

MONOGRAFIES DE L'INSTITUT D'INVESTIGACIÓ  
EN INTEL·LIGÈNCIA ARTIFICIAL  
Number 10



Institut d'Investigació  
en Intel·ligència Artificial

## Monografies de l'Institut d'Investigació en Intel·ligència Artificial

- Num. 1 J. Puyol, *MILORD II: A Language for Knowledge-Based Systems*
- Num. 2 J. Levy, *The Calculus of Refinements, a Formal Specification Model Based on Inclusions*
- Num. 3 Ll. Vila, *On Temporal Representation and Reasoning in Knowledge-Based Systems*
- Num. 4 M. Domingo, *An Expert System Architecture for Identification in Biology*
- Num. 5 E. Armengol, *A Framework for Integrating Learning and Problem Solving*
- Num. 6 J. Ll. Arcos, *The Noos Representation Language*
- Num. 7 J. Larrosa, *Algorithms and Heuristics for Total and Partial Constraint Satisfaction*
- Num. 8 F. Manyà, *Proof Procedures for Multiple-Valued Propositional Logics*
- Num. 9 P. Noriega, *Agent Mediated Auctions: The Fishmarket Metaphor*
- Num. 10 W. M. Schorlemmer, *On Specifying and Reasoning with Special Relations*
- Num. 11 M. López, *Approaches to Map Generation by means of Collaborative Autonomous Robots*

# On Specifying and Reasoning with Special Relations

Wernher Marco Schorlemmer

Foreword by Jaume Agustí and Jordi Levy  
Institut d'Investigació en Intel·ligència Artificial  
Bellaterra, Catalonia, Spain.

Series Editor  
Institut d'Investigació en Intel·ligència Artificial  
Consell Superior d'Investigacions Científiques

Foreword by  
Jaume Agustí and Jordi Levy  
Institut d'Investigació en Intel·ligència Artificial  
Consell Superior d'Investigacions Científiques

Volume Author  
W. Marco Schorlemmer  
Departament de Llenguatges i Sistemes Informàtics  
Universitat Politècnica de Catalunya



Institut d'Investigació  
en Intel·ligència Artificial

ISBN: 84-00-07834-9  
Dep. Legal: B-47670-99  
© 1999 by Wernher Marco Schorlemmer

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.  
**Ordering Information:** Text orders should be addressed to the Library of the IIIA, Institut d'Investigació en Intel·ligència Artificial, Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain.

Printed by CPDA-ETSEIB.  
Avinguda Diagonal, 647.  
08028 Barcelona, Spain.

*Als meus pares*



# Contents

<b>Foreword</b>	<b>xiii</b>
<b>Preface</b>	<b>xv</b>
<b>Abstract</b>	<b>xvii</b>
<b>I Introduction</b>	<b>1</b>
<b>1 Motivation</b>	<b>3</b>
1.1 Beyond Equational Rewriting . . . . .	4
1.2 Special Relations . . . . .	5
1.3 Objectives and Contributions . . . . .	7
1.4 Overview of the Thesis . . . . .	8
<b>2 Theorem Proving with Transitive Relations</b>	<b>11</b>
2.1 Preliminaries . . . . .	12
2.2 Equational Reasoning . . . . .	13
2.3 Reasoning with Transitive Relations . . . . .	15
2.4 Rewrite Techniques in Theorem Proving . . . . .	22
2.5 Putting it into Practice . . . . .	28
2.6 Recent Advances . . . . .	32
<b>II Specification and Proof with Special Relations</b>	<b>33</b>
<b>3 A Logic of Special Relations</b>	<b>35</b>
3.1 Specifying with Special Relations . . . . .	35
3.2 Syntax . . . . .	37
3.3 Semantics . . . . .	40
3.4 Conditional Sentences . . . . .	47
3.5 A Semantic Framework . . . . .	49

