph.

We have seen that the guarded fragment of simple conceptual graphs includes the fragment of simple conceptual tree. In the next section, we will consider the complexity of the projection problem in the guarded fragment.

3.3.1 Complexity of consequence for guarded SCGs

To prove that deciding the existence of a projection from a guarded simple graph to another simple graph (in normal form) is polynomial time, we propose a projection algorithm which exploits the particular structure of coreference links in the guarded graph.

Input: $H = (R_H, C_H, E_H, label_H, co_H)$ is a simple graph in normal form,
 $G = ((R, C, E, label, co), comp, n)$ is a crazed guarded simple graph such that $comp$ is a post order sequence of the components derived from a directed guarded covering $T = (C_*, U_{co})$.

Output: “Yes” if there exists a projection from G to H , “No” otherwise.

Initialisation: FOR ALL $x \in R \cup C, \delta(x) := \emptyset$

FOR $1 \leq i \leq n,$

IF G_i is an isolated concept node c	THEN (A) $\forall c' \in C_H, \text{ IF } label_H(c') \leq label(c) \text{ THEN } \delta(c) := \delta(c) \cup \{c'\}$																						
ELSE (B)	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">IF</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">let r be the single relation node in $G_i,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">FOR ALL $r' \in R_H,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B3) $\forall g \in succ(G_i),$</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table> </td> </tr> <tr> <td style="border: none; padding-right: 10px;">THEN</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table> </td> </tr> </table> </td> </tr> </table>	IF	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">let r be the single relation node in $G_i,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">FOR ALL $r' \in R_H,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B3) $\forall g \in succ(G_i),$</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table> </td> </tr> <tr> <td style="border: none; padding-right: 10px;">THEN</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table> </td> </tr> </table>	let r be the single relation node in $G_i,$		FOR ALL $r' \in R_H,$		(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and		(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and		(B3) $\forall g \in succ(G_i),$	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table>	$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$		where p is the single relation node in g		THEN	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table>	$\delta(r) := \delta(r) \cup \{r'\}$ and		$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$	
IF	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">let r be the single relation node in $G_i,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">FOR ALL $r' \in R_H,$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">(B3) $\forall g \in succ(G_i),$</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table> </td> </tr> <tr> <td style="border: none; padding-right: 10px;">THEN</td> <td style="border: none; padding-left: 10px;"> <table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table> </td> </tr> </table>	let r be the single relation node in $G_i,$		FOR ALL $r' \in R_H,$		(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and		(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and		(B3) $\forall g \in succ(G_i),$	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table>	$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$		where p is the single relation node in g		THEN	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table>	$\delta(r) := \delta(r) \cup \{r'\}$ and		$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$			
let r be the single relation node in $G_i,$																							
FOR ALL $r' \in R_H,$																							
(B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and																							
(B2) $\forall r'(j) \in C_H, label_H(r'(j)) \leq label(r(i))$ and																							
(B3) $\forall g \in succ(G_i),$	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">where p is the single relation node in g</td> <td style="border: none;"></td> </tr> </table>	$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$		where p is the single relation node in g																			
$\exists p' \in \delta(p) / \forall u = (r(j), p(k)) \in U_{co}, r'(j) = p'(k)$																							
where p is the single relation node in g																							
THEN	<table style="border: none; border-collapse: collapse;"> <tr> <td style="border: none; padding-right: 10px;">$\delta(r) := \delta(r) \cup \{r'\}$ and</td> <td style="border: none;"></td> </tr> <tr> <td style="border: none; padding-right: 10px;">$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$</td> <td style="border: none;"></td> </tr> </table>	$\delta(r) := \delta(r) \cup \{r'\}$ and		$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$																			
$\delta(r) := \delta(r) \cup \{r'\}$ and																							
$\forall j / 1 \leq j \leq arity(label(r)), \delta(r(j)) := \delta(r(j)) \cup \{r'(j)\}$																							
IF $\forall x \in R \cup C, \delta(x) \neq \emptyset$	THEN return “Yes” ELSE return “No”.																						

3.3.1.1 Description and complexity of the algorithm

The algorithm considers successively, in the chosen order which respects the general tree structure of a covering, every basic component (i.e., star graphs and isolated concept nodes) of the source graph and for each of them, searches for a maximal set of acceptable mapping targets in the target graph. To become acceptable, a potential target component must not only satisfy the labelling constraints of a projection (a specialisation relation), but as well comply with candidates that have already been chosen for the neighbourhood of the source component. There are $n \in \mathcal{O}(|C_G|)$ basic components in G (every relation node is directly linked to at least one concept node and two distinct relation nodes do not share direct concept node neighbours in a crazed graph).

There are two possible scenarios depending on the source component that we try to map:

- (Part A of the algorithm) The source component is an isolated concept node: it is either labelled with a proper name or it is a generic concept node which is not coreferent to any other concept node. In both cases, it is sufficient to find a concept node of the target graph with a more specialised label. A run through every concept node of the target graph takes $\mathcal{O}(|C_H|)$ label checks. The time required for each of these checks is not dependent on the size of the two input graphs, but on the size and the coding of the language signature. We may therefore consider for the projection problem that a label check is done in constant time.
- (Part B of the algorithm) The source component G_i is a star graph and we must find every target star graphs which is a specialisation of G_i . If we call r the unique relation node of G_i , a relation node r' in the target graph is a potential mapping candidate if its label is a specialisation of the label of r

(thus, both relations also have the same arity k) and for every i from 1 to k , the i^{th} argument of r' is a specialisation of the i^{th} argument of r . These two comparison take $\mathcal{O}(1 + |C_G|)$.

Now, if the source component has no successor in the directed guarded covering, then we are done with r' . Otherwise, we must check that the neighbourhood of r' complies with the candidates that have been selected for the successors of G_i in the directed covering (by definition of the chosen ordering, these successors have already been treated). That is, for every star graph successor of G_i , we must find among the candidates already selected for its relation node p , a relation node p' in the target graph such that every pair of coreferent arguments for p and r corresponds to a single concept node that occurs as argument of both p' and r' . G_i has at most $\mathcal{O}(|R_G|)$ successors, each p has at most $\mathcal{O}(|R_H|)$ mapping candidates and there are at most $\mathcal{O}(|C_G|)$ coreference links between G_i and one of its successors.

Therefore, an upper bound for the total complexity of the algorithm is $\mathcal{O}(|C_G| * \max(|C_H|, (|R_H| * ((1 + |C_G|) + |R_G| * |R_H| * |C_G|))))$ which is itself bound by $\mathcal{O}(k^5)$ where $k = \max(|C_H|, |R_H|, |R_G|, |C_G|)$.

3.3.1.2 Soundness and completeness of the algorithm

Completeness: if there exists a projection from G to H then the algorithm answers “Yes”.

Proof: assume a projection from G to H , i.e. a mapping $\pi : C \cup R \rightarrow C_H \cup R_H$ such that

1. $\forall c \in C, \pi(c) \in C_H$ and $\text{label}_H(\pi(c)) \leq \text{label}(c)$ and
2. $\forall r \in R, \pi(r) \in R_H$ and $\text{label}_H(\pi(r)) \leq_{\mathcal{R}} \text{label}(r)$
and $\forall 1 \leq i \leq \text{arity}(\text{label}(r)), \pi(r(i)) = (\pi(r))(i)$ and
3. $\forall (c1, c2) \in co, \pi(c1) = \pi(c2)$.

We will show that for every star graph G_m in G and every node x in G_m it holds that $\pi(x) \in \delta(x)$.

Basic step $m = 1$: by definition of the ordering, no concept node in G_1 has a successor in the directed covering T (the constraint (B3) is fulfilled).

(A) If G_1 is an isolated concept node c , then, by definition of a projection, $\pi(c)$ is a concept node of H such that $\text{label}_H(\pi(c)) \leq \text{label}(c)$. Thus, $\pi(c) \in \delta(c)$.

(B) If G_1 is a star graph. Let r be its unique relation node, k is the arity of its relation name label. π is a projection, thus, if $\pi(r) = r' \in R_H$ then $\text{label}_H(r') \leq_{\mathcal{R}} \text{label}(r)$ and $\forall j/1 \leq j \leq k, \text{label}_H(r'(j)) \leq \text{label}(r(j))$. Therefore, the tests (B1) and (B2) succeed and $r' \in \delta(r)$ and $\forall 1 \leq j \leq k, \pi(r(j)) = r'(j) \in \delta(r(j))$.

Induction step $1 < m \leq n$: either G_m is an isolated concept node or a star graph.

(A) In the first case, similarly to the case of G_1 , G_m is a single concept node whose image by π has a more specialised label. Thus, $\pi(c) \in \delta(c)$.

(B) G_m is a star graph. Let r be its unique relation node, k be the arity of its relation name label and r' its image by the projection π , it holds that $label_H(r') \leq_{\mathcal{R}} label(r)$. It remains to be shown that (B2) and (B3) are satisfied by r' . (B2) follows directly from the definition of a projection: the projection of the i^{th} argument of r (noted $r(i)$) is the i^{th} argument of r' and the label of $r'(i)$ is more specific than the label of $r(i)$. For (B3), a stronger property follows from the definition of a projection: $\forall x, y \in R_G, \forall 1 \leq i \leq arity(label(x)), \forall 1 \leq j \leq arity(label(y))$, if $x(i) \equiv_{co} y(j)$ then $\pi(x)(i) = \pi(y)(j)$. Indeed, from the third condition of the definition of the projection, $x(i) \equiv_{co} y(j)$ implies that $\pi(x(i)) = \pi(y(j))$, then, it follows from the second condition that $\pi(x(i)) = \pi(x)(i)$ and $\pi(y(j)) = \pi(y)(j)$.

$\forall x \in R \cup C, \pi(x) \in \delta(x)$, therefore the algorithm returns “Yes”.

Soundness: if the algorithm answers “Yes” then there exists a projection from G to H .

Proof: suppose that the algorithm returns “Yes”, we will extract a projection π from the mapping $\delta : C \cup R \rightarrow \mathcal{P}(C_H \cup R_H) \setminus \emptyset$.

We will prove that $\forall 1 \leq m \leq n$, (i) the recursively build mapping π is a projection from $G(m)$ to H where $G(m)$ is the restriction of G to the star graphs G_m, G_{m+1}, \dots, G_n (including the coreference links between the concept nodes of these star graphs) and (ii) for every node x in $G(m)$, $\pi(x) \in \delta(x)$.

Basic step $m = n$:

(A) if G_n is an isolated concept node c . We select a concept node c' in $\delta(c) \neq \emptyset$ and set $\pi(c) := c'$. c' has passed the test (A) in the algorithm (i.e., $label_H(c') \leq label(c)$), thus, π is a projection from G_n to H .

(B) if G_n is a star graph whose unique relation node r is labelled with a relation symbol of arity k . We can select any relation node r' in $\delta(r)$ and set $\pi(r) := r'$ and $\forall 1 \leq j \leq k, \pi(r(j)) := r'(j)$. $r' \in \delta(r)$ therefore $label_H(r') \leq_{\mathcal{R}} label(r)$ and $\forall 1 \leq j \leq k, label_H(r'(j)) \leq label(r(j))$. Furthermore, by definition of a crazed graph, two distinct concept nodes of G_n cannot be coreferent. Hence π is a projection from G_n to H .

Induction step $1 \leq m < n$: suppose that π is a projection from $G(m+1)$ to H , we extend π to a projection from $G(m)$ to H .

(A) If G_m is an isolated concept node c , similarly to the case of G_n , we may select any concept node $c'' \in \delta(c)$ and set $\pi(c) := c''$. It holds that $label_H(c'') \leq label(c)$ and c is not coreferent to another (distinct) concept node, hence, π is a projection from $G(m)$ to H .

(B) The remaining case concerns a star graph G_m whose relation node, r , is labelled with a relation of arity k . In order to obtain a projection from $G(m)$

to H we must select in $\delta(r)$, a π -image r' for r such that (i) the specialisation relation on labels is respected and (ii) for every argument $r(x)$ of r which is coreferent to a concept node c in $G(m+1)$, it holds that $\pi(r(x)) = \pi(c)$. We first note that any choice of r' in $\delta(r)$ respects the specialisation relation on labels required for a projection; that is, (part B1) $label_H(r') \leq_{\mathcal{R}} label(r)$ and (part B2) $\forall 1 \leq j \leq k, label_H(r'(j)) \leq label(r(j))$ and (conclusion of part B) $r'(j) \in \delta(r(j))$. Therefore, we must show that in $\delta(r)$, there exists a node r' such that for every pair of coreferent concept nodes $(r(x), c)$ in $G_m \times G(m+1)$, $r'(x) = \pi(c)$. However, by definition of a directed covering, all the eventual U_{co} -predecessors of concept nodes in G_m belong to a single star graph G_l such that $m < l$, and furthermore, by induction hypothesis (i.e., the restriction of π to $G(m+1)$ is a projection onto H) and the definition of a projection, $\forall (c_1, c_2) \in G(m+1) \times G(m+1), c_1 \equiv_{co} c_2$ implies that $\pi(c_1) = \pi(c_2)$. It is therefore sufficient to show that in $\delta(r)$, there exists a node r' such that for every pair of coreferent concept nodes $(r(x), p(y))$ in $G_m \times G_l$, $r'(x) = \pi(p)(y)$ where p is the relation node of G_l . By induction hypothesis, it holds that $\pi(p) \in \delta(p)$. Furthermore, $G_m \in succ(G_l)$, thus, if $\pi(p) \in \delta(p)$ (conclusion of the part (B) of the algorithm), then (part B3) $\exists r' \in \delta(r) / \forall u = (p(y), r(x)) \in U_{co}, \pi(p)(y) = r'(x)$. Now that the existence of a proper candidate r' on which r can be projected exists, we can select a such relation node and extend π to a projection from $G(m)$ to H by setting $\pi(r) := r'$ and $\forall 1 \leq j \leq k, \pi(r(j)) := r'(j)$.

$G(1) = G$, hence we have extracted from δ a projection $\pi : G \rightarrow H$.

■

Normalisation being of polynomial time complexity, we have proved that deciding the projection problem (hence, the consequence problem) for a source guarded simple conceptual graph and a target simple conceptual graph (without restriction) can be done in time polynomial in the size of the input graphs:

3.3.7. THEOREM (COMPLEXITY OF SUBSUMPTION FOR GUARDED SCG). *For a SCG G and a guarded SCG H , $G \sqsubseteq H$ is decidable in polynomial time.*

Recent developments in database theory have focussed on conjunctive queries with bounded tree-width (which correspond to the k -variable fragment of $FOL_{\{\exists, \wedge\}}$ [KV00]) and bounded hypertree-width [GLS99, GLS01]. In [GLS01], an elegant characterisation of hypertree-width is proposed in game theoretic terms. It is also shown that the k -guarded fragment (i.e., a guard is a conjunction of at most k atoms) of $FOL_{\{\exists, \wedge\}}$ is the class of conjunctive queries with bounded hypertree-width k . Hence, the guarded fragment of $FOL_{\{\exists, \wedge\}}$ corresponds to the class of conjunctive queries with hypertree-width 1. Furthermore, [GLS99] proves the polynomial time complexity of the evaluation of conjunctive queries whose hypergraphs have bounded hypertree-width. This tractability result and the strong connection between conjunctive query evaluation and containment [CM77] suggest promising extensions of the polynomial projection algorithm to a notion of k -guarded simple graphs.

3.4 Conclusions

Simple conceptual graphs are an alternative graphical notation for existential conjunctive first-order sentences. In order to formalise properties of these drawings, we have used the usual symbolic language of graph theory.

A first property of conceptual graph systems is that their languages are founded on predefined classifications of the basic vocabulary elements; the so-called ontologies in artificial intelligence. We have presented our specific choices on the form of these hierarchies.

A simple conceptual graph is a finite bipartite graph that represents a conjunction of atomic facts. In the variable-free notation inspired by Peirce's existential graphs, objects are denoted by proper names or the use of a generic marker which symbolises existential quantification. The existence of an object is stipulated by the physical drawing of its representation, a concept node. To provide a meaning to the drawings, we have interpreted the simple conceptual graphs by embedding into first-order structures.

Departing from traditional first-order calculi, these drawings admit a method for proving consequences, which is also based on the embedding idea: a simple graph follows from another simple graph if the later can be mapped onto the former. Despite the interesting visual property of the projection method, it is computationally untractable for the whole fragment of simple conceptual graphs. Driven by applications of conceptual graph systems to automated reasoning, research has focussed on tractable subfragments of simple conceptual graphs and in particular on graphs presenting a tree structure. In [PD98], projection is also studied as a resource sensitive way of making proofs in Cartesian Closed Categories.

Using the structure presented by the standard translation of modal formulae, we have extended the known tractable class of simple conceptual trees to the class of guarded simple conceptual graphs and proved its polynomial time complexity for the consequence problem. This results is a refined answer to the question "what are sufficient structural constraints on the input of projection to obtain a tractable problem?".

We have only considered constraints on one side of the input, viz. the source graph. Promising constraints on the target graph have been studied in database theory; for instance, sentences in which a relation symbol occurs at most twice [Sar91] or the 2-colourability of a sentence [KV00]. This last property as a visual aspects which would deserve some further work in the conceptual graph setting.

We have presented the simple conceptual graphs fragment as a central language of richer conceptual graph systems. We will now consider its enrichment with different forms of negation.

Chapter 4

Richer pictures

In this chapter, we will consider possible extensions of the simple conceptual graph model. In a search for additional language features and under the control of complexity theory, one cannot escape the well identified tradeoff between expressivity and computational efficiency. Expressive power change is taken here in a broad sense: more than a pure formal notion of symbolic logic comparison, it includes the cognitive impact of considering new language elements that facilitate the representation of information. Indeed, among the different extensions, we will encounter forms of conceptual graph that faithfully depict some characteristics of the represented information such as negative propositions or structured knowledge, while being logically equivalent to the basic language of simple graphs. Continuing on the previous chapter, our guidelines will remain the applications of graphical methods as calculi and the recognition of fragments for which our central benchmark problem, consequence, is tractable.

The language of simple conceptual graphs enables to express positive relational facts between objects with a bit of indeterminacy conveyed by the use of a place holder, the marker ‘*’. The first natural extension we consider, is to allow the expressibility of negative facts or in more classical terms, we introduce an atomic negation operator in the language. Chapter 4.1 examines the completeness problem encountered by projection in presence of negated atoms. A syntactical constraint is proposed to define a fragment of simple graphs with atomic negation for which projection is complete and even tractable on guarded forms.

In a second part, Chapter 4.2, we make a large step to a language of graphs as expressive as classical first-order logic. Our perspective is already influenced by the known undecidability of our benchmark problems in FOL, thus we follow a path that moves us away from efficiency considerations and instead, we study the possibility of combining two graphical methods in one complete calculus: projections and classical tableau decomposition rules. Such a generic and modular proof method for the whole language opens the possibility of examining particular tunings of the components in order to suit the needs of intermediate fragments.

Our final language takes us back to the cornerstone of this thesis, existential conjunctive FOL, but with a superstructure of hierarchical networks. In this setup, the information conveyed by a graph is partitioned into smaller local pieces of information. The representation of localised -or contextualised- information has been recognised as a need in artificial intelligence. Nested graphs, graphs whose nodes are themselves graphs, follow this principle of locality.

4.1 Atomic negation

Describing the world, one may not only want to provide a positive image! The possibility of expressing negative facts enriches the range of our language. Even in the case of a finite situation in which negative information can be left implicit as it can be extracted from a complete picture of the positive facts, the virtue of a negation operator can be defended in terms of conciseness.

4.1.1 Polarised simple graphs

Syntactically, the most obvious way of discriminating positive facts from negative ones is to attribute a colour, a sign, to every relation occurrence:

4.1.1.1. DEFINITION. [Polarised simple graph] A polarised simple conceptual graph (PSCG) $G = (R, C, E, label, co, sign)$ over a signature Σ is a simple conceptual graph such that every relation vertex is signed: i.e., $G = (R, C, E, label, co)$ is a simple conceptual graph and $sign$, a function from R to $\{+, -\}$.

To avoid some overloading of graphics, positive relation nodes are represented without explicit sign whereas negative ones have a label preceded by a “-”.

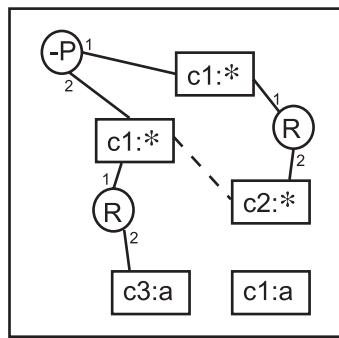


Figure 4.1: A polarised simple conceptual graph

4.1.1.2. EXAMPLE. The graph drawn in Figure 4.1 represents the information “ $c3(a) \wedge c1(a) \wedge \exists x(c1(x) \wedge c2(x) \wedge R(x, a) \wedge \exists y(c1(y) \wedge R(y, x) \wedge \neg P(y, x))$ ”.

There is one exception to the finiteness of simple graphs: the absurd graph. By opposition to the empty graph, which is satisfied by any structure, the absurd graph represents the conjunction of all possible (finite) polarised simple graphs over a given language signature.

4.1.3. DEFINITION. [Absurd graph] G_{\perp} , the absurd graph, is defined as the juxtaposition of all PSCGs over Σ .

4.1.1.1 Embedding semantics

Polarised simple graphs are interpreted in the same structures respecting the language signature as the (positive) simple conceptual graphs (Chapter 3.1.3). For a graph to be true in a structure under an assignment, the tuple of objects corresponding to the arguments of a negated relation node must not belong to the interpretation of the considered relation.

4.1.4. DEFINITION. [Truth of a polarised graph] Let $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_{\mathcal{R}}), \text{arity})$ be a signature, $G = (C, R, E, \text{label}, \text{co}, \text{sign})$ be a PSCG over Σ and $M = (D, F)$ be a Σ -structure,

- $M, f \models G$ if and only if
 1. $\forall c \in C, f(c) \in F(\text{type}(c))$ and
 2. $\forall r \in R$ such that $\text{label}(r) \in \mathcal{R}_n$,
 - if $\text{sign}(r) = +$, then $\langle f(r(1)), \dots, f(r(n)) \rangle \in F(\text{label}(r))$,
 - else ($\text{sign}(r) = -$) it holds that $\langle f(r(1)), \dots, f(r(n)) \rangle \notin F(\text{label}(r))$.
- $M \models G$ iff there exists an assignment f such that $M, f \models G$.
- $G \sqsubseteq H$ iff H is true in every Σ -structure in which G is true.

This interpretation of polarised graphs by direct mapping from the representation to the represented is in agreement with the recognised cognitive characteristics of pictures: their faithfulness which enables the formal meaning to match intuition. To be verified in a structure, negated atoms must fit into the holes like pieces of a jigsaw puzzle.

4.1.1.2 Positivising

Another way of seeing negative relation nodes is to consider that their label belongs to the vocabulary. To apply the ‘‘Positivising’’ technique (see e.g., [BP83] or [Ben88]) to the polarised fragment, we define a meaning-preserving transformation from negative atoms to positive representations.

4.1.5. DEFINITION. Let $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_{\mathcal{R}}), \text{arity})$ be a signature,

- We extend Σ to a signature $\Sigma^+ = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}^+, \leq_{\mathcal{R}^+}), \text{arity}^+)$ such that
 1. $\forall P \in \mathcal{R}, P^- \in \mathcal{R}^+$ and $\text{arity}^+(P^-) = \text{arity}^+(P) = \text{arity}(P)$,
 2. and $\forall P, Q \in \mathcal{R}, (P \leq_{\mathcal{R}^+} Q \text{ iff } P \leq_{\mathcal{R}} Q)$ and $(Q^- \leq_{\mathcal{R}^+} P^- \text{ iff } P \leq_{\mathcal{R}} Q)$
- For a polarised SCG $G = (C, R, E, \text{label}, \text{co}, \text{sign})$ over Σ , its positive form $G^+ = (C, R, E, \text{label}^+, \text{co})$ is the simple conceptual graph over Σ^+ obtained from G by replacing the label of every negative relation node by the corresponding newly introduced label:
 $\forall r \in R$,

1. if $\text{sign}(r) = +$ then $\text{label}^+(r) = \text{label}(r)$ and
2. (ii) if $\text{sign}(r) = -$ and $\text{label}(r) = P$ then $\text{label}^+(r) = P^-$.

Symmetrically, for any simple conceptual graph H over Σ^+ , H^- is the polarised graph obtained by replacing every relation node labelled with P^- by a relation node labelled with P and negatively signed.

- For a given Σ -structure $M = (D, \llbracket \cdot \rrbracket_M)$, we define a structure $M^+ = (D, \llbracket \cdot \rrbracket_{M^+})$ over Σ^+ such that
 1. $\llbracket \cdot \rrbracket_{M^+}$ is equal to $\llbracket \cdot \rrbracket_M$ on \mathcal{I}, \mathcal{C} and \mathcal{R} and
 2. $\forall P \in \mathcal{R}$ of arity n , $\llbracket P^- \rrbracket_{M^+} = D^n \setminus \llbracket P \rrbracket_M$.

Symmetrically, a Σ^+ -structure N is transformed into a Σ -structure N^- such that

$$\forall P \in \mathcal{R}, \llbracket P \rrbracket_{N^-} = \llbracket P \rrbracket_N.$$

We note that relation symbols of the original signature and newly introduced ones are $\leq_{\mathcal{R}^+}$ -incomparable.

On one hand, back and forth translations of graphs preserve all information: the sign of a relation node is just internalised in the label and vice-versa. On the other hand, structure transformations call for a more careful analysis which presents the premisses for the soundness but incompleteness of projection in the polarised fragment (further developed in Chapter 4.1.2).

4.1.6. FACT. Let M be a Σ -structure, N be a Σ^+ -structure, G be a polarised simple graph over Σ and H be a simple graph over Σ^+ ,

1. $G^{+-} = G$ and $H^{-+} = H$
2. M^+ is a proper Σ^+ -structure and N^- is a proper Σ -structure
3. $M \models G \Rightarrow M^+ \models G^+$ and $N \models H \Rightarrow N^- \models H^-$
 provided that $\forall P \in \mathcal{R}, \llbracket P \rrbracket_N \cap \llbracket P^- \rrbracket_N = \emptyset$ and H^- is not absurd.

4. $M^{+-} = M$, but there exists a Σ^+ -structure N such that $N^{-+} \neq N$.

Proof:

1. Graph transformations purely concerns the layout by gathering (or splitting) two kinds of labels attached to relation nodes: sign and relation symbol.
2. To prove that M^+ is a Σ^+ structure, it is sufficient to verify that $\forall P, Q \in \mathcal{R}$, if $P \leq_{\mathcal{R}} Q$ then $\llbracket Q^- \rrbracket_{M^+} \subseteq \llbracket P^- \rrbracket_{M^+}$. Indeed, the interpretation of the remaining symbols is preserved from M . Let $P, Q \in \mathcal{R}$, of arity n such that $P \leq_{\mathcal{R}} Q$. M is a Σ -structure, thus, $\llbracket P \rrbracket_M \subseteq \llbracket Q \rrbracket_M$. As $\llbracket Q^- \rrbracket_{M^+} = D^n \setminus \llbracket Q \rrbracket_M$ and $\llbracket P^- \rrbracket_{M^+} = D^n \setminus \llbracket P \rrbracket_M$, it directly follows that $\llbracket Q^- \rrbracket_{M^+} \subseteq \llbracket P^- \rrbracket_{M^+}$.

Symmetrically, the ordering on relation symbols is directly preserved by the transformation from N to N^- as $\llbracket P \rrbracket_{N^-} = \llbracket P \rrbracket_N$.

3. $M^+ = (D, \llbracket \cdot \rrbracket_{M^+})$ is a Σ^+ -structure of a special form: given a relation symbol $P \in \mathcal{R}$ of arity n and $\vec{d} \in D^n$, either $\vec{d} \in \llbracket P \rrbracket_{M^+}$ or $\vec{d} \in \llbracket P^- \rrbracket_{M^+}$; The classical two-valuation is forced.

Assume for an assignment f that $M, f \models G$. In G^+ , every atomic compound with a relation node labelled with a symbol in \mathcal{R} must be satisfied by M under f ; hence, by M^+ under f as the interpretation of relation symbols in \mathcal{R} is preserved by the structure transformation.

On the other hand, any atomic compound $r(\vec{t})$ in G^+ with $label(r) = P^-$ corresponds to an atomic compound $r'(\vec{t})$ in G with $label(r') = P$ and $sign(r') = -$. $M, f \models G$, thus $f(\vec{t}) \notin \llbracket P \rrbracket_M$. Hence, $f(\vec{t}) \in \llbracket P^- \rrbracket_{M^+}$.

Symmetrically, the interpretation of positive relation symbols is also preserved (i.e., $\forall P \in \mathcal{R}$, $\llbracket P \rrbracket_{N^-} = \llbracket P \rrbracket_N$) and as overvaluation is avoided (i.e., it is not the case that $\vec{t} \in (\llbracket P \rrbracket_N \cap \llbracket P^- \rrbracket_N)$ for any \vec{t} and P), it follows that $\vec{t} \in \llbracket P^- \rrbracket_N$ implies that $\vec{t} \notin \llbracket P \rrbracket_{N^-}$.

4. The chosen structure transformations are based on the preservation of the positive information:
 - (i) $\vec{t} \in \llbracket P \rrbracket_M \Rightarrow \vec{t} \in \llbracket P \rrbracket_{M^+} \Rightarrow \vec{t} \in \llbracket P \rrbracket_{M^{+-}}$ and
 - (ii) $\vec{t} \notin \llbracket P \rrbracket_M \Rightarrow \vec{t} \notin \llbracket P \rrbracket_{M^+} \Rightarrow \vec{t} \notin \llbracket P \rrbracket_{M^{+-}}$
 Hence, $M^{+-} = M$.

On the other hand, “undervaluation” in a Σ^+ -structure is transformed into negative information and “overvaluation” into positive one; hence modifying the original structure:

- (i) if both $\vec{t} \notin \llbracket P \rrbracket_N$ and $\vec{t} \notin \llbracket P^- \rrbracket_N$, then $\vec{t} \notin \llbracket P \rrbracket_{N^-}$, thus $\vec{t} \in \llbracket P^- \rrbracket_{N^{-+}}$
 - (ii) if both $\vec{t} \in \llbracket P \rrbracket_N$ and $\vec{t} \in \llbracket P^- \rrbracket_N$, then $\vec{t} \in \llbracket P \rrbracket_{N^-}$, thus $\vec{t} \notin \llbracket P^- \rrbracket_{N^{-+}}$
- Hence, there exists a Σ^+ -structure N such that $N^{-+} \neq N$

■

From the previous facts, follows a first application of the positive transformation of polarised graphs:

4.1.7. THEOREM. *For G and H two polarised simple conceptual graphs over Σ ,*

$$G^+ \sqsubseteq H^+ \text{ implies that } G \sqsubseteq H$$

Proof: assume that any Σ^+ structure that satisfies G^+ also satisfies H^+ . Let M be a Σ structure which satisfies G , by the previous fact (Fact 4.1.6(3)), N^+ is a Σ^+ structures which satisfies G^+ . Thus, by assumption, $N^+ \models H^+$. Thus, again by Fact 4.1.6(3), $N^{+-} \models H^{+-}$. Hence, by Fact 4.1.6(1 & 4), $N \models H$ ■

Unfortunately the reciprocal of the previous theorem is not true: we will see in Chapter 4.1.2 that the undervaluation case, in the proof of Fact 4.1.6(4), forbids the use of the minimal model of a polarised graph transformed into positive form as a fair representant of the information conveyed by the original polarised graph.

Our next step in manipulating the polarised fragment has become clear. Polarised graphs resemble their simple conceptual graph ancestors: drawings have the same bipartite network structure and are interpreted by the same kind of direct mapping to models. The sole difference resides in the labelling, it is thus legitimate to consider the possibility of solving the consequence decision problem by a special projection taking care of the new kind of labels.

4.1.1.3 Projection

The projection calculus corresponds to the evaluation of a source graph in the canonical model of a target graph. We have seen in the simple graph fragment that the method is complete if the target graph is isomorphic to its canonical model, or in conceptual graph terms, normalised. It seems natural to extend the projection algorithm to polarised simple graphs, by just adding for negated nodes, a constraint on labels which is symmetrical to the usual one for positive relations: while the neighbourhood of a relation node must be preserved in the mapping, a positive relation node is mapped onto another positive one which has a more specialised label, whereas a negative relation node is mapped onto another negative one with a more general label.

4.1.8. DEFINITION. [Projection for PSCGs] Let $G = (C, R, E, label, co, sign)$ and $H = (C', R', E', label', co', sign')$ be two polarised simple conceptual graphs over a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$.

A projection from the source H to the target G is a mapping $\pi : C' \cup R' \rightarrow C \cup R$ such that:

- $\forall c \in C', \pi(c) \in C$ and $type(\pi(c)) \leq_c type'(c)$
- $\forall c \in C',$ if $marker'(c) \in \mathcal{I}$ then $marker(\pi(c)) = marker'(c)$
- $\forall r \in R', \pi(r) \in R$ and
if $sign'(r) = +,$ then $sign(\pi(r)) = +$ and $label(\pi(r)) \leq_{\mathcal{R}} label'(r),$
else $sign(\pi(r)) = -$ and $label'(r) \leq_{\mathcal{R}} label(\pi(r)).$
- $\forall (r, c, i) \in E', (\pi(r), \pi(c), i) \in E$
- $\forall (c_1, c_2) \in co', \pi(c_1) = \pi(c_2)$

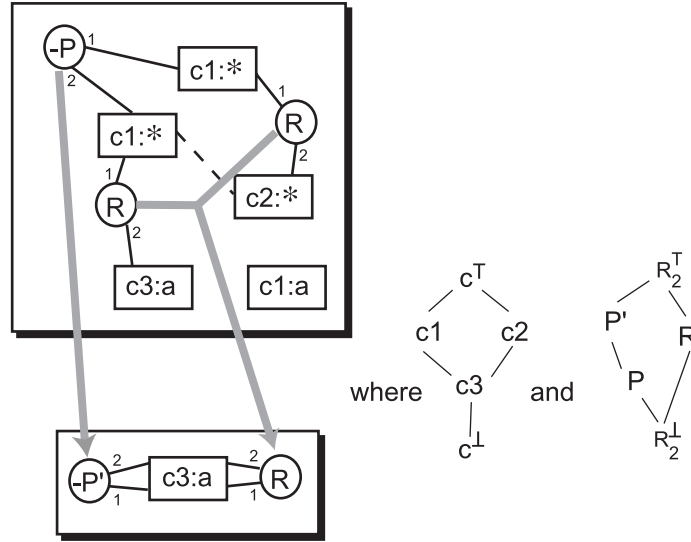


Figure 4.2: A projection with polarised graphs

4.1.9. EXAMPLE. In Figure 4.2, the source graph representing “ $c3(a) \wedge c1(a) \wedge \exists x(c1(x) \wedge c2(x) \wedge R(x, a) \wedge \exists y(c1(y) \wedge R(y, x) \wedge \neg P(y, x))$)” is a logical consequence of the target graph representing “ $c3(a) \wedge \neg P'(a, a) \wedge R(a, a)$ ”, given the information that $c3$ is a subconcept of $c1$ and $c2$ and that the relation P is a subrelation of P' .

Equivalently presented, there is a projection between two polarised simple graphs iff there is a projection between their respective positive representation:

4.1.10. FACT. For two polarised simple conceptual graphs G and H over Σ , there is a projection from H to G if and only if there is a projection, with respect to the orders in Σ^+ , from H^+ to G^+ .

Proof: it is immediate to verify that the projection-constraint on negative relation nodes is transferred to the signature order over the negated relational vocabulary. ■

4.1.11. COROLLARY (SOUNDNESS OF PROJECTION FOR POLARISED GRAPHS).
For G and H two polarised simple conceptual graphs over Σ ,

if there exists a projection from H to G , then $G \sqsubseteq H$

Proof: if there exists a projection from H to G , then there exists a projection from H^+ to G^+ by Fact 4.1.10. Thus $G^+ \sqsubseteq H^+$ by the soundness of projection on simple conceptual graphs (Theorem 3.2.6). Hence, $G \sqsubseteq H$ by Theorem 4.1.7. ■

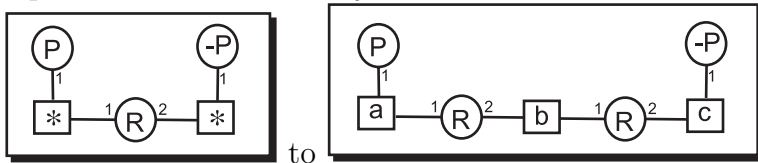
Soundness is not sufficient. We also aim at a complete method, but atomic negation has introduced a form of disjunctive information which prevents the “all in one mapping” method to capture all consequences.

4.1.2 Insufficiencies of projection

We already know from the fragment of positive simple graphs, that the target graph is required to appear in normal form (i.e., that distinct concept nodes representing a single object must be merged). Fortunately, this normal form can be obtained in polynomial time.

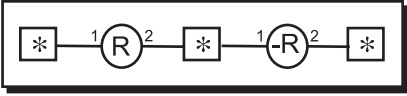
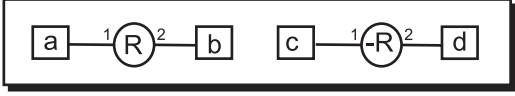
Normalisation put aside, there is still a source of incompleteness which is linked to the fact that the calculus has to cope with tautologies of the form “ $P(\vec{t}) \vee \neg P(\vec{t})$ ”.

Take, for instance, the sentence $\varphi_1 = P(a) \wedge R(a, b) \wedge R(b, c) \wedge \neg P(c)$. It entails the sentence $\varphi_2 = \exists x \exists y (P(x) \wedge \neg P(y) \wedge R(x, y))$ because, from φ_1 , either it holds that $P(b)$ and in this case the pair of variables (x, y) can be unified with (b, c) , or it holds that $\neg P(b)$ and in this case (x, y) can be unified with (a, b) . From the viewpoint of model-checking, every model of φ_1 satisfies either $P(b)$ or $\neg P(b)$. However, the conceptual graph formalism does not allow for an explicit representation of this disjunctive information and there is no projection from



The following second counter-example to the completeness of projection involves a target graph where positive and negative information are dispatched

in different connected compounds: $R(a, b) \wedge \neg R(c, d) \models_{FOL} \exists x \exists y \exists z (R(x, y) \wedge$

$\neg R(y, z))$ whereas the graph  cannot be projected onto the graph .

To take an alternative perspective on the problem, we can observe projection through the previously presented positive transformation of polarised graph (Chapter 4.1.1.2). We could be tempted to apply the reciprocal reasoning to the proof of Theorem 4.1.7:

consider G^+ and H^+ the positive forms of the respective polarised graphs G and H . Assume that $G \sqsubseteq H$ and let N be the minimal model of G^+ .

$N^- \models G^{+-}$, thus $N^- \models G$, by Fact 4.1.6(3 & 1).

By assumption, $N^- \models H$, thus, again by Fact 4.1.6(3), $N^{-+} \models H^+$.

Unfortunately, N^{-+} is not a sub-model of the original structure N , hence we cannot conclude that $N \models H^+$.

Stronger, we can conclude from the previous counter examples that it *does not hold* that for G and H two polarised simple conceptual graphs, $G \sqsubseteq H \Rightarrow G^+ \sqsubseteq H^+$.

So, are we forced to completely abandon our project of mapping polarised graphs? No, as we will see, constraints on the interaction of negations and existential quantifiers allow us to correctly base our reasoning technique on projection and even in a tractable way under guarded conditions.

4.1.3 Discriminated polarised simple graphs

We have already noticed the relevance of the notion of connected compound to the projection calculus for positive simple conceptual graphs: a connected compound is a piece of information which is independent from the remaining parts. Therefore, if each connected compound of a source graph can be projected on a target graph, then we obtain a projection of the whole source graph by taking the union of all these “small” projections. This idea can be applied to polarised graphs in order to treat separately positive and negative pieces of information.

A polarised simple graph is called discriminated if none of its relation nodes shares a concept node neighbour with a relation node of the opposite sign.

4.1.12. DEFINITION. [Discriminated simple graphs] A polarised simple graph $G = (R, C, E, label, co, sign)$ is discriminated if $\forall r \in R$ such that $sign(r) = +$ it holds that $\forall (r, c, i) \in E, (r', c, j) \in E$ implies that $sign(r') = +$.

Because splitting an instantiated concept node into many copies preserves the meaning of a graph (the distinct copies denote the same object in a model),

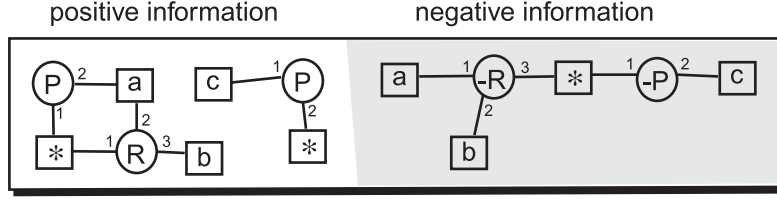
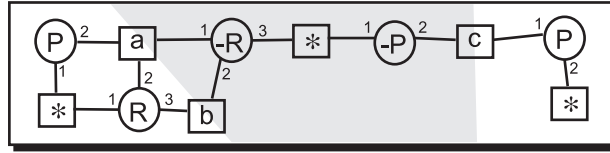


Figure 4.3: A discriminated graph

a polarised graph can be transformed into an equivalent discriminated graph if all paths of edges between two relation nodes of opposite sign include some instantiated concept nodes. For instance, the graph in Figure 4.3 is equivalent to the following non-discriminated graph:



Obviously, in terms of formulae, a discriminated graph corresponds to a conjunction of two sentences, one with only positive occurrences of relations and the other one with only negated occurrences of relations.

4.1.4 Completeness and tractability

4.1.4.1 Discriminated graphs and the splitting of labour

Before getting into the details of the desired completeness proof for the discriminated fragment, let us come back to the simple proof scheme proposed in Chapter 4.1.2. Failing to show that $G \sqsubseteq H \Rightarrow G^+ \sqsubseteq H^+$, we identified a culprit as a missing identity $N^{-+} = N$, or more precisely, a missing inclusion $N^{-+} \subseteq N$, as the satisfaction of a positive graph is preserved under model expansion.

In this section, we can still not prove that the minimal model M_{G^+} of G^+ is preserved under $-+$ transformations, but we can use the fact that the discrimination in the conclusion graph, $H^+ = H_{pos}^+ \oplus H_{neg}^+$, allows us to consider independently positive and negative information and gather the results at the end. We will prove that if the conclusion H contains either only positive facts or only negative facts, then we can exhibit a sub-model $N \subseteq M_{G^+}$ constructed from $(M_{G^+})^{-+}$ such that $N \models H^+$. Hence $(G \sqsubseteq H) \Rightarrow (M_{G^+} \models H_{pos}^+ \oplus H_{neg}^+) \Rightarrow (G^+ \sqsubseteq H^+)$.

We have seen that the transformation from a Σ^+ -structure N to the Σ^- -structure N^- favours truly positive knowledge: $\forall P \in \mathcal{R}, \llbracket P \rrbracket_{N^-} = \llbracket P \rrbracket_N$. To restore the balance, we shall use a second transformation which favours truly negative knowledge:

4.1.13. DEFINITION. [\checkmark -transformation] Given a signature Σ and a Σ^+ -structure $N = (D, \llbracket \cdot \rrbracket_N)$, we define $N^{\checkmark} = (D, \llbracket \cdot \rrbracket_{N^{\checkmark}})$ such that $\forall P \in \mathcal{R}$ of arity $n, \forall \vec{t} \in D^n$,

$$\vec{t} \notin \llbracket P \rrbracket_{N^{\checkmark}} \quad \text{iff} \quad \vec{t} \in \llbracket P^- \rrbracket_N$$

4.1.14. FACT. Given a signature Σ and a Σ^+ -structure N ,

1. N^{\checkmark} is a proper Σ -structure
2. $\forall P \in \mathcal{R}$, if $\llbracket P \rrbracket_N \cap \llbracket P^- \rrbracket_N = \emptyset$, then $\llbracket P \rrbracket_N \subseteq \llbracket P \rrbracket_{N^{\checkmark}}$
3. $N \models H \Rightarrow N^{\checkmark} \models H^-$
provided that $\forall P \in \mathcal{R}$, $\llbracket P \rrbracket_N \cap \llbracket P^- \rrbracket_N = \emptyset$ and H^- is not absurd.

Proof:

1. We must verify that the chosen ordering on relation symbols is preserved by the \checkmark -transformation. For $P \leq_{\mathcal{R}} Q$, suppose that there exists $\vec{t} \in \llbracket P \rrbracket_{N^{\checkmark}}$ such that $\vec{t} \notin \llbracket Q \rrbracket_{N^{\checkmark}}$. By Definition 4.1.13, $\vec{t} \in \llbracket Q^- \rrbracket_N$ and, as N is a proper Σ^+ -structure, $\llbracket Q^- \rrbracket_N \subseteq \llbracket P^- \rrbracket_N$, thus $\vec{t} \in \llbracket P^- \rrbracket_N$. Again, by Definition 4.1.13, it follows the contradiction: $\vec{t} \notin \llbracket P \rrbracket_{N^{\checkmark}}$.
2. If N does not contain any overvaluation, then truly positive facts are preserved by the \checkmark -transformation: suppose that $\vec{t} \in \llbracket P \rrbracket_N$ and $\vec{t} \notin \llbracket P \rrbracket_{N^{\checkmark}}$, then, by Definition 4.1.13, $\vec{t} \in \llbracket P^- \rrbracket_N$. Thus, from $\llbracket P \rrbracket_N \cap \llbracket P^- \rrbracket_N = \emptyset$, it follows the contradiction that $\vec{t} \notin \llbracket P \rrbracket_N$.
3. We have just proved that positive information is preserved by the transformation. On the other hand, $\forall P \in \mathcal{R}$ of arity n , $(D^n \setminus \llbracket P \rrbracket_{N^{\checkmark}}) = \llbracket P^- \rrbracket_N$, thus the satisfaction of negative facts is also preserved by the transformation.