<b>4</b>	<b>Rewriting Beyond Equality</b>	<b>51</b>
4.1	Rewriting Along Binary Relations . . . . .	52
4.2	Rewrite Proofs . . . . .	55
4.3	Confluence . . . . .	59
4.4	Critical Atoms . . . . .	62
4.5	Towards Effective Completion . . . . .	68
4.6	Discussion . . . . .	74
4.7	Extensions of Term Rewriting . . . . .	76
<b>III</b>	<b>Computations in Specification Frameworks</b>	<b>81</b>
<b>5</b>	<b>Specification Frameworks with Special Relations</b>	<b>83</b>
5.1	General Logics . . . . .	84
5.2	The Logic of Special Relations . . . . .	86
5.3	Mapping Membership Equational Specifications . . . . .	87
5.4	Mapping Rewriting Specifications . . . . .	95
5.5	Mapping Specifications with Set Relations . . . . .	99
5.6	Conclusion . . . . .	104
<b>6</b>	<b>On Specific Computations</b>	<b>105</b>
6.1	Order-sorted Term Rewriting . . . . .	105
6.2	Bi-rewriting Equational and Horn Logic . . . . .	110
6.3	Rewriting with Set Relations . . . . .	117
6.4	Conclusion . . . . .	120
<b>IV</b>	<b>Reasoning in a Diagrammatic Logic</b>	<b>123</b>
<b>7</b>	<b>Special Relations in a Diagrammatic Logic</b>	<b>125</b>
7.1	A Diagrammatic Horn Logic . . . . .	126
7.2	Spatial and Special Relations . . . . .	130
7.3	Diagram Chaining . . . . .	137
7.4	Diagram Transformation . . . . .	140
7.5	Conclusion . . . . .	144
<b>8</b>	<b>On Diagrammatic Reasoning in Category Theory</b>	<b>147</b>
8.1	Graphs and Categories . . . . .	148
8.2	Commutative Diagrams . . . . .	150
8.3	Reasoning by Diagram Chasing . . . . .	153
8.4	Diagram Chasing vs. Equational Reasoning . . . . .	157
8.5	Computing by Diagram Chasing . . . . .	162



<b>V</b>	<b>Conclusion</b>	<b>167</b>
<b>9</b>	<b>Conclusions and Future Work</b>	<b>169</b>
9.1	Related Research . . . . .	170
9.2	Future Work . . . . .	172
	<b>Bibliography</b>	<b>175</b>



# List of Figures

2.1	Ordered chaining proof calculus . . . . .	25
2.2	Model construction . . . . .	27
3.1	Inference rules coping with equations and type assignment . . . .	36
3.2	Inference rules of the logic of special relations . . . . .	40
4.1	Convergent variable instance atom . . . . .	72
5.1	Inference rules of membership equational logic . . . . .	88
5.2	CONGRUENCE emulation . . . . .	92
5.3	Inference rules of rewriting logic . . . . .	96
6.1	Sequence of ordered chaining inferences . . . . .	116
6.2	Kriauciukas and Walicki's proof calculus . . . . .	118
7.1	Diagrammatic Horn clauses . . . . .	129
7.2	Pattern diagram of recursion . . . . .	129
7.3	From Horn clause to diagram . . . . .	130
7.4	A diagrammatic logic program . . . . .	132
7.5	Spatially captured properties of special relations . . . . .	138
7.6	Diagram chaining . . . . .	139
7.7	Diagrammatic inferences . . . . .	140
8.1	Equational inferences vs. commutative diagrams . . . . .	159
8.2	Category providing types and basic operators . . . . .	162
8.3	Commutative diagrams specifying the behavior of operators . . .	163
8.4	Computation by diagram chasing . . . . .	164



# Foreword

Binary relations like equality, membership, order, or type assignment, play a central role in many executable specification frameworks. Although these relations share some of a set of basic properties, their computational treatment was heterogeneous by means of ad hoc proof calculi based on rewriting with equalities. Maybe this was enough for computational efficiency. However the need was there of a unified understanding of the computational issues raised by these binary relations. This book comes to satisfy this very scientific need of understanding from principles. Precisely the research presented here springs from a reconsideration of term rewriting in its simple principles. Because of this, there is much to learn from this book. It brings a general and rigorous view of the role and limits of term rewriting, aside from the more concrete results which follow from it. That is a very general notion of term rewriting along binary relations and the conditions it has to satisfy in order to provide decision procedures for theoremhood. Then a corresponding logic has been tailored to give appropriate semantics to this calculus, and conservative maps of logics have been used to highlight the pragmatics of binary relations in the concrete specification frameworks that motivated the research. Furthermore the generality of the approach allowed the author to make a foray into the terrain of diagrammatic reasoning in the last part of the book. It gives the flavor of a promising area of investigation. Definitely we believe this is a book rich in potentialities for new research.