■

To prove the completeness theorem, i.e. $(G \sqsubseteq H) \Rightarrow (G^+ \sqsubseteq H^+)$, we have learned from Chapter 2 and Chapter 3 that it sufficient to prove that $G \sqsubseteq H$ implies that $M_{G^+} \models H^+$, as G^+ and H^+ are simple conceptual graphs (over an extended vocabulary). Let us briefly recall the construction of the minimal model of the simple graph G^+ .

Without loss of generality, we assume that G is not absurd, thus $\llbracket P \rrbracket_{M_{G^+}} \cap \llbracket P^- \rrbracket_{M_{G^+}} = \emptyset$ for any relation symbol P . The universe of M_{G^+} consists in the set of concept nodes in the normal form of G^+ , or equivalently, the set of concept node labels in the graph obtained from G^+ by replacing each coreference class

by a witness. For any relation symbol P , $\llbracket P \rrbracket_{M_{G^+}} = \{\vec{t}/\exists Q \leq_{\mathcal{R}} P, Q\vec{t} \in G^+\}$ and $\llbracket P^- \rrbracket_{M_{G^+}} = \{\vec{t}/\exists Q \geq_{\mathcal{R}} P, Q^-\vec{t} \in G^+\}$.

It directly follows from the model construction that the minimal model of G^+ satisfies G under both transformations $-$ and \checkmark .

4.1.15. FACT.

$$(M_{G^+})^{\checkmark} \models G \text{ and } (M_{G^+})^- \models G$$

Proof: $M_{G^+} \models G^+$ by construction (Fact 3.2.4). As G is not absurd, M_{G^+} is not over-validated. Thus, $(M_{G^+})^- \models G^{+-}$ (by Fact 4.1.6), $(M_{G^+})^{\checkmark} \models G^{+-}$ (by Fact 4.1.14) and $G^{+-} = G$ (by Fact 4.1.6). ■

We can now employ the \checkmark and $-$ transformations to prove the following lemma:

4.1.16. LEMMA (COMPLETENESS WITH A UNIFORMLY SIGNED CONCLUSION).
*Let Σ be a signature, G be a non absurd polarised simple conceptual graph over Σ and H be a polarised simple conceptual graph whose **relation nodes all have the same sign**, it holds that*

$$(G \sqsubseteq H) \Rightarrow (G^+ \sqsubseteq H^+)$$

Proof: Assume that $G \sqsubseteq H$.

- Let H be negative (i.e., every relation node in H is negatively signed or, equivalently, every relation node in H^+ is labelled with a negatively signed relation symbol).

We have just proved that $(M_{G^+})^{\checkmark} \models G$.

Hence, by assumption $(M_{G^+})^{\checkmark} \models H$.

Thus, by Fact 4.1.6(3), $(M_{G^+})^{\checkmark+} \models H^+$

Furthermore, we can verify that $\forall P \in \mathcal{R}$, $\llbracket P^- \rrbracket_{M_{G^+}} = \llbracket P^- \rrbracket_{(M_{G^+})^{\checkmark+}}$:

$\vec{t} \in \llbracket P^- \rrbracket_{M_{G^+}}$ iff $\vec{t} \notin \llbracket P \rrbracket_{(M_{G^+})^{\checkmark}}$ (by Definition 4.1.13)

iff $\vec{t} \in \llbracket P \rrbracket_{(M_{G^+})^{\checkmark+}}$ (by Definition 4.1.5).

So, $(M_{G^+})^{\checkmark+}$ and M_{G^+} coincide on the interpretation of negated relation symbols.

Furthermore, as H be negative, we can safely eliminate from the model the sets of tuples in the interpretation of positive relation symbols:

let N have the same universe as $(M_{G^+})^{\checkmark+}$ and $\forall P \in \mathcal{R}$, $\llbracket P \rrbracket_N = \emptyset$ and $\llbracket P^- \rrbracket_N = \llbracket P^- \rrbracket_{(M_{G^+})^{\checkmark+}}$. It holds that $N \models H^+$ and that $N \subseteq M_{G^+}$.

Hence $M_{G^+} \models H^+$ as the satisfaction of a simple graph is preserved under model expansion.

- Let H be positive (it is a simple conceptual graph). Similarly to the negative case, but using the $-$ transformation, we can construct a submodel of the minimal model of G^+ that satisfies H^+ :

$$(M_{G^+})^- \models G \Rightarrow (M_{G^+})^- \models H \Rightarrow (M_{G^+})^{-+} \models H^+$$

let N have the same universe as $(M_{G^+})^{-+}$ and $\forall P \in \mathcal{R}, \llbracket P \rrbracket_N = \llbracket P^- \rrbracket_{(M_{G^+})^{-+}}$ and $\llbracket P^- \rrbracket_N = \emptyset$. It holds that $N \models H^+$ and that $N \subseteq M_{G^+}$. Hence, $M_{G^+} \models H^+$.

■

4.1.4.2 Projection completeness

We are now ready to prove the completeness of projection from a discriminated graph to a polarised one in normal form. We can apply the lemma to each part of a discriminated graph:

4.1.17. COROLLARY. *For G and H two polarised simple conceptual graphs over Σ such that $G \neq G_\perp$ and H is discriminated,*

$$G \sqsubseteq H \Rightarrow G^+ \sqsubseteq H^+$$

Proof: H is discriminated, so we can equivalently rewrite it as the juxtaposition of two polarised simple graphs H_{pos} and H_{neg} such that all relation nodes in H_{pos} are positive and all relation nodes in H_{neg} are negative; $H = H_{pos} \oplus H_{neg}$.

Assume that $G \sqsubseteq H$. From Lemma 4.1.16, we have $\exists f, g$ assignments such that $M_{G^+}, f \models H_{pos}^+$ and $M_{G^+}, g \models H_{neg}^+$. Hence, $M_{G^+}, (f \cup g) \models H_{pos}^+ \oplus H_{neg}^+$ as f and g coincide on the interpretation of individual marker (by definition of an assignment) and H_{pos}^+ and H_{neg}^+ do not share any existential concept node. Hence $M_{G^+} \models H^+$. ■

The purpose of normalisation is twofold: first, as for positive graphs, concept nodes representing a single object are merged in order to obtain a graph isomorphic to its minimal model (if such a model exists). Furthermore, if the graph contains contradictory information, then it is replaced by the absurd graph. This operation guarantees that any graph can be projected onto the normal form of a graph representing contradictory information (more specifically, any graph can be mapped onto its copy which is part of the absurd graph).

4.1.18. DEFINITION. [Normal form of a polarised graph] $Norm(G)$, the normal form of a PSCG $G = (C, R, E, label, co, sign)$ is defined in two steps as follows:

1. let $G' = (C', R, E', label', sign')$ be the PSCG obtained from G by merging all occurrences of concept nodes having the same individual marker and, for each coreference equivalence class, merging all elements of the class. After merging two nodes, the resulting one is labelled with the meet of the concept label of the original nodes.
2. if in G' there are two relation nodes r and r' in R such that $label'(r') = p$, $sign'(r') = +$, $label'(r) = q$, $sign'(r) = -$, $p \leq_{\mathcal{R}} q$ and $\forall (r, c, i) \in E, (r', c, i) \in E$, then $Norm(G) = \perp$; otherwise $Norm(G) = G'$.

4.1.19. THEOREM (COMPLETENESS OF PROJECTION). *For a polarised graph in normal form G and a discriminated graph H ,*

$$G \sqsubseteq H \text{ iff there exists a projection from } H \text{ to } G$$

Proof: we have already considered the “if direction” in Corollary 4.1.11.

Reciprocally, if G is absurd, then H is a subgraph of G (as G is in normal form). Hence, there is projection from H to G . Otherwise, $G^+ \sqsubseteq H^+$ by Corollary 4.1.17 and building on the completeness of projection for simple conceptual graphs (Theorem 3.2.6), we can conclude that there is a projection from H^+ to G^+ . Hence there is a projection from H to G by Fact 4.1.10. ■

4.1.4.3 Tractability harvest

We can directly benefit from the complete transposition of discriminated polarised graphs into the simple conceptual graph fragment.

4.1.20. COROLLARY. • *The empty graph is the sole valid polarised graph.*

- *Deciding satisfiability of a polarised graph is in polynomial time.*
- *$G \subseteq H$ is NP-complete, for G and H two polarised graphs over a common signature and such that H is discriminated.*

Proof:

- A structure with empty universe can only satisfy the empty graph which is by definition also satisfied by any structure (no required embedding).
- As in the corresponding textual fragment (Proposition 2.6.2), we can verify that no opposite relation nodes share the same arguments by a simple check through a polarised graph: a brutal procedure that goes through the whole graph for each relation node requires a quadratic time.
- By Theorem 4.1.17, the polarised consequence problem with a discriminated conclusion is in NP. Indeed, the translation of a polarised graph into its positive

form is only a linear renaming. The NP lower bound is the one of subsumption in simple graphs as any simple graph is a polarised one. ■

We can exploit even further the correspondence to (positive) simple graphs:

4.1.21. DEFINITION. [Guarded discriminated polarised graphs] A discriminated polarised graph is guarded if it is the result of juxtaposing two polarised graphs G and H such that (i) all relation nodes in G are positive, (ii) all relation nodes in H are negative and (iii) G and H are guarded simple graphs if we make abstraction of the signs.

4.1.22. COROLLARY. *Given two polarised graphs G and H over a common signature, $G \sqsubseteq_{NCG} H$ is decidable in polynomial time if H is guarded and discriminated.*

Proof: Immediate consequence of Theorem 4.1.17 and Theorem 3.3.7 as the transformation of the problem into the (guarded) fragment of simple graphs is polynomial. ■

4.1.5 Concluding remarks

The lesson we can draw from this first extension of the simple graph fragment is that a very simple additional language feature such as atomic negation can already compromise the enterprise of basing reasoning on a single picture comparison.

Nevertheless, we have been able to identify a constraint on the disposition of negations in a picture which defines a fragment of polarised graphs where the projection technique safely applies. Discriminated polarised graphs are representations built from the juxtaposition of proper polarised graphs which contain either only positive facts or only negative ones.

Furthermore, by transposing back the problem into the guarded fragment of simple graphs, we have reinforced the relevance of tree-like structures for the computational tractability of the consequence problem.

We should note that to our disadvantage, disjunctive forms that, by nature, do not lend themselves easily to pictorial representations, are among the most extensively decorticated fragments of FOL in the literature. For instance, languages based on clausal forms, such as Horn-fragments, have proved their computational efficiency ([BGG97] provides a very extensive survey of complexity results and bibliographic references for such clausal languages). Also in database theory, algorithmic solutions to the query containment problem in different languages of conjunctive queries including forms of negation have been proposed; see e.g. [LS93], [LMSS93], [LS95] or [FTU99].

Back to the graphs, we may not have yet played all our cards on projection with atomic negation: the implicit disjunctive information hidden in polarised representations could possibly be handled by combining alternative simultaneous projections.

This idea of decomposing the proof of a disjunction into parallel subproofs is also the core of a classical type of calculus in logic, semantic tableaux. Leaving open the question of the applicability of simultaneous projections, we will examine how tableaux can be combined to projections in a wider graph fragment.

4.2 Full classical negation

This section is taken up with a rather different angle than the previous ones. Indeed, by considering a conceptual graph language which is equivalent to FOL, our goal cannot be tractability (and even decidability) anymore, but consistently with the former fragments, our focus can still be directed towards graphical methods that naturally apply to the pictorial representations.

We will propose a proof method interlacing decomposition steps by tableau rules¹ applied to complex graphs and projection steps on sufficiently simple subgraphs. For its generic character, the calculus is not aimed as such at a direct efficient implementation, but rather at being easily reinforced by heuristics that abound in both the tableau and the graph homomorphism literature. Such a fine tuning should be application driven and take place when specific forms of graphs are considered.

It should be emphasised that despite their origins in textual symbolic logics, tableaux offer an overwhelming position to a graphical aspect: trees serve the purpose of collecting in a compact way the deterministic construction of alternative (counter) models.

4.2.1 Conceptual Graphs

In a preliminary stage, we define a conceptual graph representation of negation inspired by Peirce's notation in existential graphs (e.g., [Rob73]) and also very similar to the box style of DRS syntax (e.g., [KR93]).

4.2.1.1 Negation box syntax

The negation of a graph is represented by a closed line surrounding the graph (or in Peirce's terms, a line cutting the portion of the sheet where the graph is drawn). Furthermore, closed lines are not crossing each other. Hence they define

¹See e.g., [Smu68] for a very pleasant presentation of the method or [Fit90] for an implementation oriented introduction. The proceedings of the annual tableau conference also form a rich knowledge base of application oriented techniques.

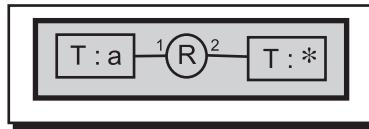
an order, a tree structure, on the nesting of delimited portions of the assertion sheet. Coreference links are required to respect this order.

4.2.1. DEFINITION. [Conceptual graphs] A Conceptual Graph (CG) over a signature Σ , is a set of entries strengthened by a coreference equivalence relation on existential concept nodes, $G = (\{G_1, \dots, G_{n_{(1 \leq i \leq n)}}\}, co)$, such that the following holds:

1. an entry is
 - (a) either simple if it is a simple conceptual graph (possibly the empty graph G_\emptyset) over Σ :
 $G' = (R', C', E', label', co')$ where co' is the identity relation on C'_* ;
 - (b) or boxed if it is of the form $\boxed{G'}$ where G' is a CG over Σ
2. G_1 is simple.
3. if two concept nodes, c and c' , are labelled with the same object name, $m \in \mathcal{I}$, or are co -equivalent, then they are also labelled with the same concept type $t \in \mathcal{C}$
4. the coreference relation co extends the coreference relation of the n entries: $c \equiv_{co} c'$ implies that
 - (a) either the two nodes are already coreferent in a single entry $G_{i_{(1 \leq i \leq n)}}$ in which they occur
 - (b) or there exists an existential concept node c'' in the simple entry G_1 such that $c \equiv_{co} c' \equiv_{co} c''$.

$CG(\Sigma)$ denotes the set of all finite conceptual graphs with respect to the signature Σ . Abusing the former notation for simple graphs, we call G_\emptyset the empty conceptual graph: viz., the graph whose single entry is the empty SCG.

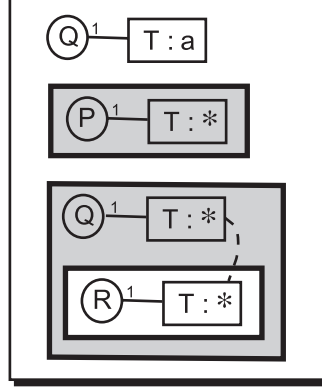
4.2.2. EXAMPLE. To emphasise the nesting of boxes, we use two different background tints for the subgraphs surrounded by an odd number of boxes and those surrounded by an even number of boxes.



The simple entry is the empty graph. The conceptual graph represents the negation of the information conveyed by the enclosed simple graph:

$$\neg(\exists x(Tx \wedge Ta \wedge Rax))$$

As in simple graphs, coreference is represented by a covering set of dashed edges:



This conceptual graph represents

$$Ta \wedge Qa \wedge \neg(\exists x(Tx \wedge Px)) \wedge \neg(\exists y(Ty \wedge Qy \wedge \neg(Ty \wedge Ry)))$$

or equivalently (in classical FOL)

$$Ta \wedge Qa \wedge \forall x(Tx \rightarrow \neg Px) \wedge \forall y((Ty \wedge Qy) \rightarrow Ry)$$

Note that the three existential concept nodes cannot form a coreference equivalence class as they belong to different entries and cannot find an anchor concept node in the simple entry of the whole CG.

This last graph exemplifies the fact that a finite support for the vocabulary can now directly be expressed in the graph language. Nevertheless, we choose to define CGs over hierarchical signatures to emphasise the role of the simple conceptual graph building block, both syntactically and as part of a CG-calculus.

Symptomatic of the textual representation of conceptual graph pictures, we need additional and somehow overloaded formal definitions to support our set-oriented exposition of the framework.

4.2.3. DEFINITION. [Degree, independent and dominant declarations] For any Conceptual Graph, $G = (\{G_1, \dots, G_{n(1 \leq n)}\}, co)$.

- $simple(G)$ denotes the SCG G_1 and $complex(G)$ the set of complex entries of G . G is called simple if it contains no complex entry: $G = \{simple(G)\}$.
- To facilitate the notations, nodes, edges and labels are recursively uplifted from entries to the whole graph:

$$\forall X \in \{C, R, E, label\}, X_G = \bigcup_{1 \leq i \leq n} X_{G_i}$$

where for $X \in \{C, R, E, label, co\}$, on a complex entry $G'' = \boxed{G'}$, $X_{G''} = X_{G'}$ and on a SCG $G' = (C', R', E', label', co')$, $X_{G'} = X'$.

- Degree of a conceptual graph: if G is simple then $degree(G) = 0$, else $degree(G)$ is the sum of the degrees of the complex entries in G . The degree of a complex entry $\boxed{G'}$ is $1 + degree(G')$.
- Declarations are existential concept nodes in the simple entry of a conceptual graph. They are partitioned into:
 1. independent declarations: $IDec(G)$ is the set of existential concept nodes in the simple entry of G , which are not coreferent to other distinct concept nodes.

$$IDec(G) = \{c \in simple(G) / marker_G(c) = * \ \& \ class_{co}(c) = \{c\}\}$$

2. dominant declarations: the declarations which bind other concept nodes deeper in the nesting of entries.

$$DDec(G) = \{c \in simple(G) / marker_G(c) = * \ \& \ class_{co}(c) \neq \{c\}\}$$

$$Dec(G) = IDec(G) \cup DDec(G) = \{c \in simple(G) / marker_G(c) = *\}$$

- it is convenient to transform an entry into a proper conceptual graph. $Graph(G_1) = (\{G_1\}, co_{G_1})$ and $graph(G_{i(1 < i \leq n)}) = (\{G_\emptyset, G_i\}, co_{G_i})$.
When it is clear from the context, we often lighten notations by writing \boxed{G} instead of $graph(\boxed{G})$.

4.2.1.2 Interpretation of negated graphs

Extending the embedding semantics of simple graphs, it is natural to interpret a negated graph by testing whether its content can be mapped on part of a model. We first need to define a way to convey recursively in the nesting of negation boxes successful partial mappings: assignments. A partial assignment of a conceptual graph in a structure extends the interpretation function of individual markers and relation symbols, but it can remain undefined for some existential concept nodes:

4.2.4. DEFINITION. [Partial Assignments]

For a signature $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_R), arity)$, a CG G over Σ and a Σ -structure $M = (D, F)$, a partial assignment of G in M is a partial function from the concept nodes in G to D such that $\forall c, c' \in C_G$, both:

1. if $marker(c) \in \mathcal{I}$ then $f(c) = F(marker(c))$ and $f(c) \in F(type(c))$
(hence, $f(c)$ is defined on all instantiated concept nodes)
2. if $c \equiv_{co} c'$ and $f(c)$ is defined, then it holds that $f(c') = f(c) \in F(type(c))$.

The empty assignment of G in M is the unique partial assignment, f , of G in M such that $\forall c \in C_G$, $marker(c) = *$ implies that $f(c)$ is undefined.

The meaning of a conceptual graph can now be recursively defined from the meaning of simple sub-graphs:

4.2.5. DEFINITION. [Truth of a CG] Let $\Sigma = (\mathcal{I}, (\mathcal{C}, \leq_c), (\mathcal{R}, \leq_{\mathcal{R}}), arity)$ be a signature, $G = (\{G_1, \dots, G_{n_{(1 \leq n)}}\}, co)$ be a CG over Σ and $M = (D, F)$ be a Σ -structure.

Let f be a partial assignment of G in M .

- $M, f \models G$ iff there exists a partial assignment g of G in M such that:
 1. g extends f and the restriction of g to $simple(G)$ is an assignment of $simple(G)$ in M (cf. Definition 3.1.7) and
 2. $M, g \models simple(G)$ (cf. Definition 3.1.8) and
 3. if $n > 1$, then for every complex entry of G , $G_{i_{(1 < i \leq n)}} = \boxed{G'}$, it is **not** the case that $M, g \models G'$.
- $M \models G$ iff $M, f \models G$ where f is the empty assignment of G in M .
- G is satisfiable iff there exists a Σ -structure, M , such that $M \models G$.
- G is valid iff for every Σ -structure, M , it holds that $M \models G$.
- A set of conceptual graphs is interpreted as the conjunction of its elements. Let S be a set of CGs, $M, f \models S$ iff $\forall s \in S (M, f \models s)$.

4.2.1.3 Translation to FOL

To provide an alternative and traditional view on the meaning of conceptual graphs, the Φ -translation of the previous chapter is extended to negation boxes. A notion of substitution is used to propagate, inside boxes, the translation of $*$ -markers by variables. In other words, substitutions are the syntactic side of the former assignments.

4.2.6. DEFINITION. [Substitutions] Let G be a conceptual graph, c be an existential concept node in G , m be a term and t be a concept type.

- $G[c/m]$ is the graph G in which the marker of every concept node coreferent to c has been replaced by m .
- $G[c/(t, m)]$ is the graph G in which the label of every concept node coreferent to c has been replaced by (t, m) .

In both cases, the modified concept nodes are eliminated from the domain of the coreference equivalence relation.

As conceptual graphs with variable markers have not been defined, $G[c/m]$ is a proper conceptual graph iff $m \in \mathcal{I}$.

4.2.7. DEFINITION. [Extension of Φ to CGs] Let G be a conceptual graph,

1. We first define a translation of concept node markers at the global level of the whole graph. *term* is a function which associates to each concept node in G , a term such that

- (a) $\forall c \in \{x/x \in C_G \ \& \ \text{marker}_G(x) \in \mathcal{I}\}$, $\text{term}(c) = \text{marker}(c)$ and
- (b) $\forall c, c' \in \{x/x \in C_G \ \& \ \text{marker}_G(x) = *\}$, $\text{term}(c) = \text{term}(c') \in VAR$ iff $c \equiv_{co} c'$.

2. We then recursively translate the entries in two steps as follows:

- (a) in order to shortcut the quantification of variables in Definition 3.1.16, the chosen variable is substituted to the *-marker of every concept node which is “bound by a quantifier occurring in the simple entry of the graph”: let $\{c_1, \dots, c_n\} = Dec(G)$,

$$G' = G[c_1/\text{term}(c_1)] \dots [c_n/\text{term}(c_n)]$$

- (b) let $\{x_1, \dots, x_n\} = \text{term}(Dec(G))$,

$$\Phi(G) = \exists x_1 \dots \exists x_n (\Phi(\text{simple}(G')) \wedge \bigwedge_{\boxed{G''} \in \text{complex}(G')} \neg(\Phi(G'')))$$

We have presented the translation to FOL as a sidetrack in our tour through conceptual graph fragments, a guideline for anchoring graphical items to their corresponding notions in a well-established framework. Therefore, we will skip both the proof that the translation is in agreement with the former truth definition and the exposition of a reciprocal translation from first-order logic to conceptual graphs that would be required for validating the equivalence of the fragments. Instead, we will focus on the promised proof method that directly manipulates the graphical syntactic items.