Bellaterra, November 1999

Jaume Agustí and Jordi Levy  
Institut d'Investigació en Intel·ligència Artificial  
Consell Superior d'Investigacions Científiques  
`{agusti,levy}@iia.csic.es`



# Preface

El meu primer agraïment és per a en Jaume Agustí. Des del primer dia que em va acceptar com a doctorand seu, quan l'IIIA encara estava al CEAB, i fins al darrer moment de l'elaboració d'aquesta tesi, he pogut comptar sempre amb el seu suport, ajuda i inspiració; i no només ha sabut formar-me com a científic, sinó que també m'ha moldejat com a persona.

També estic enormement agraït a en Jordi Levy, que acceptà codirigir la meva tesi. He tingut la sort de poder aprofitar els seus coneixements sobre molts dels temes que componen aquesta tesi, ja que en certa manera m'he atrevit a continuar una línia de recerca que ell i en Jaume Agustí van encetar anys enrere.

Voldria també donar les gràcies a en Jordi Puigsegur. Els treballs conjunts que vam fer sobre la semàntica operacional del seu llenguatge lògic visual han estat molt fructífers i han enriquit la meva tesi amb un parell de capítols dels quals estic molt satisfet.

José Meseguer i Narciso Martí-Oliet també han contribuït de manera significativa en l'enfocament d'aquesta tesi, i mentre jo visitava el Computer Science Laboratory a SRI International sempre han estat disponibles quan em sorgien milers de dubtes 'categòrics'. Junt amb Francisco Duran, Manuel Clavel i Peter Ōlvezcky han fet del meu pas per SRI International una estada inoblidable en aquesta etapa de doctorand.

Moltes altres persones han anat contribuint amb el seu gra de sorra a edificar aquesta tesi, sigui per algun comentari que m'han fet, per alguna xerrada que hem tingut, perquè van revisar alguns dels meus treballs, o perquè m'han tutelat la tesi. Gràcies, doncs, Robert Nieuwenhuis, Dave Robertson, Hendrik Decker, Gabriel Valiente i Maria Lluïsa Bonet. Estic especialment agraït a Fernando Orejas, Mario Rodríguez Artalejo, Donald Sannella, Francesc Esteva i Albert Rubio per haver acceptat formar el tribunal de tesi, i pels seus comentaris que han fet possible incorporar millores en aquesta monografia.

L'entorn professional i humà que m'ha proporcionat l'IIIA, tant quan estava a Blanes, com després a Bellaterra, també han estat molt positius per dur a terme la meva recerca, i sempre he trobat totes les portes obertes per preguntar dubtes a qualsevol dels *iiiaencs* o *iiiaenques*. Gràcies a tots. I gràcies, també, per publicar aquesta tesi dins de la sèrie de monografies de l'IIIA.

No puc oblidar, tampoc, els nombrosos amics i familiars que han hagut de patir les meves penes mentre feia la tesi, sobretot en els mesos finals. Ells també

han hagut de carregar amb el pes de moltes responsabilitats que he anat deixant penjades. Agraïxo la comprensió i l'ànim rebuts per part de tothom; espero poder recompensar-los i reprendre la feina que he deixat a mitges.

Però tot el meu agraïment final va dirigit especialment als meus pares. Sense el seu suport moral i la seva completa i sincera disponibilitat no sé pas com hagués pogut arribar fins al final. Gràcies.

El treball contingut en aquesta memòria ha estat realitzat a l'Institut d'Investigació en Intel·ligència Artificial (IIIA) del Consell Superior d'Investigacions Científiques (CSIC) amb el finançament obtingut dels projectes DISCOR (TIC 94-0847-C02-01) i MODELOGOS (TIC 97-0579-C02-01) de la CICYT, d'una beca predoctoral del CSIC, i d'una beca per a la recerca a l'estranger de la Direcció General de Recerca de la Generalitat de Catalunya.

Barcelona, novembre de 1999

W. Marco Schorlemmer  
Departament de Llenguatges i Sistemes Informàtics  
Universitat Politècnica de Catalunya  
`marco@lsi.upc.es`



# Abstract

Motivated by the results obtained from the generalization of term rewriting techniques to inclusion theories put forth by Levy and Agustí in their *calculus of refinements*, we further investigate the use of these techniques for a variety of different specification frameworks, such as membership equational logic, rewriting logic, and non-deterministic specifications with set relations. All these frameworks use several binary relations —we call them *special relations*— in their axioms, and since we believe that many of these special relations have properties that can be exploited computationally by term rewriting, we extend the term rewriting technique to reason simultaneously with multiple special relations.