4.2.2 Combining tableaux with projections

A keyword of tableau systems is again simplicity: a tableau is model construction by successive decomposition of an input into smaller pieces; hence, the analytical qualification of the method.

These decompositions are guided by the form of the input representations at each stage. Therefore, to any conceptual graph is associated a type depending on the underlying logical operator which is dominating the representation:

4.2.8. DEFINITION. [Graph types] Let $G = (\{G_1, \dots, G_{n(1 \leq n)}\}, co)$ be a conceptual graph. We distinguish three exclusive types of graphs:

1. G is of type α if one of the following conditions holds:
 - (a) $n = 1$ and $Dec(G) \neq \emptyset$;
 G is a simple graph with existential nodes
 - (b) $n = 2$ and $simple(G) \neq G_\emptyset$;
 G is a conjunction of a non-empty simple graph and the negation of a graph.
 - (c) $n > 2$;
 G is a conjunction of non-empty entries.
2. G is of type β if $n = 2$ and $simple(G) = G_\emptyset$ and $Dec(G') \neq \emptyset$ where $G_2 = \boxed{G'}$;
 G is the negation of a graph with existential nodes
3. G is of type χ if one of the following conditions holds:
 - (a) $n = 1$ and $Dec(G) = \emptyset$;
 G is a completely instantiated simple graph
 - (b) $n = 2$ and $simple(G) = G_\emptyset$ and $Dec(G') = \emptyset$ where $G_2 = \boxed{G'}$;
 G is the negation of a completely instantiated simple graph

It is straightforward to verify that the classification defines a partition of the set of all conceptual graphs over a given support.

4.2.9. FACT. Any conceptual graph is of one and only one of the three types.

4.2.2.1 Decomposition rules

The model construction proceeds by transforming an input graph into a disjunction of conjunctions preserving the satisfiability of the input. The disjunctive form is captured as a tree which represents the disjunction of its branches, while each branch is the conjunction of all nodes occurring on it.

A graph of type α is basically an existentially quantified conjunction of subgraphs. It is decomposed as follows: first, dominating existentially quantified concept nodes, which may be coreferent to other concept nodes in the different subgraphs, are replaced by witnesses (i.e., new constants). Then, the different conjuncts are split along a branch.

By opposition, a graph of type β is the negation of an α -graph or, in classically equivalent terms, a universally quantified disjunction of subgraphs. Before splitting the disjuncts, the concept nodes corresponding to dominant universal quantifiers are instantiated. Then, the disjunction can be represented by a branching in the tree.

4.2.10. DEFINITION. [Tableaux] A tableau is a tree whose nodes are occurrences of CGs. The tableau \mathcal{T} may be extended if one of the following two cases applies.

α : an α -type graph $\alpha = (\{\alpha_1, \dots, \alpha_{n(1 \leq n)}\}, co)$ occurs on the branch B_H from the root to a leaf H in \mathcal{T} .

Let X_α be a set of concept nodes in $simple(\alpha)$ such that $DDec(\alpha) \subseteq X_\alpha \subseteq Dec(\alpha)$.

Let Θ_α be a substitution which associates a new object name to every concept node in X_α ;

i.e. $\forall x \in X_\alpha$ it holds that (i) $\Theta_\alpha(x) = m \in \mathcal{I}$ and (ii) m does not occur in \mathcal{T} and (iii) $\forall y \in X_\alpha (x \neq y \rightarrow \Theta_\alpha(x) \neq \Theta_\alpha(y))$.

For $1 \leq i \leq n$, we define $\alpha'_i = graph(\alpha_i)[\Theta_\alpha]$.

We may adjoin successively $\alpha'_1, \dots, \alpha'_n$ such that α'_1 is the sole successor of H and $\alpha'_{2 \leq i \leq n}$ is the sole successor of α'_{i-1} (if $n = 1$, then α'_1 is the sole successor of H).

β : a β -type graph $\beta = (\{G_\emptyset, \beta_2 = \boxed{\beta'}\}, co)$ occurs on the branch B_H from the root to a leaf H in \mathcal{T} . $\beta' = (\{\beta'_1, \dots, \beta'_{n(1 \leq n)}\}, co')$ and if $n > 1$, then $\beta'_{i(1 < i \leq n)} = \boxed{\beta''_i}$.

Let X_β be a set of concept nodes in $simple(\beta')$ such that $DDec(\beta') \subseteq X_\beta \subseteq Dec(\beta')$.

Let Θ_β be a substitution which associates an instantiated label to every concept node in X_β such that:

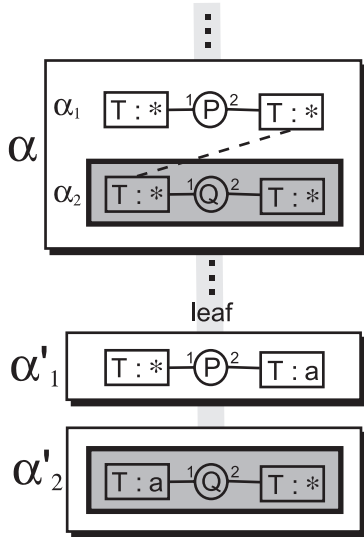
$\forall x \in X_\beta$ it holds that (i) $\Theta_\beta(x) = (t, m) \in \mathcal{C} \times \mathcal{I}$ and (ii) if there is a concept node labelled with (t', m) in \mathcal{T} , then $t = t'$, else $t \leq_{\mathcal{C}} type_\beta(c)$.

$\beta'''_1 = (\{G_\emptyset, \boxed{\beta'_1}\}, co_{\beta'_1})[\Theta_\beta]$ and
if $n > 1$, then $\beta'''_{i(1 < i \leq n)} = graph(\beta''_i)[\Theta_\beta]$.

We may simultaneously adjoin the graphs β'''_1 to β'''_n as successors of H .

4.2.11. EXAMPLE. To illustrate the decomposition of an α -graph, consider the

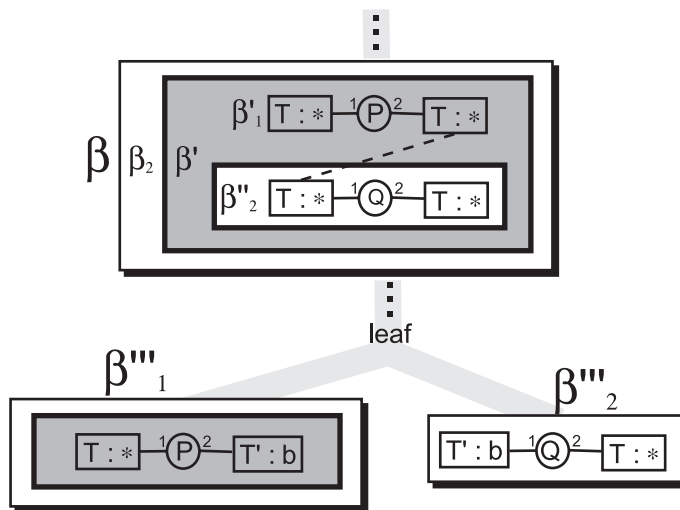
following tableau extension:



The simple entry of the graph contains two declarations, among which one is dominant. We may instantiate either both concept nodes or only the dominant one. We chose this last option and replace the marker $*$ by an object name which does not occur in the tableau yet: a . The second declaration is left unchanged: the existential node can later be ‘handled’ by projection.

A branch is interpreted as the conjunction of its nodes. The graph α is an assertion of the conjunction of the two entries α_1 and α_2 . Hence, after the substitution, we may assert both entries on the branch.

To illustrate a β -application, we can consider the negation of the previous α graph:



The graph β asserts a universally quantified disjunction. We may replace these universal quantifiers by any object name (new or already occurring in the tableau, but in the last case, we must ensure the coherence of the typing) and split the branch into the different disjuncts. As in the α -case, only dominant declarations need to be instantiated before the splitting.

A branch being a conjunction, it is not satisfiable if it contains the negation of a graph which logically follows from the concatenation of other graphs on the branch; it is then called closed. If all the branches of a tableau are closed then it is not possible to find a model for the represented disjunction of conjunctions. Furthermore, in this last case, as we will prove that the satisfiability of the input graph (i.e., the root of the tree) is preserved by the tableau construction, we can conclude that the input graph is not satisfiable. Hence, a validity proof of a conceptual graph is a closed tableau of the negation of the graph.

4.2.12. DEFINITION. [Proofs] Let B be a branch of a tableau \mathcal{T} and $positive(B)$ be the normal form of the concatenation of all simple graphs occurring as node of B ,

- B closes if it contains a node which is the negation of a simple graph H and there exists a projection from H to $positive(B)$.
- A proof of G is a tableau started with \boxed{G} , which has all of its branches closed.

4.2.2.2 Completeness of the calculus

The following proof follows the line of standard tableau completeness proofs. The main differences with usual textual systems reside in the following two features: (i) dominating boolean connectors and quantifiers are here handled in one expansion step and (ii) the search for a contradiction on a branch is performed by projection of simple subgraphs.

4.2.13. LEMMA. *If S is a satisfiable finite set of conceptual graphs, then:*

F1: *if an α occurs in S then $S \cup \{\alpha'_1, \dots, \alpha'_n\}$ is satisfiable.*

F2: *if a β occurs in S then at least one of the n sets $S \cup \{\beta''_1\}, \dots, S \cup \{\beta''_n\}$ is satisfiable.*

Proof:

- F1, case 1: suppose that Θ_α is empty. Hence, there is no dominant declaration to replace in α . Thus, $S \cup \{\alpha'_1, \dots, \alpha'_n\}$ is satisfiable.

case 2: Θ_α is not empty. $S \cup \{\alpha\}$ is satisfiable, say in a structure $M = (D, F)$. Hence, there is at least one partial assignment $f : X_\alpha \rightarrow D$ such that $M, f \models S \cup \{\alpha\}$ is satisfiable. Θ_α is a function from X_α to object names that do not occur in S , therefore, we can transform M into a model $M' = (D, F')$ such that $M' \models S \cup \{\alpha[\Theta_\alpha]\}$ by defining $F' = f$ on the constants in the codomain of Θ_α and $F' = F$ on the remaining constants. Now, there are no more dominant declarations in $\alpha[\Theta_\alpha]$, thus, by case 1, we may safely split the conjunction of entries: $S \cup \{\alpha'_1, \dots, \alpha'_n\}$ is satisfied in M' .

- F2, case 1: suppose that Θ_β is empty and S is satisfiable. β is the negation of a conjunction of graphs that are pairwise not connected by coreferences. From the interpretation of a negation box, it is clear that for at least one of these subgraphs, the conjunction of its negation with S is satisfiable.

case 2: Θ_β is not empty. $S \cup \{\beta'\}$ is satisfiable, thus it is not the case that there exists a substitution for the declarations of β' such that $S \cup \beta'$ is

satisfiable. Hence, for any such substitution, Θ_β , it holds that $S \cup \{\boxed{\beta'} [\Theta_\beta]\}$ is satisfiable. $\boxed{\beta'} [\Theta_\beta]$ is the negation of a conjunction of subgraphs which are not bridged by coreferences, thus, by case 1, it can safely be decomposed.

■

4.2.14. THEOREM (SOUNDNESS). *If there is a closed tableau started with \boxed{G} , then G is valid*

Proof: if the root \boxed{G} is satisfiable then at least one branch is satisfiable (by induction on the number of rules applied and the preservation of satisfiability). Projection is complete for simple graphs (cf. Theorem 3.2.6) and tableau branches (viz. sets of CGs) are interpreted as the conjunction of their nodes, thus a closed branch is unsatisfiable. Hence, one branch of the tableau must be open.

Thus, as the tableau is closed, the root \boxed{G} must be unsatisfiable and G is valid. ■

It remains to be proved that the tableau construction does not overlook any closing case; i.e, that by applying a systematic procedure that enumerates all possible expansions, if a branch remains open, then it is satisfiable.

4.2.15. DEFINITION. [Complete tableaux] A branch, B , of a tableau, \mathcal{T} , is exhausted if for every β -graph in B and every constructible substitution function, Θ_β , at least one resulting extension β_i''' occurs in B .

A tableau, \mathcal{T} , is called complete if for any branch B of \mathcal{T} , the following holds:

1. if an α occurs in B then for at least one choice of the substitution function Θ_α all corresponding α'_i occur in B , and
2. B is either closed or exhausted.

Fulfilling the exhaustion may require infinitely many steps. In a systematic procedure, an order on individual markers, e.g. the alphabetical one, must be chosen in order to enable the complete enumeration requested by the exhaustion of β applications. We assume that we have a systematic procedure for constructing a complete tableau, which guarantees that if the process of extending a branch does not terminate, then the resulting infinite branch is exhausted (for example an adaptation of Smullyan's systematic procedure in [Smu68]).

4.2.16. LEMMA. *In a complete tableau, \mathcal{T} , every exhausted open branch, B , is satisfiable.*

Proof:

1. We construct a canonical model $M = (D, F)$ from the information conveyed by χ -graphs in B .
 - (a) D is the set of all object names occurring in B .
 - (b) The interpretation function F is the identity on D and for every object name $x \in \mathcal{I} \setminus D$, $F(x) = \odot$.
 - (c) By construction of a tableau, an object name can only occur in association to a single concept type. Thus, we can respect the semi-lattice condition, by collecting the type of an object name and propagating this name upward to the interpretation all dominating concept types: for every $d \in D$ and every concept type $t' \in \mathcal{C}$, let $t \in \mathcal{C}$ be the type of any concept node labelled with d in B , it holds that $d \in F(t')$ iff $t \leq_{\mathcal{C}} t'$.
 - (d) For every relation name P of arity n and every sequence $\langle d_1, \dots, d_n \rangle \in D^n$, it holds that $\langle d_1, \dots, d_n \rangle \in F(P)$ if and only if there exists in B a (positive) simple conceptual graph in which occurs a relation node whose label is P' such that $P' \leq_{\mathcal{R}} P$ and whose (ordered) arguments are labelled with the object names $\langle d_1, \dots, d_n \rangle$.

It is immediate to verify that M satisfies the hierarchical constraints conveyed by the underlying language signature.

2. We must show that every conceptual graph, G , occurring as a node B , is satisfiable in the structure M . By induction on the degree of G .
 - (a) If $\text{degree}(G) = 0$, then G is a simple graph.
 - i. If $\text{Dec}(G) = \emptyset$, then it is obvious that G is satisfiable in M . Indeed, by construction of M , every atomic piece of information in G has been included in the structure and if there were two contradictory atomic pieces of information, then the branch would be closed by projection.
 - ii. Else, $\text{Dec}(G) \neq \emptyset$ and G is an α . Since the tableau is complete, there exists in B a copy of G with all declarations replaced by new object names: say, α'_1 obtained from G by applying a substitution Θ_α (note that tableau rules enable to obtain this instantiated graph in a finite number of successive steps, but for a complete tableau, it is requested to be obtained in a single simultaneous substitution of all declarations). α'_1 corresponds to the previous case and thus, it is satisfiable in M . Now we can use the function $\Theta_\alpha \cup F$ as an assignment which satisfies G in M .

(b) $\text{degree}(G) > 0$

i. If G is an α (with $n > 1$ as $\text{degree}(G) > 0$), then α'_1 to α'_n (obtained with the substitution function Θ_α) occur in B . Since $\text{degree}(\alpha'_{i(1 \leq i \leq n)}) < \text{degree}(G)$, then by induction hypothesis, each of these $\alpha'_{i(1 \leq i \leq n)}$ is true under an assignment f_i in M .

Furthermore, the coreference class of a dominant declaration in G has been instantiated with a single new object name. Hence, each of the $\alpha'_{i(1 \leq i \leq n)}$ is satisfied by M under $f = \bigcup_{1 \leq i \leq n} f_i$. Therefore, G is satisfied by $f \circ \Theta_\alpha$ in M .

ii. G is a β of the form $\boxed{\beta'_1, \beta''_2, \dots, \beta''_n}$.

B is exhausted, therefore it holds that for every constructible substitution function Θ_β , at least one of the graphs in $\{\boxed{\beta'_1[\Theta_\beta]}\} \cup \bigcup_{1 < i \leq n} \beta''_i[\Theta_\beta]$ occurs in B and is satisfiable in M (by induction hypothesis and since the degree of all these graphs is less than the degree of G). Hence, for every Θ_β , $\boxed{\beta'_1[\Theta_\beta], \beta''_2[\Theta_\beta], \dots, \beta''_n[\Theta_\beta]}$ is satisfiable in M . Thus, it is not the case that there exists a substitution $f : \text{Dec}(G) \rightarrow D$ such that $\forall c \in \text{Dec}(G), f(c) \in F(\text{type}(c))$ and $\{\beta'_1, \beta''_2, \dots, \beta''_n\}[f]$ is satisfiable. Therefore G is satisfiable in M .

iii. G is the negation of a simple conceptual graph: $\boxed{\beta'_1}$.

Suppose that G is not satisfiable in M , then β'_1 is satisfiable in M , viz. there is an embedding of β'_1 into M . Furthermore, by construction, M is the canonical model of the concatenation of all simple conceptual graphs in B (i.e., $\text{positive}(B)$). Hence, by completeness of projection (cf. Theorem 3.2.6), there exists a projection from β'_1 to $\text{positive}(B)$ and B is closed. But, B is an exhausted open branch, thus G is satisfiable in M .

■

4.2.17. THEOREM (COMPLETENESS). *If a conceptual graph G is valid, then there exists a proof of G .*

Proof: A proof of G is a closed tableau started from the negation of G : \boxed{G} .

If there is no proof of G , then there is an exhausted open branch B in any complete tableau started from \boxed{G} . By Lemma 4.2.16, B is satisfiable and therefore the root, \boxed{G} , which is an element of the branch, is satisfiable. Hence, G is not valid. ■

4.2.3 Related work and further directions

In this brief detour outside our track of efficiently computable graph methods, we have described a general and modular procedure for combining tableau and projection algorithms. Though borrowed from the traditional textual courant of symbolic logic, tableau methods perfectly fit the graphical setting of conceptual graphs: their representation system are trees.

The separation of work between both implied types of calculi, tableaux and homomorphisms, allows their almost independent tuning when practical efficiency is desired. For instance, a well-known technique of tableau implementations consists in delaying the instantiations and delegating the choice to a unification procedure. A free-variable variant of tableaux for conceptual graphs has been proposed in [Ker97]. In this case, projection can be seen as a meta-unification procedure on pieces of information which are not necessarily atomic, but quantified conjunctions of atoms.

Peirce’s existential graphs. Other complete calculi for conceptual graph languages equivalent to FOL have been proposed since the early days of conceptual graphs (e.g., [Sow84], [Wer95] or [Ham98]), but their faithfulness to Peirce’s original system of existential graphs have distanced them from possible implementation. Indeed, although some argue for the intuitive characteristics of Peirce’s α and β systems (see e.g., [Rob73, Rob92] and [Thi75] for detailed presentation of Peirce original work on FOL calculi), these proof methods are not analytical and rely extremely on human intuition to guess the next proof step. For instance, the following rule is an essential part of the propositional α calculus ([Rob73] p41) “*The rule of insertion.* Any graph may be scribed on any oddly enclosed area”. Of course, it is classically valid to derive $\neg(A \wedge B)$ from $\neg(A)$, but when it comes to use the rule in an automated way, the infinite choice of graphs offered to take the place of B is problematic.

Far from the ideological debate on the ease of learning or using such-and-such type of calculi, we have decided to adopt Peirce’s notion of closed areas to represent negations, while choosing an analytical proof method that a machine can apply without much intuition. After all, the part reserved to cognitive efficiency in this thesis is sufficiently occupied by graphical items.

DRT-Tableaux. In [BE82], it was already advanced that the interaction of tableau reasoning and picture embedding can serve linguistics purposes such as the resolution of some kinds of anaphoric bindings. The existential nature and the sole use of implication and conjunction in discourse representation structures [Kam81] lend themselves to the analytical steps of tableau constructions. [KR96], an early tableau calculus for DRT, has been an important source of inspiration for preliminary versions of this chapter: interlacing projections and tableaux

have been proposed in [Ker96, Ker97] for a language of conceptual graph based on implication.

Graph rules. Having implication as sole logical connective between simple conceptual graphs has also been applied in a study of a resolution-like calculus [SM96]. In [Sal97], this method combining resolution steps and adapted projections has been experimentally compared to classical resolution techniques, in order to highlight the advantages of projection for an early detection of dead-ends in the explored proof tree. Comparing the resolution style and the tableau one in the conceptual graph framework has also been our theme in [KS97].

Intermediate decidable fragments. Among the infinity of fragments standing between simple graphs and full conceptual graphs, most interesting are the decidable ones for which only a careful application of the (adapted) tableau rules can guarantee to satisfy the sharp bounds laid by theoretical complexity analysis.

One such appealing logic is the guarded fragment of first-order logic (2-EXP-complete satisfiability problem [Grä99]) which expands the tree property to new horizons. A guarded syntax of conceptual graphs has been proposed in [BKM99] and the calculus side would also deserve some attention, witness the subtleties employed in [Niv98] for adapting resolution to guarded FOL. So, the next task for the proponents of a systematic exploration of the conceptual graph landscape lies straight ahead.

For the time being, we propose an alternative route that offers the possibility of closing a loop opened in the previous chapter: reintroducing the tree-like structures of guarded simple graphs into their modal ancestors.

4.3 Nested graphs

A keyword in knowledge representation is *modularity*. Imagine how tragic it would be, if to access a web page, the whole content of all internet sites would first need to be downloaded. The world wide web is a typical modular knowledge base in which, among other properties, pieces of information grouped in pages are made available by different authors, pages or even predefined locations in pages are named and can be referred by other pages and in some cases, chosen pieces of information can only be reached by specific sequences of actions. This modular setting is common to almost all kinds of complex representations; e.g., the sectioning units in a book, the division of a cooking receipt into small tasks which can sometimes be accomplished in parallel, a large scale schema with enlargement of specific items, object-oriented programming languages, etc.

When the represented knowledge is partitioned into local subsets, the way *navigation* from one group to an other is offered, takes a preponderant rôle; after

all, representing information makes sense if it possible to retrieve any piece even in the deepest nested subunit.

Modularity does not only prove salient to representation: we have observed, in Chapter 2, how a switch of interpretation viewpoint, from the global one of first-order logic fragments to the local one of modal logics, can beneficially influence the complexity of reasoning.

In this section, we will combine these representational and computational themes into a framework offering the possibility to localise information conveyed by simple conceptual graphs.

4.3.1 Modularity by nesting

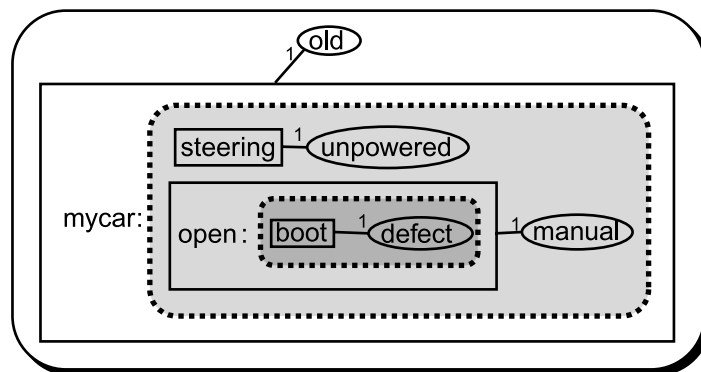
In a single picture, a way to distinguish knowledge levels is to nest pieces of information inside other ones. Here, the term *level* should not necessary convey a notion of preferability, importance, but rather a geography of information with traced paths between the different pieces. The implicit navigational support associated to nesting is some kind of *zooming in* procedure which enables to let the focus jump from the level at stake to one of the pieces nested in it.

Nested simple conceptual graphs emerged as an extension of the simple conceptual graph model in [Sow92] but have only been thoroughly studied in recent work [CM97, Sim98, CMS98] on the basis of a better understanding of the underlying simple structures.

4.3.1.1 The traditional conceptual graph approach to nesting

Information is still represented by simple graphs with the difference that an entire graph can be associated to the concept node it describes, can be *nested* in it.

4.3.1.1. EXAMPLE. If I should describe my car, I would not start by exposing the failure of a small non-vital system in one of its components (i.e., the opening of the boot), but rather with the general features of the vehicle (e.g., its age, the engine main characteristics, the type of steering system, etc.):



Building on our understanding of simple graphs, it is most natural to interpret simple graph components of a nested graph in classical first order structures and consider that the notion of nesting corresponds to some relation between the different structures described.

This sketch of semantics already raises many questions about the relation between the simple structures involved in a model for a nested graph. Not only, how they are connected to each other, but also whether they have the same universe, whether nesting is associated to some notion of inheritance, etc. The range of alternatives implied by such questions is familiar to modal predicate logics and the semantics for nested conceptual graphs proposed in the forthcoming sections is inevitably not the unique one, but it will conciliate the chosen syntactic features of nested graphs to the intuitive meaning we will associate to them.

Individuals and domains A nested graph is intended to represent some structured information about a set of objects that may be named with individual markers. Furthermore, the scope of quantification will be allowed to go beyond the boundaries of nested components through coreference links. Hence, it makes sense to associate the same denotation to any individual marker in every substructure where it exists. Concerning this last point, we note that some objects may not be relevant to all substructures: in Example 4.3.1, the boot is not an object that necessarily belongs to the first level of description of my car, whereas it is relevant to the context of the openings. Hence, every substructure will have as domain a subset of the global universe.

Navigation There is a prime syntactical distinction between the notion of nesting in conceptual graphs and the usual notion of modal operator in modal logics: a subgraph is associated to a particular term in a graph, while a modal subformula is usually directly nested in another formula. This notion of term-nesting can be the representation of a relation connecting objects in a substructure to other substructures.

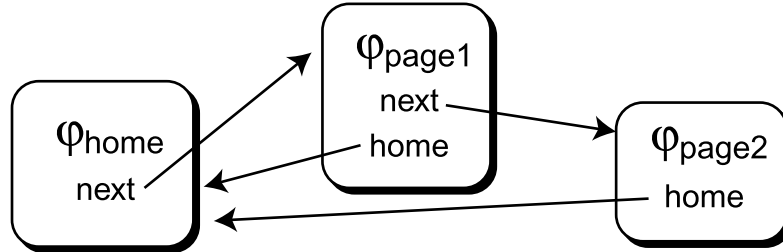
As any individual marker has a fixed denotation in every substructure, should it also have the same connections in every substructure or, on the contrary, should occurrences of an individual marker in distinct subgraphs correspond to objects that have a priori nothing in common except sharing a name? We argue that such a choice should be expressible in the language and hence, that we need to extend the usual syntax of nested conceptual graphs. Let us consider the description of a web-site outline in which two different uses of nesting are interlaced:

4.3.2. EXAMPLE. We have three web-pages: *home*, *page1* and *page2*. The author has designed two buttons called *next* and *home* that are used in the different pages:

- On *home*, φ_{home} holds (e.g., φ_{home} can be a representation of the content of the page) and the button *next* is a link to *page1*.

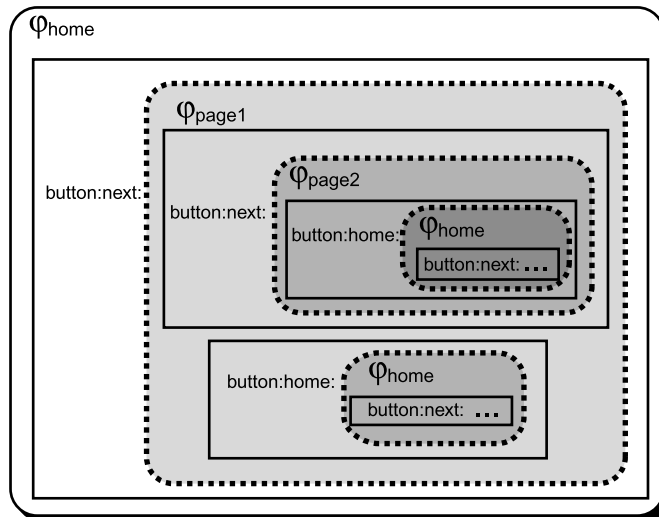
- On *page1*, φ_{page1} holds, *next* leads to *page2* and *home* to the home-page.
- Finally, on *page2*, φ_{page2} holds and the button *home* points to the home-page.

The structure we want to represent by a nested graph has the following outline:



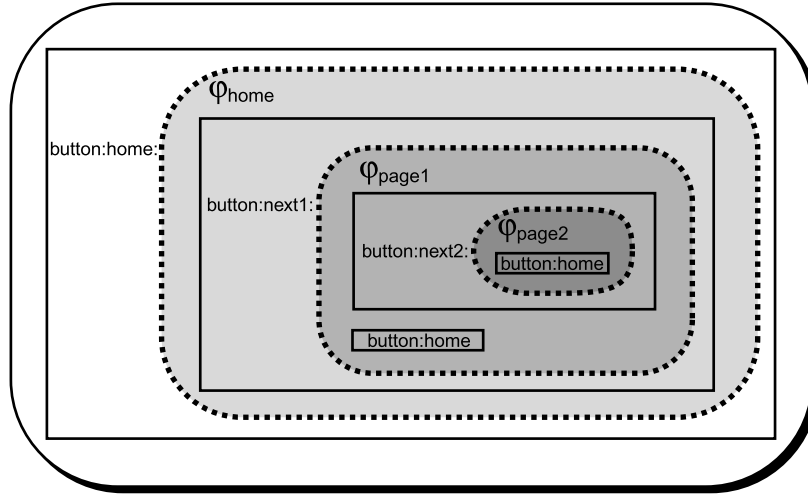
Chein et al [CMS98] proposes two alternative semantics for nested conceptual graphs:

1. In the first one, two occurrences of a term share a priori nothing excepted their name. Therefore, to ensure the capture of the fact that the page pointed by *home* verifies φ_{home} , it is required to nest a copy of the home-page into the concept node `button:home`. Unfortunately, an adequate conceptual graph representation of the site becomes an **infinite** chain of nesting with no available syntactical shortcut for the repeated pattern:



2. In the second semantics proposed by Chein et al., all occurrences of a term have the same description (i.e., an object is connected to the same substructures in all substructures where it exists). In this case, the representation of the home-link is no more problematic, but now, all occurrences of the

next-button need to be named differently:



The unsatisfactory point in this setting is that it does not represent the fact that both occurrences of the button *next* correspond to a single object occurring in different pages.

We can attack the problem directly at its source: the missing syntactical element is a way to name a nested subgraph such that latter references to it are enabled. Hybrid logics (e.g., [BT99, ABM99b, Are00]) are tackling a similar problem in modal logics; they introduce nominals, which are terms in a first-order logic manner, that denote worlds in propositional modal languages.

Similarly, we will name every nested subgraph with either a “graph constant” or an indefinite marker representing existential quantification.

4.3.1.2 Exploiting the power of guards in the nested setup

The guarded fragment of simple conceptual graphs (and its sub-fragment of trees) had already a modular flavour: localised pieces of information connected by constrained paths of coreference links. This is not surprising as guarded fragments arose from the study of modal languages. It is interesting to “close a loop” by reimporting the notion of guards into a modal language. Indeed, by including FOL, modal predicate logics inherit undecidable decision problems. However, we will prove that forcing guarded patterns of nesting is a way to highlight a tractable fragment of nested conceptual graphs.

4.3.2 Nested conceptual graphs

A common ground to the different “flat” conceptual graph fragments is a close resemblance of, on one hand, the graph representations and on the other hand, the represented formal structures; actually this homomorphic setting has been

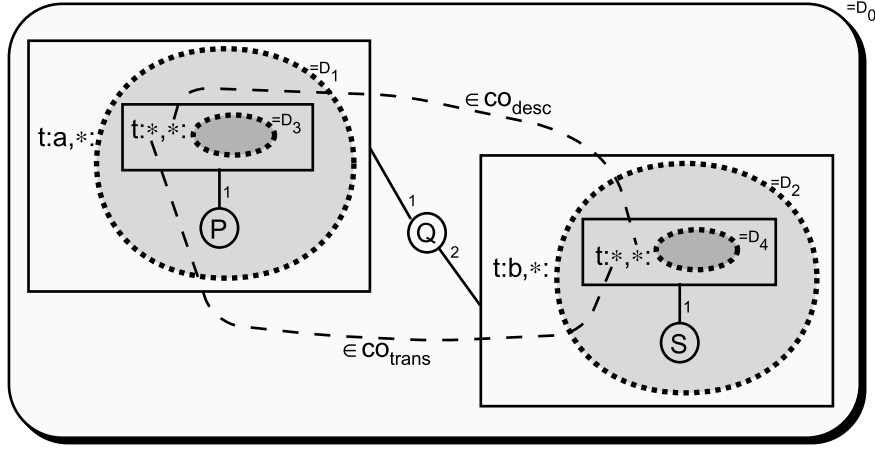


Figure 4.4: A nested conceptual graph

equally highlighted as the justification of the projection calculus completeness and as a cognitive strength of the graphs.

4.3.2.1 Syntax

Nesting has been intuitively presented as a way of clustering pieces of first-order information into modules of a network. We have argued for the necessity of naming these modules: nominals are introduced in the vocabulary.

4.3.3. DEFINITION. [Extension of a simple graph signature] A nested signature Σ is a pair (σ, \mathcal{N}) where $\sigma = (\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}}), (\mathcal{R}, \leq_{\mathcal{R}}), \text{arity})$ is a signature for simple conceptual graphs² and \mathcal{N} is a non-empty set of nominals with a distinguished element N_0 and $\mathcal{I} \cap \mathcal{N} = \emptyset$.

To simplify the reading, we will use lower case letters for object names in \mathcal{I} and capital ones for nominals in \mathcal{N} .

In the introductory examples, not all concept nodes were holding a nested subgraph. However, for the sake of simplicity, it is convenient to define a nested conceptual graph as a simple conceptual graph in which a nested conceptual (sub)graph is associated to *every* concept node. The end-point of the recursion is obtained by taking the empty simple conceptual graph (i.e, the logical constant *True*) as a nested one.

We adopt a syntactic definition which will later facilitate the translation to the simple graph fragment: instead of using a purely recursive definition, we emphasise the set of simple graph components involved in a nested graph.

4.3.4. DEFINITION. [Syntax of a nested conceptual graph] Given a nested signature $\Sigma = (\sigma, \mathcal{N})$, a nested conceptual graph (NCG) over Σ is a finite graph $G = (D = \{D_0, \dots, D_n\}, \text{desc}, \text{nom}, \text{co}_{\text{desc}}, \text{co}_{\text{trans}})$ where $0 \leq n$ and

²Definition 3.1.1

1. D , the set of descriptions, is a non-empty set of normalised simple conceptual graphs over σ . For $0 \leq i \leq n$, $D_i = (R_{D_i}, C_{D_i}, E_{D_i}, label_{D_i}, co_{D_i})$. The set of concept nodes occurring in G is noted $concept_G$; i.e., $concept_G = \bigcup_{D_i \in D} C_{D_i}$.
2. $desc$ is a bijection from $concept_G$ to $D \setminus \{D_0\}$; there is a one-to-one correspondence between concept nodes and descriptions (with the exception of the outermost description D_0).
3. nom is a labelling function from D to $\mathcal{N} \cup \{*\}$ such that $label(D_0) = N_0$.
4. co_{desc} , the coreference relation on descriptions, is an equivalence relation on the set of descriptions labelled with the marker “*”; i.e., $\{D_i/D_j \in D \text{ and } nom(D_i) = *\}$.
5. co_{trans} , the trans-description coreference relation, is an equivalence relation on existential concept nodes in D such that two distinct nodes are coreferent only if they occur in distinct descriptions.

In other words, $desc$ associates a unique description to every concept node and each description is named with either a nominal or the place holder “*”.

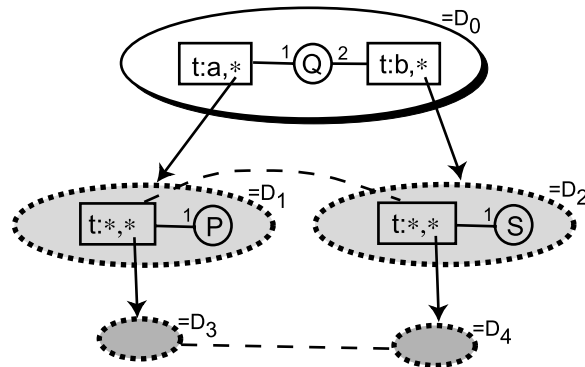


Figure 4.5: The underlying tree structure of the NCG in Figure 4.4

The underlying recursive structure of nesting can easily be extracted from D and $desc$: it is a tree with root D_0 such that a description D_i is a direct successor of a description D_j with $0 \leq i, j \leq n$ and $i \neq j$ iff there exists in D_j a concept node $c \in C_{D_j}$ such that $desc(c) = D_i$. For instance, in Figure 4.5 is represented the underlying tree of the nested graph in Figure 4.4.

It should however be noted that co_{trans} and co_{desc} are external to this recursive structure. Indeed, two idems occurring in independent branches can be coreferent. This notion of cross-world quantification generalises the usual one in predicate modal logic as, here, the scope of existential quantifiers is not bound by the one of surrounding modalities. For instance, the nested graph in Figure 4.4 represents a

situation in which (i) two objects a and b are in relation Q , (ii) in the description of a there is an object which has the property P and (iii) the same object has the property S in the description of b , while this object does not necessarily exist in the world depicted by the root D_0 .

4.3.2.2 Nested structures

The intuitive meaning of a graph can be formalised by an embedding into a nested structure. To each description D_i corresponds a first-order structure, a world w_i , verifying the content of the description (the simple conceptual graph at stake) and for each concept node c of this description D_i , if w_j is the world corresponding to the subgraph $desc(c)$, then it must hold that $(w_i, \llbracket c \rrbracket, w_j)$ is in a ternary accessibility relation R_{acc} connecting objects in a world (the first two arguments) to other worlds.

4.3.5. DEFINITION. [Nested Σ -structures] A nested structure over $\Sigma = (\sigma, \mathcal{N})$ is a tuple $(W, O, \llbracket \cdot \rrbracket, S, R_{acc})$ such that:

1. W is a set of worlds
2. O is a set of (discourse) objects
3. $\llbracket \cdot \rrbracket : \begin{cases} \mathcal{I} \rightarrow O \cup \{\odot\} \\ \mathcal{N} \rightarrow W \cup \{\odot\} \end{cases}$
4. S is a function from W to the set of σ -structures such that:

$$\forall w \in W, \begin{cases} S(w) = (dom^w, \llbracket \cdot \rrbracket^w) \\ dom^w \subseteq O \\ \forall x \in \mathcal{I}, \text{ if } \llbracket x \rrbracket \in dom^w, \text{ then } \llbracket x \rrbracket^w = \llbracket x \rrbracket \text{ else } \llbracket x \rrbracket^w = \odot \end{cases}$$
5. and R_{acc} is a ternary relation in $(W \times O \times W)$ such that $(w, o, w') \in R_{acc}$ only if $o \in dom^w$.

Nominals, like individual markers, have a fixed denotation. The structure assigned to a world by the function S is a usual non-nested first-order structures respecting the chosen ordering on concept and relation names as proposed in Definition 3.1.6. Furthermore, the definition forces an individual marker to have the same denotation in all worlds where it exists.

4.3.6. FACT. An object is in the domain of a world if and only if it is in the denotation of the supremum in the concept type hierarchy at the world:

$$\forall w \in W, \forall o \in O, o \in dom^w \text{ iff } o \in \llbracket c^\top \rrbracket^w$$

Proof: Definition 3.1.6 assigns the whole domain of a flat structure to the denotation of the top-concept. ■

By allowing trans-description coreference links, we have abandoned the recursive syntax of nested graphs and therefore, we define assignments globally on the whole representation:

4.3.7. DEFINITION. [Truth definition of NCGs]

Given a nested signature Σ , a nested Σ -structure $M = (W, O, \llbracket \cdot \rrbracket, S, R_{acc})$ and a NCG $G = (D, desc, nom, co_{desc}, co_{trans})$ over Σ ,

- an assignment is a function $\llbracket \cdot \rrbracket_a$ extending $\llbracket \cdot \rrbracket$ from names to graph items (concept nodes and descriptions) and respecting coreferences;

$$\llbracket \cdot \rrbracket_a : \begin{cases} D \rightarrow W \\ concept_G \rightarrow O \end{cases}$$

such that

$$\begin{cases} \forall x \in D, nom(x) \in \mathcal{N} \Rightarrow \llbracket x \rrbracket_a = \llbracket nom(x) \rrbracket \\ \forall x, y \in D, (x, y) \in co_{desc} \Rightarrow \llbracket x \rrbracket_a = \llbracket y \rrbracket_a \\ \forall c \in concept_G, marker(c) \in \mathcal{I} \Rightarrow \llbracket c \rrbracket_a = \llbracket marker(c) \rrbracket \\ \forall c, c' \in concept_G, (c, c') \in co_{trans} \Rightarrow \llbracket c \rrbracket_a = \llbracket c' \rrbracket_a \end{cases}$$

- $M \models_{NCG} G$ iff there exists an assignment $\llbracket \cdot \rrbracket_a$ s.t. $(M, \llbracket N_0 \rrbracket, \llbracket \cdot \rrbracket_a) \models_{NCG} D_0$

$$\bullet (M, w, \llbracket \cdot \rrbracket_a) \models_{NCG} D_i \text{ iff } \begin{cases} \forall c \in C_{D_i}, \llbracket c \rrbracket_a \in dom^w \\ \forall c \in C_{D_i}, \langle w, \llbracket c \rrbracket_a, \llbracket desc(c) \rrbracket_a \rangle \in R_{acc} \\ \forall c \in C_{D_i}, (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCG} desc(c) \\ (S(w), \llbracket \cdot \rrbracket_a) \models_{SCG} D_i \end{cases}$$

- $G \sqsubseteq_{NCG} H$ iff for every Σ -structure M , if $M \models_{NCG} G$, then $M \models_{NCG} H$

We now have a formal grasp on the meaning of nested representations, but how complex is reasoning in the nested framework?

4.3.3 Complexity and Guards

We have defined representations and structures that resemble each other and it should be quite natural to expand the strategy applied to simple graphs: the definition of a projection calculus which simulates the embedding of a nested graph into the canonical model of another one. Chein et al. [CM97, CMS98] define a recursive projection-procedure based on the calculus for simple graphs and they handle the alternative semantics for their nested language by defining alternative

canonical models (via different forms of normalised nested graphs). However a such line of thinking does not provide an immediate clue on the complexity of the recursive method (only a lower NP bound due to the matching of two simple graphs).

We here choose a different strategy: we skip the definition of a calculus and translate the nested setup into a fragment of simple conceptual graphs. The completeness of such a translation would have two corollaries: a measure of the difficulty of reasoning in the nested fragment and a direct exploitation of the tractability of guarded simple graphs (cf. Chapter 3.3).

4.3.3.1 From nested graphs to simple ones

A first step concerns a way of encoding the notions of world-partition and local-substructure into a usual first-order structure.

By associating a local-substructure to every world, we have let the properties of each individual (i.e., its concept type and the relations that link it to other individuals of the world at stake) be relative to the local notion of world. This information can be captured by an extra argument to every predicate. Furthermore, from this encoding, we can also derive the domain of a given world: an individual o belongs to the domain of a world w if and only if w and o occur together in the interpretation of the super-concept c^\top . The transitions from objects in worlds to other worlds were captured by the accessibility relation R_{acc} ; It will prove convenient for guarded representations to include this information in the denotation of other predicates.