We thus tailor a logic endowed with an abstract semantics based on the category theory of relations that acts as a general framework of specifications with special relations; through maps of logics we describe particular specification frameworks as instances of our logic. We believe that these maps highlight the pragmatics of special relations in these specification frameworks.

We extend the term rewriting technique to multiple special relations and very general monotonicity and antimonotonicity properties of function symbols, and squeeze out all the computational power of this technique, by exploring its fitness for effective use. Thus we exploit the notion of *polarity* in order to tackle the tractability problems arising from our general approach to term rewriting, and introduce notions such as *well-polarized signature* and *polarization relation of a signature*.

The application of our general view of term rewriting to particular specification frameworks leads us to unify notions that up to now have been considered separately, namely confluence and sort-decreasingness, and we further revise previous work on the generalization of term rewriting techniques for non-deterministic specifications.

We also formalize the relationship existing between a visual formalism put forth by Agustí et al. and the general framework we are proposing, and present a completely visual inference mechanism inspired on proof techniques developed in this thesis. This last research line led us to explore diagrammatic reasoning in the more general framework provided by category theory, thus presenting a formalization of the diagram chasing technique with its relation to standard equational reasoning, which further highlights the intuitiveness of graphical representations and their fitness to conduct inferences with them.



## Part I

# Introduction



# Chapter 1

## Motivation

It is safe to say that equational logic lies in the core of the algebraic specification discipline for formal program development. At least this is true if one is concerned with property-oriented specification languages, for which executability is their primary goal (Ehrig and Mahr, 1985; Wirsing, 1990). The fundamental idea of this discipline is to model programs as algebras by first fixing a signature and then specifying the intended properties of a program with equational axioms. The success of this approach relies, on one hand, on its intuitiveness and mathematical simplicity, and on the other hand, on the powerful reasoning tool it provides: the familiar technique of replacing equals by equals captured by the term rewriting technique. Specifications based on equational theories are thus directly executable, acting as a first prototype, and allowing one to validate its conformance to the initial requirements. By successive refinements of such executable specifications, one eventually derives efficient and with respect to the initial specification correct implementations, written in a modern high-level programming language, which usually is a functional programming language with rich data structures (Sannella and Tarlecki, 1997; Bidoit et al., 1991).

Of course, pure equational theories are by far not expressive enough to cover many of the features of modern programming languages, like e.g., typing, polymorphism, non-determinism, partiality, higher-order functions, object-orientation, and concurrency. Thus more expressive logics have been pursued, but the vast majority of the approaches rely on equations as the core predicate to build specifications, and they take more or less sophisticated variants of equational term rewriting as its computational paradigm.

Let us now emphasize a couple of paradigm shifts that happened during the last decade. At first sight they may seem to be quite unrelated, but its relationship happens to be of central importance for the better understanding of the research accomplished during the development of this thesis. Let us also stress that this emphasis is in no way intended to give these ‘shifts’ any special importance with respect to other aspects of the algebraic specification discipline; they have special importance for grasping our research aims.

On one hand, there was more or less a shift from a purely syntactic treat-

ment of sorts towards a semantic treatment of them, moving information on sort classification explicitly into the axioms of a specification, thus gaining on expressiveness (Mosses, 1993). Some examples in this direction are classified algebras (Wadge, 1982), unified algebras (Mosses, 1989), type algebras (Manca et al., 1990), galactic algebras (Mégrelis, 1992), and more recently, membership algebras (Meseguer, 1998). In particular, within this tendency lies also Levy and Agustí's *calculus of refinements* (COR) (Levy, 1995), a formal specification system based on inclusions, suitable for the preliminary specification of complex systems and their successive refinements (Robertson et al., 1994).

On the other hand, there was a generalization of the term rewriting technique to handle other transitive relations in addition to equality. Actually, Levy and Agustí were the first to do this generalization, by using term rewriting systems to reason within logical theories involving inclusions (Levy and Agustí, 1993). This gave rise to the so called *bi-rewrite systems* (Levy and Agustí, 1996), which underlie the operational semantics of COR, and which later on served for generalizing paramodulation and superposition calculi to first-order theories with arbitrary transitive relations (Bachmair and Ganzinger, 1994c; Bachmair and Ganzinger, 1994a). Independently, Meseguer showed that the implicit logic underlying rewrite systems was not equational logic, as advocated by Huet (1980), but *rewriting logic*, and he thoroughly studied and developed its mathematical semantics (Meseguer, 1992).

Inspired by these two trends in algebraic specification, we want to establish a common ground for both, motivated by the following observations.

## 1.1 Beyond Equational Rewriting

Several positive results proved the potential success of the bi-rewriting technique for automated reasoning. Levy and Agustí proposed implementations of non-deterministic specifications with bi-rewrite systems (Levy and Agustí, 1992), and their approach inspired the subsequent work done by Kriaučiukas and Walicki applying rewriting to non-deterministic specifications with set relations (Kriaučiukas and Walicki, 1995; Kriaučiukas and Walicki, 1996). But probably the word problem for free lattices is the most significant example of the computational gain obtained by using bi-rewrite systems. Levy gave a decision algorithm based on bi-rewriting (Levy, 1995), though no standard associative-commutative term rewriting system deciding the theory of free lattices exists (Freese et al., 1993). This was formally proved by Struth (1997).

Despite of this gain, the above mentioned frameworks (type algebras, galactic algebras, membership algebras) still base their respective proof calculi on standard equational rewriting, though they extend the expressiveness of many-sorted conditional equational logic by using additional binary relations besides equality in their axioms. Even though dealing with several binary relations at once, not only with equality, they do not extend term rewriting to these other binary relations in a uniform way, but embed an *ad hoc* and heterogeneous treatment of these relations within term rewriting.

For instance, Manca, Salibra, and Scollo say that equational type logic “can be viewed as Horn clausal logic with equality and one (binary) predicate, viz. type assignment” (Manca et al., 1990). But their *equational type rewrite systems* make a clear distinction between standard rewrite rules and so called *type assignment rules*. They generalize confluence results given by Bergstra and Klop (1986) for conditional term rewriting systems, taking into account this distinction. But since they do not give any results on termination, completion, and on how a decision procedure for type checking would look like, we are not able to appreciate to what extent their generalization suffices for the purposes of computation. Such as with equational type rewrite systems, the recently developed *conditional membership/rewriting systems* for membership equational logic make also a clear distinction between standard rewrite rules and type assignment rules (called *membership rules*). But unlike Manca, Salibra, and Scollo’s formalism, Bouhoula, Jouannaud, and Meseguer do present *Knuth-Bendix*-like completion methods, and suggest powerful proof techniques (Bouhoula et al., 1997a). Even so, we believe that the distinction between standard rewrite rules, type assignment rules, and membership rules may hinder further development of these techniques, because they still embed a heterogeneous treatment of binary relations.

Without devaluing the interest and success of previous approaches—they aim at providing efficient rewriting-based proof calculi by focusing on very specific proof techniques—our purpose is to explore how the recent results on generalizing term rewriting techniques to other binary relations in addition to equality translate to all these frameworks. Though such generalization, of course, adds an additional degree of technical complexity, we believe it sheds light on the role term rewriting plays within these specification frameworks.

## 1.2 Special Relations

We will call the binary relations, with which the above mentioned specification frameworks construct specifications, *special relations*, because they have certain properties that we claim can be exploited computationally by term rewriting. We are thinking of properties such as reflexivity, symmetry, antisymmetry, monotonicity, antimonotonicity, replacement, transitivity, and compositionality.

Let us see what we mean by a *special relation*. Consider, for example, the following (quite trivial) specification in equational type logic:

**spec NAT is**  
 $0 : nat$   
 $nat = posint$   
 $id(0) = 0$   
**endspec**

Symbols  $0$ ,  $nat$ , and  $posint$  are constants, and  $id$  stands for a unary function.

As already mentioned, such a specification is a Horn theory<sup>1</sup> with two differ-

---

<sup>1</sup>In this particular example no conditional axioms are given; but they could be.

ent predicates, actually binary relations, namely equality ( $=$ ) and type assignment ( $:$ ). Equality is a transitive binary relation for which also the replacement property holds, and that is the reason why term rewriting captures it so well. But in addition, the following two axioms hold, for all  $x, y$ , and  $z$ :

$$\begin{array}{llll} x : y & \text{and} & y = z & \text{imply} & x : z \\ x = y & \text{and} & y : z & \text{imply} & x : z \end{array}$$