4.3.8. DEFINITION. [Derived signature] Given a nested signature $\Sigma = ((\mathcal{I}, (\mathcal{C}, \leq_{\mathcal{C}})), (\mathcal{R}, \leq_{\mathcal{R}}), arity), \mathcal{N}$), its derived signature is a signature for simple conceptual graphs $\sigma = (\mathcal{I} \cup \mathcal{N}, \{c^\sigma\}, (\mathcal{R}^\sigma, \leq_{\mathcal{R}^\sigma}), arity^\sigma)$ such that:

1. c^σ is the unique concept type
2. $\forall r \in \mathcal{R}^\sigma, \exists k \in \mathbf{IN}^+$ such that $arity^\sigma(r) = 2k + 1$
3. $\mathcal{R}_3^\sigma = \mathcal{C} \cup \mathcal{R}_1$ and $\leq_{\mathcal{R}_3^\sigma}$ preserves both orders $\leq_{\mathcal{C}}$ and $\leq_{\mathcal{R}_1}$.
4. $\forall k > 1, (\mathcal{R}_{2k+1}^\sigma, \leq_{\mathcal{R}_{2k+1}^\sigma}) = (\mathcal{R}_k, \leq_{\mathcal{R}_k})$

How the arity-change is employed becomes clear in the following transformations:

4.3.9. DEFINITION. [λ - μ translations between nested structures and flat ones] Given a nested signature Σ with derived signature σ , let c^\top be the supremum in the concept type hierarchy of Σ ,

- For a nested Σ -structure $M = (W, O, [\cdot]_M, S, R_{acc})$, $\lambda(M)$ is a flat σ -structure $(X, [\cdot]_{\lambda(M)})$ such that

1. relevant worlds:

$$W' = \llbracket \mathcal{N} \rrbracket_M \cup \{w \in W / \exists x \in W, \exists o \in O, \langle x, o, w \rangle \in R_{acc}\}$$

relevant objects:

$$O' = \llbracket \mathcal{I} \rrbracket_M \cup \{o \in O / \exists x, y \in W', \langle x, o, y \rangle \in R_{acc}\}$$

$$X = W' \cup O'$$

2. $\forall x \in \mathcal{I} \cup \mathcal{N}$,

$$\llbracket x \rrbracket_{\lambda(M)} = \llbracket x \rrbracket_M$$

3. $\forall P \in \mathcal{R} \cup \mathcal{C}$ such that³ $\text{arity}(P) = k$,

$$\llbracket P \rrbracket_{\lambda(M)} = \left\{ \langle w, o_1, v_1, \dots, o_k, v_k \rangle / \left[\begin{array}{l} w \in W' \\ \& \langle o_1, \dots, o_k \rangle \in \llbracket P \rrbracket_M^w \\ \& \forall 1 \leq i \leq k, \langle w, o_i, v_i \rangle \in R_{acc} \end{array} \right] \right\}$$

- For a flat σ -structure $M' = (X, \llbracket \cdot \rrbracket_{M'})$,
 $\mu(M')$ is a nested Σ -structure $(W, O, \llbracket \cdot \rrbracket_{\mu(M')}, S, R_{acc})$ such that

1. $R_{acc} = \left\{ \langle w, o, v \rangle / \left[\begin{array}{l} \langle w, o, v \rangle \in \llbracket c^\top \rrbracket_{M'} \\ \text{and } v \text{ is uniformly accessible from } o \text{ in } w \end{array} \right] \right\}$

where v is uniformly accessible from o in w

$$\text{iff } \left[\begin{array}{l} \forall P \in \mathcal{R} \cup \mathcal{C} \text{ such that } \text{arity}(P) = k \\ \text{and } \forall \langle w, o_1, v_1, \dots, o_k, v_k \rangle \in \llbracket P \rrbracket_{M'} \\ \text{if } o_i = o \text{ with } 1 \leq i \leq k, \text{ then } \langle w, o_1, v_1, \dots, o_k, v_k \rangle [v_i/v] \in \llbracket P \rrbracket_{M'} \end{array} \right]$$

2. $W = \llbracket \mathcal{N} \rrbracket_{M'} \cup \{w \in X / \exists x, y \in X \text{ and } \langle x, y, w \rangle \in R_{acc}\}$

3. $\forall w \in W$,

$$(a) \text{ dom}^w = \{o \in X / \exists x \in W \text{ and } \langle w, o, x \rangle \in R_{acc}\}$$

$$(b) \forall x \in \mathcal{I}, \llbracket x \rrbracket_{\mu(M')}^w = \left[\begin{array}{l} \llbracket x \rrbracket_{M'} \text{ if } \llbracket x \rrbracket_{M'} \in \text{dom}^w \\ \odot \text{ otherwise} \end{array} \right]$$

$$(c) \forall P \in \mathcal{R} \cup \mathcal{C} \text{ such that } \text{arity}(P) = k,$$

$$\llbracket P \rrbracket_{\mu(M')}^w = \left\{ \langle o_1, \dots, o_k \rangle / \left[\begin{array}{l} \langle w, o_1, v_1, \dots, o_k, v_k \rangle \in \llbracket P \rrbracket_{M'} \\ \text{and } \forall 1 \leq i \leq k, \langle w, o_i, v_i \rangle \in R_{acc} \end{array} \right] \right\}$$

$$(d) S(w) = (\text{dom}^w, \llbracket \cdot \rrbracket^w)$$

4. $O = \llbracket \mathcal{I} \rrbracket_{M'} \cup \bigcup_{w \in W} \text{dom}^w$

5. $\forall x \in \mathcal{I} \cup \mathcal{N}, \llbracket x \rrbracket_{\mu(M')} = \llbracket x \rrbracket_{M'}$

We note that both structure translations do not preserve those pieces of information which cannot play a rôle in the satisfaction of a nested graph. For

³By convention, the arity of a concept type in \mathcal{C} is 1.

instance, a world which is neither in the denotation of nominals nor in the accessibility relation is eliminated by λ . Also, world connections are conveyed by the accessibility relation in a nested structure, while the information is duplicated in the arguments of relations in a flat structure and μ eliminates some incomplete patterns; e.g., suppose that a flat structure M' is partially described by (i) “ $\{\langle w, a, y \rangle\} = \llbracket P \rrbracket_{M'}$ and $\{\langle w, a, x \rangle\} = \llbracket Q \rrbracket_{M'}$ ”. Neither x nor y is uniformly accessible from a in w , hence, (i) is not preserved in $\mu(M')$.

4.3.10. DEFINITION. [Interpretation of NCGs into flat structures] Given a nested signature Σ with derived signature σ , a σ -structure $M = (O, \llbracket \cdot \rrbracket)$ and a NCG $G = (D, desc, nom, co_{desc}, co_{trans})$ over Σ ,

- an assignment is a function from nodes to objects which coincides with the denotation of constants

$\llbracket \cdot \rrbracket_a : D \cup concept_G \rightarrow O$ such that

$$\forall x, y \in D \text{ and } \forall c, c' \in concept_G, \begin{cases} nom(x) \in \mathcal{N} \Rightarrow \llbracket x \rrbracket_a = \llbracket nom(x) \rrbracket \\ (x, y) \in co_{desc} \Rightarrow \llbracket x \rrbracket_a = \llbracket y \rrbracket_a \\ marker(c) \in \mathcal{I} \Rightarrow \llbracket c \rrbracket_a = \llbracket marker(c) \rrbracket \\ (c, c') \in co_{trans} \Rightarrow \llbracket c \rrbracket_a = \llbracket c' \rrbracket_a \end{cases}$$

- $M \models_{NCGflat} G$ iff $\exists \llbracket \cdot \rrbracket_a$ assignment s.t. $(M, \llbracket N_0 \rrbracket, \llbracket \cdot \rrbracket_a) \models_{NCGflat} D_0$
- $(M, w, \llbracket \cdot \rrbracket_a) \models_{NCGflat} D_i$ iff

$$\begin{array}{l} \forall c \in C_{D_i}, \\ \forall r \in R_{D_i}, \end{array} \begin{cases} \langle w, \llbracket c \rrbracket_a, \llbracket desc(c) \rrbracket_a \rangle \in \llbracket type(c) \rrbracket \\ \langle w, \llbracket c_1 \rrbracket_a, \llbracket desc(c_1) \rrbracket_a, \dots, \llbracket c_k \rrbracket_a, \llbracket desc(c_k) \rrbracket_a \rangle \in \llbracket label_{D_i}(r) \rrbracket \\ (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} desc(c) \end{cases}$$

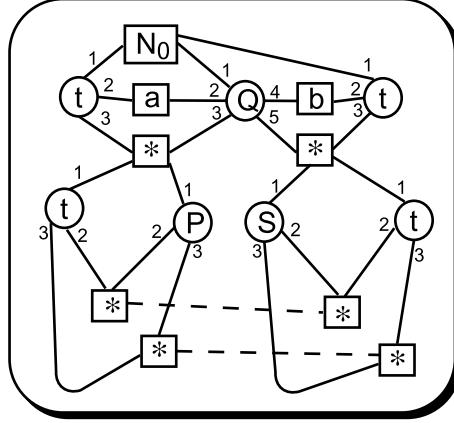
where k is the arity of the relation symbol labelling r and $c_{j, 1 \leq j \leq k}$ is the j^{th} concept node neighbour of r in D_i .

- $G \sqsubseteq_{NCGflat} H$ iff for every σ -structure M , $M \models_{NCGflat} G$ implies that $M \models_{NCGflat} H$

In Theorem 4.3.12, we will prove that the model transformation λ and μ preserve the satisfaction of nested graphs. For the time being, we will go directly to the point of this section: the translation of a nested graph into a simple one.

4.3.11. DEFINITION. [κ translation] Given a nested signature Σ with derived signature σ and a NCG G over Σ , $\kappa(G) = (R, C, E, label, co)$ is a simple conceptual graph over σ defined from $G = (D, desc, nom, co_{desc}, co_{trans})$ as follows:

1. $\forall D_i = (R_{D_i}, C_{D_i}, E_{D_i}, label_{D_i}, co_{D_i}) \in D$,

Figure 4.6: The κ -translation of the NCG in Figure 4.4

- (a) $\kappa_{nom}(D_i)$ is a concept node $\boxed{c^\sigma : nom(D_i)}$
 i.e., to each description corresponds a concept node referring to the name of the described world

(b) $\forall c \in C_{D_i},$ $\left[\begin{array}{l} \kappa_{marker}(c) \text{ is a concept node } \boxed{c^\sigma : marker(c)} \\ \kappa_{type}(c) \text{ is a relation node } \textcircled{type(c)} \\ \kappa_{edge}(c) = \left\{ \begin{array}{l} (\kappa_{type}(c), \kappa_{nom}(D_i), 1), \\ (\kappa_{type}(c), \kappa_{marker}(c), 2), \\ (\kappa_{type}(c), \kappa_{nom}(desc(c)), 3) \end{array} \right\} \end{array} \right.$

i.e., a concept node is transformed into a relation node with as arguments the description in which it occurs, its marker and the description it is pointing to.

- (c) $\forall r \in R_{D_i}, \kappa(r) = (r, \kappa_{nom}(D_i), 1)$ and $\forall e = (r, c, k) \in E_{D_i},$
 $\kappa(e) = \{(r, \kappa_{marker}(c), 2k), (r, \kappa_{nom}(desc(c)), 2k + 1)\}$
 i.e., arities of relation are changed according to the transition from a nested signature to its derived one.

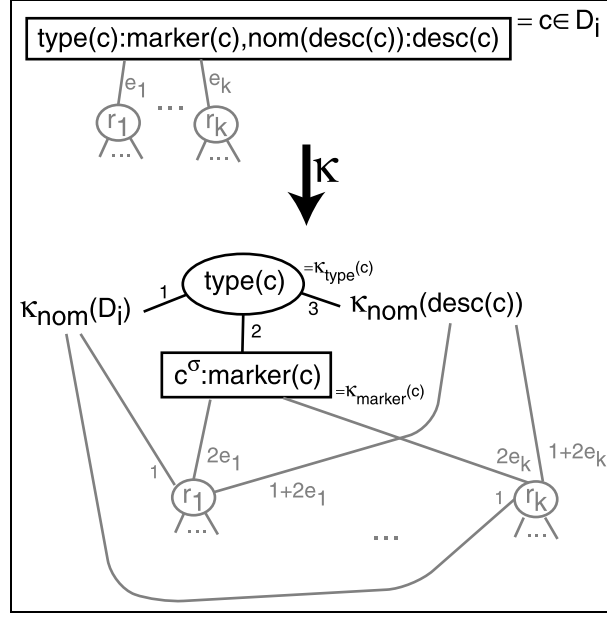
2.

$$R = \bigcup_{D_i \in D} R_{D_i} \cup \bigcup_{c \in concept_G} \kappa_{type}(c)$$

$$C = \bigcup_{D_i \in D} \kappa_{nom}(D_i) \cup \bigcup_{c \in concept_G} \kappa_{marker}(c)$$

$$E = \bigcup_{D_i \in D} \kappa(E_{D_i}) \cup \bigcup_{D_i \in D} \kappa(R_{D_i}) \cup \bigcup_{c \in concept_G} \kappa_{edge}(c)$$

$$label = \bigcup_{D_i \in D} label_{D_i}(R_{D_i}) \cup \bigcup_{c \in concept_G} label(\kappa_{marker}(c)) \cup \bigcup_{c \in concept_G} label(\kappa_{type}(c))$$

Figure 4.7: κ -translation of a concept node and its neighbourhood

$$\forall c, c' \in \text{concept}_G, \quad \kappa_{\text{marker}}(c) \equiv_{co} \kappa_{\text{marker}}(c') \text{ iff } (c, c') \in co_{\text{trans}}$$

$$\forall D_i, D_j \in D, \quad \kappa_{\text{nom}}(D_i) \equiv_{co} \kappa_{\text{nom}}(D_j) \text{ iff } (D_i, D_j) \in co_{\text{desc}}$$

In Figure 4.6, the typing of concept nodes has not been represented in the κ -translation. Indeed, the unique concept type does not convey any useful information; it is just a necessary component of the simple conceptual graph syntax.

We need to prove that the proposed translations are meaning preserving:

4.3.12. THEOREM (COMPLETENESS OF THE TRANSLATION). *Given a nested signature Σ with derived signature σ and two nested conceptual graphs G and H over Σ ,*

$$G \sqsubseteq_{NCG} H \text{ iff } G \sqsubseteq_{NCG_{\text{flat}}} H \text{ iff } \kappa(G) \sqsubseteq_{SCG} \kappa(H)$$

4.3.13. LEMMA. *Given a nested signature Σ with derived signature σ and a nested conceptual graphs G over Σ ,*

1. $\forall M$ nested Σ -structure, $M \models_{NCG} G$ iff $\lambda(M) \models_{NCG_{\text{flat}}} G$
2. $\forall M$ flat σ -structure, $M \models_{NCG_{\text{flat}}} G$ iff $\mu(M) \models_{NCG} G$
3. $\forall M$ flat σ -structure, $M \models_{NCG_{\text{flat}}} G$ iff $M \models_{SCG} \kappa(G)$

Proof of Lemma 4.3.13(1): let $M = (W, O, [\cdot]_M, S, R_{\text{acc}})$, $\lambda(M) = (X, [\cdot]_{\lambda(M)})$,

$$\begin{aligned}
M \models_{NCG} G &\Rightarrow \left[\begin{array}{l} \llbracket N_0 \rrbracket_M \neq \odot \Rightarrow \llbracket N_0 \rrbracket_{\lambda(M)} \neq \odot \\ \text{and } \exists \llbracket \cdot \rrbracket_a / (M, \llbracket N_0 \rrbracket_M, \llbracket \cdot \rrbracket_a) \models_{NCG} D_0 \end{array} \right. \quad \text{by Def } \lambda.1, 2 \\
&\quad (M, w, \llbracket \cdot \rrbracket_a) \models_{NCG} D_i \\
&\Rightarrow \left[\begin{array}{l} \forall c \in C_{D_i}, \left[\begin{array}{l} \langle w, \llbracket c \rrbracket_a, \llbracket desc(c) \rrbracket_a \rangle \in R_{acc} \quad (i) \\ (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCG} desc(c) \quad (ii) \\ \llbracket c \rrbracket_a \in \llbracket type(c) \rrbracket_M^w \quad (iii) \end{array} \right. \\ \forall r \in R_{D_i}, \langle \llbracket r(1) \rrbracket_a, \dots, \llbracket r(k) \rrbracket_a \rangle \in \llbracket label(r) \rrbracket_M^w \quad (iv) \end{array} \right. \\
&\Rightarrow \left[\begin{array}{l} \forall c \in C_{D_i}, \left[\begin{array}{l} \llbracket c \rrbracket_a \in X \quad \text{by } (i) \& \text{Def } \lambda.1 \\ \langle w, \llbracket c \rrbracket_a, \llbracket desc(c) \rrbracket_a \rangle \in \llbracket type(c) \rrbracket_{\lambda(M)} \quad \text{by } (i), (iii) \& \text{Def } \lambda.3 \\ (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} desc(c) \quad (ii) \end{array} \right. \\ \forall r \in R_{D_i}, \langle w, \llbracket r(1) \rrbracket_a, \llbracket desc(r(1)) \rrbracket_a, \dots, \llbracket r(k) \rrbracket_a, \llbracket desc(r(k)) \rrbracket_a \rangle \\ \quad \in \llbracket label(r) \rrbracket_{\lambda(M)}^w \text{ by } (i), (iv) \& \text{Def } \lambda.3 \end{array} \right. \\
&\Rightarrow (M, w, \llbracket \cdot \rrbracket_a) \models_{NCGflat} D_i
\end{aligned}$$

We skip the proof of the reciprocal and the one of Lemma 4.3.13(2) which are very similar checks that enough information is conveyed by the model translations.

Proof of Lemma 4.3.13(3): by induction on the structure of G .

Let $\llbracket \cdot \rrbracket_f = \llbracket \kappa_{nom}(D) \rrbracket_a \cup \llbracket \kappa_{marker}(concept_G) \rrbracket_a$

1. $(M, \llbracket D_i \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} c \in D_i$

$$\text{with } c = \boxed{type(c) : marker(c), nom(desc(c)) : desc(c)}$$

$$\text{iff } \left[\begin{array}{l} \langle \llbracket D_i \rrbracket_a, \llbracket c \rrbracket_a, \llbracket desc(c) \rrbracket_a \rangle \in \llbracket type(c) \rrbracket \\ \text{and } (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} desc(c) \end{array} \right.$$

$$\text{iff } \left[\begin{array}{l} \langle \llbracket \kappa_{nom}(D_i) \rrbracket_f, \llbracket \kappa_{marker}(c) \rrbracket_f, \llbracket \kappa_{nom}(desc(c)) \rrbracket_f \rangle \in \llbracket type(c) \rrbracket \\ \text{and } (M, \llbracket desc(c) \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} desc(c) \end{array} \right.$$

$$\text{iff } (M, \llbracket \cdot \rrbracket_f) \models_{SCG} \kappa(graph(c))$$

where $graph(c)$ is the SCG composed of the single node c

2. $(M, \llbracket D_i \rrbracket_a, \llbracket \cdot \rrbracket_a) \models_{NCGflat} r(c_1, \dots, c_k) \in D_i$

$$\text{iff } \langle \llbracket D_i \rrbracket_a, \llbracket c_1 \rrbracket_a, \llbracket desc(c_1) \rrbracket_a, \dots, \llbracket c_k \rrbracket_a, \llbracket desc(c_k) \rrbracket_a \rangle \in \llbracket label_{D_i}(r) \rrbracket$$

$$\text{iff } (M, \llbracket \cdot \rrbracket_f) \models_{SCG} \kappa(r(c_1, \dots, c_k))$$

3. assignments take into account coreferences.

■

It is now straight forward to prove the completeness of the translations.
Proof of Theorem 4.3.12:

$$\begin{aligned} \forall \Sigma - \text{structure } M, \quad M \models_{NCG} G &\Rightarrow M \models_{NCG} H \\ &\Downarrow \text{Lemma 4.3.13(1)} \\ \lambda(M) \models_{NCGflat} G &\Rightarrow \lambda(M) \models_{NCGflat} H \end{aligned}$$

$$\begin{aligned} \forall \sigma - \text{structure } M', \quad M' \models_{NCGflat} G &\Rightarrow M' \models_{NCGflat} H \\ &\Downarrow \text{Lemma 4.3.13(2)} \\ \mu(M') \models_{NCG} G &\Rightarrow \mu(M') \models_{NCG} H \end{aligned}$$

Hence, $G \sqsubseteq_{NCG} H$ iff $G \sqsubseteq_{NCGflat} H$

$$\begin{aligned} \forall \sigma - \text{structure } M', \quad M' \models_{NCGflat} G &\Rightarrow M' \models_{NCGflat} H \\ &\Downarrow \text{Lemma 4.3.13(3)} \\ M' \models_{SCG} \kappa(G) &\Rightarrow M' \models_{SCG} \kappa(H) \end{aligned}$$

Hence, $G \sqsubseteq_{NCGflat} H$ iff $\kappa(G) \sqsubseteq_{SCG} \kappa(H)$

■

4.3.3.2 Complexity of reasoning in nested graphs

Through the previous correspondences, we can directly harvest some results on the complexity of our benchmark problems in the nested framework.

4.3.14. COROLLARY. • *No nested graph is valid.*

- *A nested graph is always satisfiable.*
- *$G \sqsubseteq_{NCG} H$ is NP-complete, for G and H nested conceptual graphs over a common nested-signature.*

Proof: It comes to no surprise that validity and satisfiability are not informative notions for positive graphs.

- By analogy to flat structures having an empty universe, a nested structures can have an empty set of worlds, hence providing no world to interpret the description D_0 and thus no possible assignment.
- A nested graph G is always satisfied by the μ -translation of the canonical model of $\kappa(G)$ (cf. Fact 3.2.4).
- By Theorem 4.3.12, subsumption of nested graphs is in NP. Indeed, the κ -translation is polynomial: for a nested graph G with c concept nodes, r relation nodes and e edges, $\kappa(G)$ is a graph with $(3c + 1)$ concept nodes, $(r + c)$ relation nodes and at most $(3c + 3e)$ edges. The NP lower bound is the one of subsumption in simple graphs as we can encode any simple graph G as a nested graph $\kappa^{-1}(G)$

with a single non-empty description, D_0 , and with each concept node described by an instance of the empty graph labelled with N_0 . It is obvious that $G \sqsubseteq_{SCG} H$ iff $\kappa^{-1}(G) \sqsubseteq_{NCG} \kappa^{-1}(H)$ as any concept node of the premiss introduces a required link to the world $\llbracket N_0 \rrbracket$. ■

Despite the cognitive impact of a modular layout of represented information, the expressive power of the nested fragment is the same as the one of the simple graph fragment. For tractability reasons, we can turn ourselves to guarded quantification.

4.3.15. COROLLARY. *Given a nested signature Σ with derived signature σ and two nested conceptual graphs G and H over Σ , $G \sqsubseteq_{NCG} H$ is decidable in polynomial time if $\kappa(H)$ is a guarded simple graph.*

Proof: Immediate consequence of Theorem 4.3.12 and Theorem 3.3.7. ■

4.3.4 Concluding remarks

In this section, we have presented how some modular knowledge can be represented by a language of nested graphs. To overcome a syntactical lack in the traditional setting of nested conceptual graph, we have introduced nominals. As consequence, the different kinds of knowledge found in the literature under alternative semantics ([CM97, Sim98, CMS98]) can now be expressed in a single framework. Furthermore, by translation to the simple conceptual graph fragment, we have prove the low, though untractable, complexity of reasoning in the nested framework and we have isolated a fragment of nested graphs for which subsumption is tractable.

4.3.4.1 Related work

Peirce's gamma-graphs. Under the name of gamma-system (see e.g., [Rob73, Thi75]), Peirce also studied a modal version of its propositional graphs, but his work remained rather informal and unfinished and, so far, the gamma-system has not yet proved any computational nor cognitive appeals compared to textual propositional logic.

Hybrid and Description logics. Hybrid logics (see for instance Areces' recent thesis [Are00]) and their use of nominals to name worlds in propositional modal logics have been a source of inspiration for the proposed setting of nested graphs. It should be noted that most complexity studies in hybrid logics have focussed on rich fragments for which satisfiability is often difficult or even undecidable.

Description logics are hybrid logics with a different syntax and a high concern for low complexity reasoning tasks. Most relevant to nested conceptual graphs is the fragment \mathcal{ELIRO}^1 , a propositional modal logic with only diamonds, conjunctions and nominal-constants for which Baader et al. [BMT99] propose a graph representation of formulae and a tractable calculus based on graph homomorphisms.

Context logics. Among predicate modal logics, context logics (e.g., [Buv96, MB97, Buv98]) have in common with nested conceptual graphs that modalities are associated to terms. Syntactically, a description ϕ attached to a term t is represented by a modal formula $ist(t, \phi)$ (which reads “from the present context, ϕ is true in the context of t ”). In [Ker99b], a link between nested graphs and the context logic of Buvač is further explored and a calculus combining modal-tableaux and nested projections is proposed for an extension of the language containing negation.

4.3.4.2 Further work.

Guarded nested graphs have been defined as nested graphs that translate into guarded simple ones. It remains to capture on nested representations the constraints that force the guarded property but a formal presentation would require some extensions of the syntax; for instance, crazed graphs (Definition 3.1.13) would need to be able to take part of nested graphs whereas we have deliberately simplify the setting by only allowing descriptions in normal form. An obvious necessary condition is that every description in a nested graph must be a guarded simple graph. Studying the guarded structure of coreference between subgraphs is left for further work.