These axioms are expressed in a much more compact way as relation-algebra expressions:

$$; = \sqsubseteq : \quad (1.1)$$

$$= ; : \sqsubseteq : \quad (1.2)$$

The symbol  $;$  denotes composition of relations, and  $\sqsubseteq$  a partial order over relations capturing implication. Transitivity of equality can also be expressed this way:

$$= ; = \sqsubseteq =$$

Term rewriting works so well for equality, because it captures this composition of equality with itself in a natural way. Thus it is reasonable to think that it could also capture other slightly more general compositions of binary relations. Let us look at the specification above as the following term rewriting system:

$$\begin{array}{lll} 0 & \rightarrow & nat \\ & \vdots & \\ nat & \rightarrow & posint \\ & = & \\ id(0) & \rightarrow & 0 \\ & = & \end{array}$$

With these generalized rewrite rules, and taking into account the compositional relationship existing between  $=$  and  $:$  given in axioms (1.1) and (1.2), we can compose rewrite steps as follows:

$$\begin{array}{c} id(0) \xrightarrow{=} 0 \xrightarrow{:} nat \xrightarrow{=} posint \\ \underbrace{\qquad\qquad\qquad}_{\xrightarrow{:}} \\ \underbrace{\qquad\qquad\qquad}_{\xrightarrow{:}} \end{array}$$

In general, we could compose rewrite steps whenever relation-algebra axioms of the form  $\alpha; \beta \sqsubseteq \gamma$  hold, where  $\alpha, \beta$ , and  $\gamma$  denote arbitrary binary relations.

Term rewriting also well captures another property of equality, the replacement property, stated in the following axiom: For every function symbol  $f$  and for every argument position  $i = 1 \dots n$ , and for all  $x$  and  $y$ ,

$$x = y \text{ implies } f(\dots, \overset{i)}{x}, \dots) = f(\dots, \overset{i)}{y}, \dots) .$$



In general, we may have that for a specific function symbol  $f$  and some argument position  $i \in [1 \dots n]$ ,

$$x \alpha y \text{ implies } f(\dots, \overset{i}{x}, \dots) \beta f(\dots, \overset{i}{y}, \dots) ,$$

where now  $\alpha$  and  $\beta$  denote arbitrary binary relations. We now can generalize term rewriting to allow replacements of terms only on certain subterm positions, according to such *monotonicity properties* of function symbols. We will use the notion of *polarity* to capture these properties, inspired by Manna and Waldinger's work on defining paramodulation-like inference rules involving special relations (Manna and Waldinger, 1986; Manna and Waldinger, 1992).

### 1.3 Objectives and Contributions

We believe that a suitable notion of term rewriting —we call it *term rewriting along binary relations*— nicely captures many basic properties of special relations, and that it provides a natural way to study some computational issues concerning these special relations. We claim that this notion actually constitutes the *bare bones* of term rewriting, namely

1. *replacement* of terms, applying one rewrite rule;
2. application of replacements on *certain subterm positions* of terms; and
3. successive *composition* of replacements.

Because of these observations, it is necessary

1. to identify the general framework in which we are making all these assumptions: This leads us to look at equational type logic, membership equational logic, and other extensions of many-sorted equational logic as particular instances of a logic of special relations. We believe such logic better grasps the pragmatics of special relations in specification and reasoning.
2. to squeeze out all the computational power of our general view of term rewriting by studying how far such a generalization has actually sense for practical use: For this purpose we play with the notion of polarity in order to tackle the tractability problems arising from such general perspective of term rewriting. We are able to determine the cases in which we may obtain effective reasoning mechanisms based on term rewriting, and also to identify what specific role the basic properties of special relations play in the tractability of those mechanisms.
3. to ensure that the proposed general framework indeed suites well for studying specification paradigms that go beyond equational logic in the sense explained above: The technique defining maps between suitable abstract axiomatizations of logics proves to be useful for this purpose. These maps

also highlight the relationship existing between several special relations and their interplay within a particular specification framework.

4. to show how our technique applies within these specific frameworks: We bring together, under one unique notion of local confluence, several decidability conditions that have previously treated separately. We also revise previous work on the generalization of term rewriting for non-deterministic specifications, putting special emphasis on the subtleties of such generalization.

## Visual Formalisms

Taking the fundamental idea of specifying with inclusions coming from the previous work done by Levy and Agustí in defining COR, Agustí, Robertson, and Puigsegur defined the graphical specification language GRASP (Agustí et al., 1995), with the intention of making the use of formal specification languages more accessible to non-logicians. It comes naturally to explore the relationship existing between this visual formalism and the general framework we are proposing, since they are inspired both by the same previous work done on COR. In particular we identify chaining-based inference rules for GRASP, and lay them down in a completely diagrammatic way. Together with Puigsegur and Agustí we further explore a visual operational semantics based on the inference mechanism we present in this thesis. This operational semantics is inspired on resolution-based theorem proving by transformation of diagrams. This joint work motivates us to further explore diagrammatic reasoning in the more general framework provided by category theory. The result is a formal definition of diagram chasing in category theory within the context laid down by the diagrammatic reasoning community; and also a formal presentation of its relation to standard equational reasoning, by means of a map of entailment systems. We also hint at some possible application in visual declarative programming.

## 1.4 Overview of the Thesis

The thesis is structured into five parts:

**Part I:** After this motivating introduction of chapter 1, where we summarize the starting points that stimulated the research presented in this thesis, we provide the reader with some historical background, basic notions, and terminology for the better understanding of our main results.

Thus we survey, in chapter 2, related work and alternative approaches followed by those researches that preceded us in the use of term rewriting techniques for equational reasoning and for reasoning with transitive relations. We put special emphasis on the breakthrough that the bi-rewriting technique meant for automated deduction in first-order theories with transitive relations, and discuss some of the key aspects to have in mind when putting such technique into practice.

**Part II:** Next, we formally establish our general framework. In chapter 3, we define the syntax and semantics of a logic of special relations that attempts to capture the pragmatics of special relations in several specification paradigms. In general, it is a Horn logic with predicates restricted to binary relations, but mainly we focus on the fragment consisting of positive unit clauses. The basic properties of these binary relations are built into the rules of deduction by endowing the syntax of the logic with a relation-algebra structure and with a notion of polarity. In a first approach we construct intuitive set-theoretic models of theories in this logic, but in order to provide a very abstract framework that indeed captures many disparate specification frameworks, we move towards a model theory based on the category theory of relations, and thus construct models as a certain kind of allegories.

In chapter 4, we then thoroughly present and study a general notion of term rewriting along binary relations. We redefine the well-known notions of rewrite relation, rewrite proof, termination, confluence, critical peaks, and completion for this generalization, and show the main advantages and disadvantages of our perspective of term rewriting. For this reason we explore the important role played by polarity, and use this notion for taming the framework and explaining the subtleties that arise in the generalization. We thus introduce the notions of ‘well-polarized and well-commuting signature’ and ‘polarization relation of a signature’, which turned out to be very useful for revising previous work on the generalization of the term rewriting technique.

**Part III:** Through maps of logics, we formally show, in chapter 5, how our logic of special relations suites well for describing several specification frameworks. We choose membership equational logic, rewriting logic, and specifications with set relations to explain how these maps translate these frameworks into our logic, thus building into the deduction mechanism the properties of special relations by means of relation-algebra expressions. We believe these maps illustrate the role played by special relations within these frameworks.

In chapter 6, we further study some computational issues of these three specific instances of our logic from the perspective of term rewriting along binary relations. We are able to show how our general notion of term rewriting actually unifies separate decidability properties of specific theories, such as the notions of confluence and sort-decreasingness in order-sorted term rewriting. Our framework also provides a nice way to show that sort-decreasingness is actually too strong a notion of decidability in membership equational specifications. We also discuss the role of the bi-rewriting technique for theories in rewriting logic and its relationship to ordered chaining inferences; and as already mentioned, we make use of the notion of well-polarization introduced before to revise previous work on generalizing term rewriting techniques for non-deterministic specifications

with set relations.

**Part IV:** In the chapters forming this part, we make a foray into the terrain of diagrammatic reasoning. We show, in chapter 7, how our logic is general enough as to capture a novel diagrammatic logic, which our research group currently develops for high-level specification and programming. By realizing that such diagrammatic logic is based on visual clues that exploit the spatial relationship of box containment, we explore the role of this spatial relation as a special relation in the deduction process. Therefore we provide the foundations for a completely visual chaining-based inference mechanism, which directly exploits the intrinsic properties of the visual containment relation, such as transitivity of box containments.

In chapter 8, we go a little bit further and discuss some aspects of diagrammatic reasoning in category theory. We suggest a formal definition of diagram chaining by axiomatizing it as an entailment system in a general logic, and then show how we can establish a map of entailment systems between diagram chasing and standard equational reasoning. This map highlights the expressive power of diagram chasing in order to capture specific properties of the special relation ‘equality’. Thus we briefly discuss how we may obtain new formal tools for diagrammatic reasoning and computation out of the visual formalisms provided by category theory.