Secondly, the extension of the language to some restricted forms of negation, such as the limited atomic negation of Chapter 4.1, would also be interesting. Indeed, from description logics and modal propositional logics, we know the tractability of reasoning in some modal fragments including negation. How would guards interact with negations in the modal predicate framework?

Finally, other forms of nesting would also be worth studying. For instance, instead of nesting graphs inside concept nodes, we could have adopted a more usual modal syntax by nesting a graph on the sheet of assertion of another graph (in the style of negated zones in Chapter 4.2). How useful could graph methods be in fragments of usual modal predicate logics?

Conceptual graphs ally symbolic order-sorted reasoning with the visual understanding of drawn information. The role of graphical items is twofold: (a) they compose networks that host pieces of knowledge; (b) they are also the fuel of reasoning methods. The main results we have achieved relate the diagrammatic framework to the well-established area of symbolic reasoning. In this context, conceptual graphs turn out to be related to current interests in fine-structure studies of the expressiveness/complexity balance. We have been able to occupy relevant positions in a landscape of fragments of known logical languages.

We have also emphasised salient visual aspects of the diagrams. In our opinion, a better understanding of the computational power of visual items may call for escaping the shackles of the traditional symbolic computation theory. This chapter concludes with a brief discussion of some graphical factors that one would expect to play a significant role in more visually oriented computation models.

In short. The main results that we have achieved can be summarised in:

- (i) a systematic exploration of conceptual graph languages
- (ii) an analysis of structural properties reappearing in several fragments
- (iii) the positioning of conceptual graphs in a landscape of standard logics

More concretely, we can recall a few specific results.

New horizons in poor fragments. Diverging from a classical trend in “pure logic” where the satisfiability problem has remained the predominant benchmark for judging a fragment, logics applied to computer science, knowledge representation or natural language processing emphasise the practical computational feasibility of “poor languages”. Applied logics also stress the importance of decision problems such as consequence, model checking or model comparison. When we step down from the pedestal of full first-order logic, other decision problems gain their independence from satisfiability and therefore, become a relevant source of

information; “saying less is often hearing more”. Besides observing the separations of decision problems, new collapses are appear as well. One such equivalence that we have met at the low level of existential conjunctive logic, the domain of simple conceptual graphs and cliques, concerned the relation between consequence, truth and model comparison:

$$\text{for two simple conceptual graphs } G \text{ and } H, \\ G \sqsubseteq H \text{ iff Minimal-Model}(G) \models H \text{ iff } H \text{ projects onto Normal-form}(G)$$

This key equivalence of decision problems has pushed forward the simple graph fragment to a central position in the study.

Mappings reign. The model-theoretic approach that we have proposed, reveals that simple graphs enjoy a direct resemblance to the structures they represent. Therefore, the meaning of a simple graph is conveyed by a structure preserving mapping. Furthermore, in the light of the previous equivalence between decision problems, this direct embedding semantics stands up as a guarantee for the adequacy of an embedding calculus, projection.

The guarded safety belt. It is common knowledge that tree structures are a key for efficiency in computation. Tree properties of modal logics translate into guarded fragments of classical logic. By adapting the notion of guards to simple graph diagrams, we have brought to light a fragment in which subsumption can be decided in polynomial time. This guarded fragment of simple graphs includes all previously known tractable fragments of simple graphs. A notable characteristic of guarded graphs is that the traditional notion of tree, as non-cyclic paths of edges, has left the place for one where acyclicity concerns coreferences between atomic subgraphs. For this guarded fragment, we have proposed a polynomial time projection algorithm that builds a mapping by eliminating impossible local correspondences along a single run through the recursive structure of the source guarded graph.

Extensions. While enriching simple graphs with additional logical connectors, we have kept in mind those properties that made us advocate for simple graphs: (i) neat semantics by direct embedding into resembling structures, (ii) the use of graphical calculi that take the best of the diagrammatic features in the representations and (iii) the computational power of guards. In particular, the fragments of discriminated graphs with atomic negation and nested graphs can be embedded into the language of simple graphs. As a consequence, guarded restrictions of these languages enjoy a tractable consequence problem. Furthermore, the proof-tree construction of tableau algorithms has been combined to projections in an analytical calculus for first-order logic conceptual graphs.

The connections that we settled between conceptual graphs and symbolic logics are bidirectional. We can learn from standard logic interesting properties of graph languages and adapt symbolic techniques to overcome the limitations of some pure graphical reasoning. Furthermore, links to symbolic logics allow new developments of conceptual graphs to be more systematic and understandable. Reciprocally, we can point out structural properties that naturally emerge from the diagrams and translate visual techniques into the symbolic world.

Open questions. All the results that we have obtained suggest new questions. We raise a few of them that, in our opinion, could present some interest for the conceptual graph research agenda.

- **Multi-projection.** Projection that is based on a single homomorphism, is incomplete for the language of simple graph extended with atomic negation. However, could we define an efficient consequence algorithm with simultaneous mappings to cope with the disguised disjunctions in the fragment? In other words, can we incorporate the branching idea of tableaux directly into the projection method?
- **New graph languages.** In the line of this study, well-behaved modal and hybrid logics may remain an important source of inspiration to decidable conceptual graph languages. In particular, we may wonder which “modal negations” could elegantly fit the diagrammatic framework.
There might also be relevant graphical applications of non-classical semantics such as linear logic that already uses diagrammatic proof-nets.
- **Graphical properties of guarded-FOL.** Could a graph approach bring a novel view on the consequence problem in the full guarded fragment of FOL or in rich subfragments of it? Can the clique property of loosely guarded fragments be exploited from a purely graphical point of view?
- **Usable modal predicate logics.** Tractable modal predicate logics may find direct applications to network technologies and partitioned knowledge bases. Can we expand the nested fragment presented in this work into new tractable areas? How far can we avoid well-known problems on cross world quantification?
- **Meta-guards.** The tree property has occurred recurrently in this thesis: under its most usual setting of acyclic paths of edges in a graph and later, in the acyclicity of coreferences between atomic subgraphs. It would be very interesting to lift again the observation view point and consider acyclic networks of (not necessarily atomic) subgraphs that are bound in size. Some recent developments on hypertree characterisations of conjunctive queries[GLS01] in database theory suggest that the projection method could preserve its polynomial complexity.

The general patterns behind significant complexity results are largely matters like: (a) guarded syntax, (b) the tree model property, and (c) embedding semantics. This outcome of our expressiveness/complexity analysis raises the question how such features relate to actual visual reasoning. The remainder of this chapter is a very brief discussion of the main points as we see them.

But once again, why are drawings efficient? With no adapted tools at hand (i.e., the lack of a geometrical computation model of equal standing to Turing machines), we cannot provide a definitive answer. However, in the light of our gathered experience on drawings, we can briefly discuss the potential relevance of two graphical factors that one would expect to play a significant role in more visually oriented computation models.

- **Direct mapping to reality.** It is generally acknowledged that a large part of the cognitive efficiency of drawings resides in their faithfulness to what they represent.

In the specific case of conceptual graphs, the resemblance of pictures to their model has recurrently emerged. In an obvious way, this property eases the interpretation process by providing a direct match between the intuitive meaning and the formal one. For example, a path is a self-speaking connection between items. Nesting conveys the pictorial message of a delimited zone reachable from another one; formally, world accessibility. Furthermore, we have related this close resemblance between representation and represented to a collapse of distinct decision problems.

We should note that there is a preliminary condition for a drawing to convey the intended intuitive meaning. If complexity studies often try to be as neutral as possible with respect to the layout of information¹, the disposition of graphical items in a drawing is crucial. Clarity is not a property of drawings in general, but only of “good” ones. A good drawing must mirror the information it represents. Drawing writers have to respect some Gricean maxims of quantity in order to convey an informative message. Such maxims are proposed by Oberlander [Obe96] in the framework of computer-assisted design in electronics or by Tufte in his compendium of graphics [Tuf83]. Closer to our concept languages, automatised tools have been developed for drawing “readable” lattices of concepts in formal concept analysis; see e.g., [Wil89]. It seems difficult to define quality guidelines that would embrace all kinds of graphics. However, this is possible in specific domains such as

¹We have seen that the layout can also be relevant to classical complexity measures, witness the case of a structure which is expanded with a universal relation while preserving a tolerable size: only the tuples of objects that are not in relation with each other are encoded. Metaphorically, we can imagine this situation as if the negative of a photo would be lighter to carry than the usual positive printing, but just as informative.

the conceptual graph framework. For example, it is often taken for granted that branches of a tree should not cross each other and that an orientation is chosen in the two-dimensional plane from the root to the leaves. Complete graphs (cliques) are almost always represented with a regular placing of the nodes on a cycle. Lattice drawings enjoy symmetrical patterns.

With its linear disposition of information, the tape of a Turing machine cannot faithfully capture simple geometrical features such as a branching or a cycle. Hence, following a cyclic path on any linear representation of a graph would appear to cost more efforts than shifting a pencil along the same path in a two-dimensional drawing of the structure.

So, if one admits “good drawing norms” as part of a graphical syntax, the faithfulness of drawings is efficiently exploited by the human vision to anchor recognised geometrical constraints in a representation to similar constraints in the represented situation. On the contrary, linear representation systems will not grant the easiness of recognising some geometrical patterns.

- **Gestalt view.** The human ability of recognising shapes and mapping them on each other is striking. One can argue that it is the fruit of a continuous training starting in early childhood when we are asked to learn concepts on images, assemble similar representations or picture reality.

The perception of forms on different levels of “shallowness”² in a representation has also been salient in guarded and nested drawings. By making abstraction of some “details”, we have been able to extract spinal structures that were essential to the efficiency of computing the reasoning.

The possibility of perceiving irregularities in a gestalt view of a drawing enables to recognise regular patterns such as trees or cliques. Acyclicity seems to be verifiable at first sight (provided the respect of norms discussed previously): by bringing an irregularity in the picture, the intersection of two branches must disfigure the general form so much that it captures the attention in an overview examination. Trees are not only easy to be recognised; they appear to facilitate form matching. The mapping of a tree on another graph is guided by the direction chosen in the representation. This suggests the one round elimination process that has been adopted for guarded projection.

Based on our conventional representation of cliques (i.e., with a regular cyclic disposition of nodes), the verification that a given graph is really complete also seem to be an instantaneous process of finding if any irregularity attracts the attention. This does not mean that the clique problem

²From a symbolic perspective, recognising and formally applying levels of under-specification in computational linguistics is a goal of the project *Computing with meaning*; see <http://turing.wins.uva.nl/mdr/Projects/CoMe/index.html>

(i.e., finding a clique of a given size in a graph) should be easier when manipulating pictures, but more that by visualising an almost complete graph, we obtain “for free” its complementary; “the photo and its negative”³.

To take a final concrete example, the perception of global shapes and irregularities also forms the foundation of a well spread technique of tendency analysis on stock-exchange graphics. We note that these graphics are only converted to matrices of numbers to serve as representation for computer treatment; People seem to extract/perceive a more easily usable information from reading the drawings.

The gestalt view on drawings seems to offer the possibility of immediately separating distinct levels of information and capturing salient irregularities. On the contrary, extracting similar information from “equivalent” linear representation often necessitates a more costly systematic analysis of the data.

The upshot of our discussion is that there are further features of visual reasoning not captured by our conservative approach. But we also see this as a worth. The symbolic approach in the preceding chapters allows us to see more clearly what is genuinely visual and what isn't.

Brief summary. Visual reasoning is a key area of current interest where a lot of disciplines meet: philosophy, computer science, logic, linguistics. Conceptual graphs live at the interface of many of these. In this study we have investigated them in parallel with classical perspectives from logic, language and computation, exploiting analogies wherever possible. One general advantage of this cautious approach is a certain discipline of thinking in an area which is sometimes dominated by appealing metaphors. Nevertheless, there also remains an empirical cognitive science dimension to the workings of visual patterns, which transcends what can be gotten from Turing machines and language fragments and therefore, this thesis is not the last word on pictures...

³In a symbolic representation, we can also minimally encode the graph as the set of pairs of points that are not connected, but in the picture we do not need to “cheat” on the input.

Bibliography

- [AB96] Gerard Allwein and Jon Barwise. *Logical Reasoning with Diagrams*. Oxford University Press, 1996.
- [ABM99a] Carlos Areces, Patrick Blackburn, and Maarten Marx. Hybrid logics. Characterization, interpolation and complexity. Technical Report PP-1999-07, ILLC, University of Amsterdam, 1999. To appear in *Journal of symbolic logic*.
- [ABM99b] Carlos Areces, Patrick Blackburn, and Maarten Marx. A road-map of the complexity of hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999.
- [ABM00] Carlos Areces, Patrick Blackburn, and Maarten Marx. The computational complexity of hybrid temporal logic. *Logic Journal of the IGPL*, 8(5):653–679, 2000.
- [ABN98] Hajnal Andréka, Johan van Benthem, and István Németi. Modal logics and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [Are00] Carlos Areces. *Logic Engineering: the Case of Description and Hybrid Logics*. PhD thesis, ILLC, University of Amsterdam, 2000.
- [BB98] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language: A First Course in Computational Semantics. Studies in Logic, Language, and Information*, CSLI Press, 1998.

- [BBH⁺90] Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, and J.H. Siekman. Concept logics. In *Computational Logic Symposium Proceedings, Brussels, Belgium*. Springer-Verlag, 1990.
- [BE82] Johan van Benthem and Jan van Eijck. The dynamics of interpretation. *Journal of Semantics*, 1(1):3–20, 1982.
- [BE98] Jon Barwise and John Etchemendy. Computers, visualization, and the nature of reasoning. In Terrell Ward Bynum and James Moor, editors, *The Digital Phoenix: How Computers are Changing Philosophy*. Blackwell Publishers, 1998.
- [Ben85] Johan van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, The Humanities Press, Atlantic Heights (N.J.), 1985.
- [Ben88] Johan van Benthem. *A Manual of Intensional Logic*. CSLI Lecture Notes 1, Center for the Study of Language and Information, Stanford University, 1988.
- [Ben96] Johan van Benthem. *Exploring Logical Dynamics*. Studies in Logic, Language and Information, FOLLI and CSLI Publications, 1996.
- [Ben97] Johan van Benthem. Dynamic bits and pieces. Technical Report LP-97-01, ILLC, University of Amsterdam, 1997.
- [BGG97] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1997.
- [BHR90] Karl-Hans Bläsius, Ulrich Hedtstück, and Claus-Rainer Rollinger. Sorts and types in artificial Intelligence. In *Proceedings of Workshop Eringerfeld, FRG, April 1989*, volume 418 of *LNAI*. Springer-Verlag, 1990.
- [BKM99] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.
- [BMT99] Franz Baader, Ralf Molitor, and Stefan Tobies. Tractable and decidable fragments of conceptual graphs. In Tepfenhart and Cyre [TC99], pages 480–493.
- [BP83] Jon Barwise and John Perry. *Situation and Attitudes*. MIT Press, Cambridge, MA, 1983.

- [BRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.
- [BS28] Paul I. Bernays and Moses Schönfinkel. Zum entscheidungsproblem der mathematischen logik. *Math. Annalen*, 99, 1928.
- [BS85] Ron J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [BT99] Patrick Blackburn and Miroslava Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.
- [Buv96] Saša Buvač. Quantificational logic of context. In *proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 600–606. AAAI/MIT Press, 1996.
- [Buv98] Saša Buvač. *Contextual Information Integration*. PhD thesis, Stanford University, 1998.
- [CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of the Workshop on Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, 1981.
- [Chu36] Alonzo Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936.
- [CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.
- [CM92] Michel Chein and Marie-Laure Mugnier. Conceptual graphs, fundamental notions. *RIA*, 6.4:365–406, 1992.
- [CM97] Michel Chein and Marie-Laure Mugnier. Positive nested conceptual graphs. In Lukose et al. [LDK⁺97], pages 95–109.
- [CMS98] Michel Chein, Marie-Laure Mugnier, and Geneviève Simonet. Nested graphs: A graph-based knowledge representation model with FOL semantics. In Anthony G. Cohn, Lenhard K. Schubert, and Stuart C. Shapiro, editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy*, pages 524–535. Morgan Kaufmann, 1998.

- [CSO94] Richard Cox, Keith Stenning, and Jon Oberlander. Graphical effects in learning logic: reasoning, representation and individual differences. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society, Georgia, USA, 1994*.
- [DLN⁺92] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, Bernhard Hollunder, Werner Nutt, and Alberto Marchetti Spaccamela. The complexity of existential quantification in concept languages. *Journal of Artificial Intelligence*, 53, 1992.
- [DLNN97] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. *Information and Computation*, 134, 1997.
- [DLNS94] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4(4), 1994.
- [DPP01] Agostino Dovier, Carla Piazza, and Alberto Policriti. A fast bisimulation algorithm. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France*, volume 2102 of *LNCS*, pages 79–90. Springer, 2001.
- [EF95] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 1995.
- [Eij98] Jan van Eijck. Dynamic reasoning without variables. Technical Report INS-R9801 / ISSN 1386-3681, CWI, Amsterdam, 1998.
- [ELRS95] Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa, editors. *Proceedings of ICCS'95, Santa Cruz, USA*, volume 954 of *LNAI*. Springer-Verlag, 1995.
- [FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [Fit90] Melvin Fitting. *First Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.
- [Fre79] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879.
- [FTU99] Carles Farré, Ernest Teniente, and Toni Urpí. The constructive method for query containment checking. In A. Min Tjoa Trevor J.

- M. Bench-Capon, Giovanni Soda, editor, *Database and Expert Systems Applications, Proceedings of the 10th International Conference DEXA '99*, volume 1677 of *LNCS*, pages 583–593. Springer, 1999.
- [FV99] T. A. Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: a study through datalog and group theory. *SIAM Journal on Computing*, 104, 1999.
- [Gab71] Dov Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*. Reidel, 1971.
- [Gai82] Haim Gaifman. On local and nonlocal properties. In J. Stern, editor, *Proceedings of Logic Colloquium'81*, pages 105–135. North Holland, 1982.
- [GB97] Fausto Giunchiglia and Paolo Bouquet. Introduction to contextual reasoning. an artificial Intelligence perspective. *Perspectives on Cognitive Science*, 3, 1997.
- [GC97] David Genest and Michel Chein. An Experiment in document retrieval using conceptual Graphs. In Lukose et al. [LDK⁺97], pages 489–504.
- [Gho96] Bikash C. Ghosh. *Conceptual Graph Language : A Language of Logic and Information in Conceptual Structures*. PhD thesis, Asian Institute of Technology, Bangkok, Thailand, 1996.
- [GKV97] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.
- [GLS99] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems PODS'99*, pages 21–32. ACM, 1999. To appear in *Journal of Computer and System Sciences*.
- [GLS01] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, Marshals, and Guards: Game Theoretic and Logical Characterizations of Hypertree Width. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, 2001.
- [Grä92] Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theoret. Comp. Sci.*, 101(1):35–57, 1992.

- [Grä99] Erich Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
- [Gro96] Martin Grohe. Equivalence in finite-variable logics is complete for polynomial-time. *Proc. 1996 IEEE Symposium on Foundations of Computer Science*, 1996.
- [GS91] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.
- [GSV96] Jeroen Groenendijk, Martin Stokhof, and Frank Veltman. Changez le contexte ! *Langages*, 123:8–29, 1996.
- [GW95] Bikash C. Ghosh and Vilas Wuwongse. A direct proof procedure for definite conceptual graph programs. In Ellis et al. [ELRS95], pages 158–172.
- [GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis*. Springer, 1999.
- [Hal95] Joseph Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75, 1995.
- [Ham95] Eric M. Hammer. *Logic and Visual Information*. CSLI Publications and FOLLI, 1995.
- [Ham98] Eric M. Hammer. Semantics for Existential graphs. *Journal of Philosophical Logic*, 27:489–503, 1998.
- [Hem01] Edith Hemaspaandra. The complexity of poor man’s logic. *Journal of Logic and Computation*, 11(4):609–622, 2001. Also in the Essays Dedicated to Johan van Benthem on the Occasion of his 50th Birthday, AUP, Amsterdam, 1999.
- [HM92] Joseph Y. Halpern and Yoram Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54, 1992.
- [HV91] Joseph Y. Halpern and Moshe Y. Vardi. Model checking vs. theorem proving: A manifesto. In J. Allen, R. E. Fikes, and E. Sandewall, editors, *Proceedings 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, KR’91*, pages 325–334. Morgan Kaufmann Publishers, San Mateo, CA, 1991.
- [Imm82] Neil Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

- [Imm99] Neil Immerman. *Descriptive complexity*. Springer, 1999.
- [JJ92] Joseph Já Já. *An introduction to parallel algorithms*. Addison-Wesley, 1992.
- [Kah62] A. Kahr. Improved reductions of the entscheidungsproblem to subclass of $\forall\exists\forall$ formulas. In *Proc. Symp. on Mathematical Theory of Automata*, pages 57–70, Brooklyn Polytechnic Institute, New York, 1962.
- [Kal36] Laszlo Kalmár. Zurückführung des entscheidungsproblems auf den fall von formeln mit einer einzigen, binären funktionsvariablen. *Compositio Mathematica*, 4, 1936.
- [Kam81] Hans Kamp. A theory of truth and semantic representation. In Jeroen Groenendijk, Theo Janssen, and Martin Stokhof, editors, *Formal Methods in the Study of Language*. Mathematical Centre, Amsterdam, 1981.
- [Kar72] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New-York, 1972.
- [Ker96] Gwen Kerdiles. Conceptual graphs: deduction from a perspective of graph matching. In *proceedings of Accolade'96, Nijmegen, The Netherlands*, 1996.
- [Ker97] Gwen Kerdiles. Projection, a unification procedure for tableaux in conceptual graphs. In Didier Galmiche, editor, *Proceedings of Tableaux'97, Pont à Mousson, France*, volume 1227 of *LNAI*, pages 216–230. Springer-Verlag, 1997.
- [Ker99a] Gwen Kerdiles. Dynamic semantics for conceptual graphs. In Tepfenhart and Cyre [TC99], pages 494–507.
- [Ker99b] Gwen Kerdiles. Graph matching in contextual reasoning. Technical Report 99034, LIRMM, Université Montpellier II, France, 1999.
- [Koz81] Dexter Kozen. Positive first-order logic is np-complete. *IBM Journal of research and development*, 25(4):327–332, 1981.
- [KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.
- [KR96] Hans Kamp and Uwe Reyle. A calculus for first order discourse representation structures. *JoLLI*, 5(3-4):297–348, 1996.

- [KS97] Gwen Kerdiles and Eric Salvat. A sound and complete cg proof procedure combining projections with analytic tableaux. In Lukose et al. [LDK⁺97], pages 371–385.
- [Kur94] Robert P. Kurshan. *Computer-aided verification of coordinating processes*. Princeton University Press, 1994.
- [KV00] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000. A preliminary version appeared in Proc. of PODS’98 pp.205-213.
- [Lad77] Richard E. Ladner. The computational complexity of provability in systems of modal logic. *SIAM Journal of computing*, 6:467–480, 1977.
- [LB87] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
- [LDK⁺97] Dikson Lukose, Harry Delugach, Mary Keeler, Leroy Searle, and John F. Sowa, editors. *Conceptual Structures: Fulfilling Peirce’s Dream (Proceedings of ICCS’97, Seattle, USA)*, volume 1257 of *LNAI*. Springer-Verlag, 1997.
- [Leh92] Fritz Lehmann, editor. *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford, 1992.
- [Lew80] H. R. Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21:317–353, 1980.
- [LMSS93] Alon Y. Levy, Inderpal S. Mumick, Yehoshua Sagiv, and Oded Shmueli. Equivalence, query-reachability and satisfiability in Datalog extensions. In *Proceedings of the PODS’93*, pages 109–122. ACM, 1993.
- [Löw15] Leopold Löwenheim. Über möglichkeiten im relativkalkül. *Mathematische Annalen*, 76:228–251, 1915.
- [LS93] Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In *Proceedings of the 19th VLDB Conference*, pages 171–181. Morgan Kaufmann, 1993.
- [LS95] Alon Y. Levy and Yehoshua Sagiv. Semantic query optimization in datalog programs. In *Proceedings of the PODS’95*, pages 163–173. ACM, 1995.

- [Mas98] Fabio Massacci. *Efficient Approximate Deduction and Application to Computer Security*. PhD thesis, Università Degli Studi Di Roma "La Sapienza", 1998.
- [MB97] John McCarthy and Saša Buvač. Formalizing context (expanded notes). In A. Aliseda, R. van Glabbeek, and D. Westerståhl, editors, *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. Center for the Study of Language and Information, Stanford University, 1997.
- [MC92] Marie-Laure Mugnier and Michel Chein. Polynomial algorithms for projection and matching. In *Proceedings of the 7th Annual Workshop on Conceptual Graphs*, pages 49–58, New Mexico State University, Las Cruces, New Mexico, 1992.
- [MC96] Marie-Laure Mugnier and Michel Chein. Représenter des connaissances et raisonner avec des graphes. *RIA*, 10(1):7–56, 1996.
- [MC98] Marie-Laure Mugnier and Michel Chein, editors. *Conceptual Structures: Theory, Tools and Applications (Proceedings of ICCS'98, Montpellier, France)*, volume 1453 of *LNAI*. Springer-Verlag, 1998.
- [McC87] John McCarthy. Generality in artificial Intelligence. In *ACM Turing Award Lectures, The First Twenty Years*. ACM Press, 1987.
- [McD76] Drew V. McDermott. Artificial intelligence meets natural stupidity. *SIGART Newsletter, ACM*, 57, 1976.
- [McM93] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.
- [Mil90] Robin Milner. Operational and algebraic semantics on concurrent processes. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 1201–1242. Elsevier Science and MIT Press, 1990.
- [Min74] Marvin Minsky. A framework for representing knowledge. In Ron J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*. Morgan-Kaufmann, 1974.
- [Min00] Guy W. Mineau. The extensional semantics of the cg formalism. In Bernhard Ganter and Guy W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues, 8th International Conference on Conceptual Structures, ICCS 2000, Darmstadt, Germany, August 14-18, 2000, Proceedings*, volume 1867 of *LNCS*. Springer, 2000.

- [Mug92] Marie-Laure Mugnier. *Contributions algorithmiques pour les graphes d'héritage et les graphes conceptuels*. PhD thesis, LIRMM, Université Montpellier II, 1992.
- [Mug95] Marie-Laure Mugnier. On generalization / specialization for conceptual graphs. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:325–344, 1995.
- [Niv98] Hans de Nivelles. A resolution decision procedure for the guarded fragment. In Claude Kirchner and Hélène Kirchner, editors, *Automated Deduction - CADE-15, 15th International Conference on Automated Deduction, Lindau, Germany*, volume 1421 of *LNCS*, pages 191–204. Springer, 1998.
- [Obe96] Jon Oberlander. Grice for graphics: pragmatic implicature in network diagrams. *Information Design Journal*, 8, 1996.
- [Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [PD98] Anne Preller and Patrice Duroux. Proof graphs. Technical Report 98198, LIRMM, Université Montpellier II, France, 1998.
- [Pei58] Charles Sanders Peirce. *Collected Papers of C.S. Peirce*. C. Hartshorne, P. Weiss and A. Burks, editors. Harvard University Press, Cambridge, MA, 1931-1958.
- [Pre98] Susanne Prediger. Simple concept graphs: A logic approach. In Mugnier and Chein [MC98], pages 225–239.
- [PTB85] Robert Paige, Robert E. Tarjan, and Robert Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40:67–84, 1985.
- [Rob73] Don D. Roberts. *The existential graphs of C.S. Peirce*. Mouton, The Hague, 1973.
- [Rob92] Don D. Roberts. The existential graphs. In Lehmann [Leh92], pages 639–663.
- [Rou97] William C. Rounds. Feature logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science, 1997.
- [Sal97] Eric Salvat. *Raisonnement avec des opérations de graphes: graphes conceptuels et règles d'inférence*. PhD thesis, LIRMM, Université Montpellier II, 1997.

- [Sar91] Yatin P. Saraiya. *Subtree elimination algorithms in deductive database*. PhD thesis, Stanford University, 1991.
- [Sch91] Klaus Schild. A correspondence theory for terminological logics. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th IJCAI*, pages 466–471. Morgan Kaufmann, 1991.
- [SE46] Antoine de Saint-Exupéry. *Le Petit Prince*. Gallimard, 1946.
- [Shi93] Sun-Joo Shin. *Peirce and the logical status of diagrams*. History and Philosophy of Logic, 1993.
- [Shi95] Atsushi Shimojima. *On the Efficacy of Representation*. PhD thesis, Indiana University, 1995.
- [Sim98] Geneviève Simonet. Two FOL semantics for simple and nested conceptual graphs. In Mugnier and Chein [MC98], pages 240–254.
- [Sko20] Thoralf Skolem. Logisch-kombinatorische untersuchungen über die erfüllbarkeit oder beweisbarkeit mathematischer sätze nebst einem theorem über dichte mengen. *Norsk Vid-Akad. Oslo Mat.-Natur Kl. Skr.*, 4, 1920.
- [SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th Annual ACM Symposium on the Theory of Computing*, pages 1–9, New York, 1973. Association for Computing Machinery.
- [SM96] Eric Salvat and Marie-Laure Mugnier. Sound and complete forward and backward chaining of graph rules. In Peter W. Eklund, Gerard Ellis, and Graham A. Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua (Proceedings of ICCS'96, Sydney, Australia)*, volume 1115 of *LNAI*, pages 248–262. Springer-Verlag, 1996.
- [Smu68] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.
- [Sow84] John F. Sowa. *Conceptual Structures, Information Processing in Mind and Machine*. Addison Wesley, 1984.
- [Sow92] John F. Sowa. Conceptual graph summary. In Timothy E. Nagle, Janice A. Nagle, Laurie L. Gerholz, and Peter W. Eklund, editors, *Conceptual Structures: Current Research and Practice*, pages 3–51. Ellis Horwood Series in Workshops, 1992.
- [Sow99] John F. Sowa. *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole, 1999.

- [Spa93] Edith Spaan. *Complexity of Modal Logics*. PhD thesis, ILLC, University of Amsterdam, 1993.
- [SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNAI*. Springer-Verlag, Edited by J.Siekman, 1989.
- [SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attribute concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
- [SW90] Peter H. Schmitt and Wolfgang Wernerke. Tableau calculus for Order sorted logic. In *Sorts and Types in Artificial Intelligence*, volume 418 of *LNAI*, pages 49–60. Springer-Verlag, 1990.
- [TC99] William Tepfenhart and Walling Cyre, editors. *Conceptual Structures: Standards and Practices (Proceedings of ICCS'99, Blacksburg, VA, USA)*, volume 1640 of *LNAI*. Springer-Verlag, 1999.
- [Thi75] P. Thibaud. *La logique de Charles Sanders Peirce*. Editions de l'Université de Provence, 1975. Etudes Philosophiques 1.
- [Tuf83] Edward Rolf Tufté. *The visual display of quantitative information*. Graphic Press, 1983.
- [Tur37] Alan M. Turing. On computable numbers, with an application to the *Entscheidungsproblem*. In *Proceedings of the London Mathematical Society*, volume 2-42, pages 230–265, 1937.
- [Var82] Moshe Y. Vardi. On the complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on the Theory of Computing*, pages 137–146, 1982.
- [Var95] Moshe Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of 14th Ann. ACM Symposium on Principles of Database Systems PODS 95*, pages 266–276, 1995.
- [Var97] Moshe Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. Kolaitis, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 31, pages 149–184. American Math. Society, 1997.
- [Via97] Victor Vianu. Databases and finite-model theory. In N. Immerman and P. Kolaitis, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 31, pages 97–148. American Math. Society, 1997.

- [Vor99] Andrei Voronkov. The ground-negative fragment of first-order logic is π_2^p -complete. *Journal of Symbolic Logic*, 64(3):984–990, 1999.
- [Wer95] Michel Wermelinger. Conceptual graphs and first-Order logic. In Ellis et al. [ELRS95], pages 323–337.
- [Wil89] Rudolf Wille. Lattices in data analysis: how to draw them with a computer. In Ivan Rival, editor, *Algorithms and order*. Kluwer, Dordrecht-Boston, 1989.

Index

- absurd graph, 115
- alphabet, 76

- classical decision problem, 39
- clique problem, 36
- computation model, 30, 164
 - complexity class, 35
 - encoding, 31
- conceptual graph
 - FOL translation, 132
 - guarded fragment, 142
 - negation, 8, 21, 114, 128
 - semantics, 132
 - tableaux, 133
- conceptual graph rules, 142
- consequence problem, 42
- context, 12
 - logic, 159

- description logic, 11, 29, 71, 158
- diagram, 4
 - derivative meaning, 17
 - embedding, 15, 164
 - gestalt view, 14, 165
 - limited abstraction, 16
 - semantic conventions, 17, 18, 164
 - space partition, 14
 - spinal structure, 15
- discourse representation theory, 22, 141

- dynamic semantics, 23

- FOL, 4, 5, 39, 44
 - $\{\exists, \forall, \wedge, \vee\}$, 38
 - $\{\exists, \wedge, \neg_a\}$, 54, 60
 - $\{\exists, \wedge, \vee, \neg_a\}$, 55
 - $\{\exists, \wedge\} + MP$, 52
 - $\{\exists, \wedge\}$, 28, 36, 47
 - $\{\exists, \wedge\}$ guarded, 64
 - guarded fragment, 40, 46
 - order-sorted, 7, 51
- formal concept analysis, 78
- free ride, 18
- Frege's logical diagrams, 4

- Gaifman graph, 100
- guarded simple conceptual graph, 103
 - $FOL_{\{\exists, \wedge\}}$ characterisation, 64
 - covering, 104
 - projection algorithm, 107
 - translation, 105

- hybrid logic, 11, 63, 158
 - $\{\diamond, \wedge, N, @, \downarrow\}$, 70
 - $\{\diamond, \wedge, \neg, N, @\}$, 71
 - $\{\diamond, \wedge, \neg, \downarrow\}$, 71

- lattice, 78

- meaning postulate, 51
- modal logic, 63

- $\{\diamond, \wedge, \neg\}$, 69, 70
 - $\{\diamond, \wedge, \neg_a\}$, 69
 - $\{\diamond, \wedge\}$, 69
 - bounded, 71
- model checking, 43
- model comparison, 44
- nested graph, 9, 142
 - embedding semantics, 150, 153
 - guarded fragment, 158
 - nested structure, 149
 - signature, 147
 - syntax, 147
 - translation, 151, 153
- Peirce's logical diagrams, 5, 141, 158
- polarised graph, 114
 - discriminated fragment, 121
 - embedding semantics, 115
 - guarded fragment, 127
 - normal form, 125
 - projection, 118, 126
 - signature, 114
- positivising, 60, 115, 118
- problem equivalences, 48, 60, 94
- query containment, 42
- satisfiability problem, 42
- semantic network, 11
- simple conceptual graph, 6, 76
 - canonical model, 93
 - crazed form, 85
 - tree covering, 100
 - embedding semantics, 81
 - guarded, 64, 103–105
 - normal form, 83
 - projection, 8, 91, 107
 - signature, 51, 77
 - syntax, 79
 - translation, 88, 99
- simple conceptual tree, 58, 98
 - projection, 99
 - syntax, 98
- subsumption, 42
- support, 7, 51, 76
 - translation, 87

Samenvatting

Pictografische talen komen in vrijwel elk domein voor, van verkeersborden tot technische vormgeving of abstracte kunst. Computerwetenschap vormt hierop geen uitzondering. Het vraagstuk van het succes van visuele informatie in menselijke communicatie, en het geautomatiseerde gebruik ervan, nemen een prominente plaats in op de agenda van de kunstmatige intelligentie. Rekening houdend met uiteenlopende aspecten van grafische talen in kennisweergave, positioneert en ontwikkelt deze dissertatie een specifiek grafisch raamwerk, conceptuele grafen, op een kruispunt van logica, taal en berekenbaarheid.

Diverse cognitieve en linguïstisch efficiënte kenmerken van plaatjes en tekeningen spelen een onbetwistbare rol in de menselijke en mens-machine communicatie. Naast voordelen op het gebied van representatie van informatie, onderstreept de computationele efficiëntie van sommige diagram-klassen de relevantie van tekeningen in geautomatiseerd redeneren.

In deze dissertatie wordt computationele complexiteit uitgelegd in traditionele symbolische termen. Dit legt een gemeenschappelijke basis voor een nuttige interactie tussen normale textuele logica en grafische talen. Ten eerste onthullen de hier bestudeerde grafische systemen het bestaan van logische fragmenten met aantrekkelijke computationele complexiteit, die buiten de normale paden van de symbolische logica vallen. Omgekeerd blijkt een aantal symbolische technieken zich goed aan te passen aan onze grafische raamwerken. Bijvoorbeeld het begrip van 'wachters' voor kwantor-uitdrukkingen, dat oorspronkelijk opkwam bij de vertaling van modale logica in klassieke logica, omschrijft een nieuwe visuele notie van bomen in het conceptuele graaf-paradigma. Ten tweede kunnen redeneer-technieken wederzijds worden uitgewisseld of gecombineerd. Ten slotte bieden cognitieve aspecten, die worden herkend in de perceptie en manipulatie van diagrammen, nieuwe suggesties voor het uitbreiden van gevestigde symbolische computatie-modellen met additionele visuele kenmerken.

Het centrale thema van deze dissertatie is het verkennen van de interacties tussen conceptuele graaf-fragmenten en symbolische logica, in het licht van stan-

daard symbolische complexiteitsmodellen. De voornaamste resultaten, die worden verkregen, betreffen grafische bewijs-methoden voor gevolgtrekkingsproblemen en hun complexiteitsanalyse in verschillende talen van conceptuele grafen. Door de studie in een breder perspectief van visuele informatie in kunstmatige intelligentie te plaatsen hopen we bovendien bij te dragen aan een algemener onderwerp: het beter begrijpen van de formele aspecten van redeneren met diagrammen. Dit is de noodzakelijke basis voor verdere vruchtbare verbindingen tussen symbolische en grafische perspectieven.

Dit proefschrift bestaat uit vijf hoofdstukken. De eerste twee hoofdstukken plaatsen conceptuele grafen in het perspectief van meerdere disciplines, die betrokken zijn bij kunstmatige intelligentie. Hoofdstuk 1 relateert conceptuele grafen aan historische verschijningen van diagrammen in logica, pictografische talen in kennis-weergave, cognitieve studies van visuele informatie en diagrammen die worden gebruikt bij natuurlijke taal-analyse. Het brede bereik van dit overzicht legt nadruk op de relevantie van fijnmazige studies van visuele aspecten in alle gebieden van kunstmatige intelligentie. Computationale logica kan worden gezien als een gemeenschappelijke grond voor al deze gebieden als ze worden toegepast op automatisch redeneren. Dit is het onderwerp van het volgende hoofdstuk.

Hoofdstuk 2 presenteert het technische raamwerk waarin grafische systemen uit de latere hoofdstukken worden geëvalueerd. Symbolische complexiteitsbegrippen bieden een fijn gestructureerde formele analyse van redeneren met de grafen en verbindt de studie van visueel redeneren met huidige interessen in expressiviteit en complexiteit binnen de symbolische logica. Vervolgens wordt een geografie van complexiteitsresultaten in klassieke en modale fragmenten ontwikkeld. Het terrein voor de studie van conceptuele graaf-talen wordt op deze manier verkend: meerdere logische beslissingsproblemen blijken relevant en op homomorfisme gebaseerde methoden worden gerelateerd aan de probleem-equivalentie tussen model-vergelijking en consequentie die voorkomt in talen met een lage expressiviteit.

Hoofdstuk 3 introduceert het centrale fragment van simpele conceptuele grafen en projectie, een bewijsmethode voor logisch gevolg gebaseerd op gelabelde graaf-homomorfismen. Naast de normale semantiek van simpele conceptuele grafen, die wordt gegeven door een vertaling naar existentiële conjunctieve eerste-orde-logica, wordt ook een theoretisch benaderingsmodel gepresenteerd. Het biedt een direct handvat voor het associëren van projecties met model-vergelijkingen. Door het definiëren van een notie van meta-acycliciteit, gebaseerd op bewaakte kwantificatie en een aangepast projectie-algoritme, wordt een nieuw, in polynomiale tijd beslisbaar fragment van simpele conceptuele grafen aan het licht gebracht (Theorem 3.3.7). Deze taal bevat alle voorheen bekende polynomiale graaf-fragmenten (inclusief simpele conceptuele grafen die kunnen worden veranderd in equivalente bomen).

Hoofdstuk 4 verkent verschillende mogelijke uitbreidingen van de eerdere hoofd-

taal. Allereerst wordt de toevoeging van atomaire negatie onderzocht. In graafrepresentaties definieert een onderscheidingscriterium tussen positieve en negatieve informatie een fragment van simpele grafen met atomaire negatie, waarin projectie nog steeds toepasbaar blijkt (Theorem 4.1.19). Verder is in de bewaakte beperking van dit fragment het consequentie-probleem polynomiaal (Corollary 4.1.22). Ten tweede stellen we een taal van conceptuele grafen voor die equivalent is met eerste-orde-logica, plus een volledige bewijs-methode die tableau-construictieregels en projecties combineert (Chapter 4.2). Tenslotte wordt in het resterende deel van het hoofdstuk een modaal perspectief van het schakelen van grafen bestudeerd. Het gebruik van de notie van bewaakte kwantificatie in haar originele modale raamwerk maakt het mogelijk een taal van 'geneste' grafen met een polynomiale geassocieerde projectie te definiëren (Corollary 4.3.15).

In het laatste hoofdstuk trekken we onze uiteindelijke conclusies uit de complexiteits- en expressiviteits-resultaten, verkregen via de gekozen route door het landschap der conceptuele grafen. Meer in het algemeen suggereert de geconstateerde succesvolle interactie van grafische en symbolische aspecten veelbelovende verdere paden naar meer intrinsiek visueel georiënteerde vormen van berekenbaarheid.

Titles in the ILLC Dissertation Series:

- ILLC DS-1996-01: **Lex Hendriks**
Computations in Propositional Logic
- ILLC DS-1996-02: **Angelo Montanari**
Metric and Layered Temporal Logic for Time Granularity
- ILLC DS-1996-03: **Martin H. van den Berg**
Some Aspects of the Internal Structure of Discourse: the Dynamics of Nominal Anaphora
- ILLC DS-1996-04: **Jeroen Bruggeman**
Formalizing Organizational Ecology
- ILLC DS-1997-01: **Ronald Cramer**
Modular Design of Secure yet Practical Cryptographic Protocols
- ILLC DS-1997-02: **Nataša Rakić**
Common Sense Time and Special Relativity
- ILLC DS-1997-03: **Arthur Nieuwendijk**
On Logic. Inquiries into the Justification of Deduction
- ILLC DS-1997-04: **Atocha Aliseda-Llera**
Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence
- ILLC DS-1997-05: **Harry Stein**
The Fiber and the Fabric: An Inquiry into Wittgenstein's Views on Rule-Following and Linguistic Normativity
- ILLC DS-1997-06: **Leonie Bosveld - de Smet**
On Mass and Plural Quantification. The Case of French 'des'/'du'-NP's.
- ILLC DS-1998-01: **Sebastiaan A. Terwijn**
Computability and Measure
- ILLC DS-1998-02: **Sjoerd D. Zwart**
Approach to the Truth: Verisimilitude and Truthlikeness
- ILLC DS-1998-03: **Peter Grunwald**
The Minimum Description Length Principle and Reasoning under Uncertainty
- ILLC DS-1998-04: **Giovanna d'Agostino**
Modal Logic and Non-Well-Founded Set Theory: Translation, Bisimulation, Interpolation
- ILLC DS-1998-05: **Mehdi Dastani**
Languages of Perception

- ILLC DS-1999-01: **Jelle Gerbrandy**
Bisimulations on Planet Kripke
- ILLC DS-1999-02: **Khalil Sima'an**
Learning efficient disambiguation
- ILLC DS-1999-03: **Jaap Maat**
Philosophical Languages in the Seventeenth Century: Dalgarno, Wilkins, Leibniz
- ILLC DS-1999-04: **Barbara Terhal**
Quantum Algorithms and Quantum Entanglement
- ILLC DS-2000-01: **Renata Wassermann**
Resource Bounded Belief Revision
- ILLC DS-2000-02: **Jaap Kamps**
A Logical Approach to Computational Theory Building (with applications to sociology)
- ILLC DS-2000-03: **Marco Vervoort**
Games, Walks and Grammars: Problems I've Worked On
- ILLC DS-2000-04: **Paul van Ulsen**
E. W. Beth als logicus
- ILLC DS-2000-05: **Carlos Areces**
Logic Engineering. The Case of Description and Hybrid Logics
- ILLC DS-2000-06: **Hans van Ditmarsch**
Knowledge Games
- ILLC DS-2000-07: **Egbert L.J. Fortuin**
Polysemy or monosemy: Interpretation of the imperative and the dative-infinitive construction in Russian
- ILLC DS-2001-01: **Maria Aloni**
Quantification under Conceptual Covers
- ILLC DS-2001-02: **Alexander van den Bosch**
Rationality in Discovery - a study of Logic, Cognition, Computation and Neuropharmacology.
- ILLC DS-2001-03: **Erik de Haas**
Logics For OO Information Systems: a Semantic Study of Object Orientation from a Categorical Substructural Perspective
- ILLC DS-2001-04: **Rosalie Iemhoff**
Provability Logic and Admissible Rules
- ILLC DS-2001-05: **Eva Hoogland**
Definability and Interpolation: Model-theoretic investigations

ILLC DS-2001-06: **Ronald de Wolf**

Quantum Computing and Communication Complexity

ILLC DS-2001-07: **Katsumi Sasaki**

Logics and Provability

ILLC DS-2001-08: **Allard Tamminga**

Belief Dynamics. (Epistemo)logical Investigations

ILLC DS-2001-09: **Gwen Kerdiles**

Saying It with Pictures: a logical landscape of conceptual graphs