**Part V:** We conclude, in chapter 9, summarizing our main contributions. We also briefly survey other related work that shares similar objectives or methodology, either in its model-theoretic aspect, or in its proof-theoretical aspect, and finally suggest some interesting future research lines opened by the results presented in this thesis.

## Chapter 2

# Theorem Proving with Transitive Relations

Rewrite techniques have been typically applied to reason with the equality relation and have turned out to be among the most successful approaches to equational theorem proving. They implicitly capture the transitivity and congruence properties of the equality relation in a natural way and avoid the explicit use of the equality axioms, which pose severe problems in the design of efficient automated theorem provers. An effort has also been made to apply these techniques to theorem proving in first-order logic with equality.

But since rewrite rules rewrite terms in one direction, it is not only in reasoning with the equality relation where these techniques naturally apply, but in reasoning with arbitrary, probably *non-symmetric*, transitive relations. The equality relation is just a special case of monotone transitive relation that is also symmetric. Therefore, the generalization of rewrite systems is not to rewrite on equivalence classes of terms, as presented by Huet (1980), but to consider rewrite systems as a logic itself, as pointed out by Meseguer (1992), or to consider them as a deduction mechanism for theories with non-symmetric relations, the so called bi-rewrite systems of Levy and Agustí (1993; 1996). Bachmair and Ganzinger extended bi-rewrite systems to the more general ordered chaining calculus in first-order logic with transitive relations (Bachmair and Ganzinger, 1994c). The work done so far in applying rewrite techniques to arbitrary transitive relations showed several important differences with the equational case, because of the lack of symmetry, which hinders notably the deduction mechanism in theories with this kind of relations. New problems appear, which must be solved in a quite different way.

In this chapter, we survey the research accomplished by those researchers that went before us in using term rewriting techniques for reasoning with transitive relations. We will point to the reasons why an efficient treatment of this generalization from symmetric to non-symmetric transitive relations is difficult to achieve, which reveals that though symmetry is not a property captured by term

rewriting, it plays a key role for the practicability of this technique; therefore, we present reasoning with the equality relation as a special case. This survey will provide the necessary background for us to move on, in subsequent chapters, to generalize these techniques for reasoning with arbitrary binary relations.

After first introducing some preliminary notions used in this chapter, and also in chapter 4, we give a short historical sketch of the use of term rewriting techniques in automated theorem proving for reasoning with equality. Next, we give an overview of the application of these techniques to reason with arbitrary transitive relations, and how this was generalized to theorem proving for full first-order clauses with transitive relations. The description of important efficiency criteria and the discussion of their use in the **Saturate** theorem prover, implemented by Nivela and Nieuwenhuis (1993) and further developed by Ganzinger (Ganzinger et al., 1995), allows us to determine the requirements for efficient implementations of these techniques, and to expose the drawbacks that appear when trying to adapt these efficiency criteria to arbitrary transitive relations. Finally, we conclude with some of the recent advances in non-symmetric term rewriting that followed the work summarized in this chapter.

## 2.1 Preliminaries

**2.1.1 Terms and subterms.** Unless stated explicitly, we follow the notation and the standard definitions used in (Dershowitz and Jouannaud, 1990). We are concerned with first-order terms  $\mathcal{T}_\Sigma(\mathcal{X})$  over a nonempty signature  $\Sigma$  of function symbols and a denumerable set  $\mathcal{X}$  of variables. Given a term  $t$ , let  $t|_p$  denote the *subterm* that *occurs at position*  $p$ . When this occurrence is replaced with term  $s$ , we denote that by  $t[s]_p$ . We refer to  $t|_p$  as the *context* in which the replacement takes place. Sometimes we will drop the subscript  $p$  if the position is not significant. If in a term  $t$ , subterm  $t|_q$  is also subterm of subterm  $t|_p$ , we write  $p \leq q$ . Thus, if  $p \not\leq q$  and  $q \not\leq p$  then neither of both terms is subterm of the other.

**2.1.2 Literals and clauses.** We call a pair of terms  $t, t'$  related by  $\approx$  an *equation*, and write  $t \approx t'$ . We call it an *inequation* or an *inclusion*, if its terms are related by  $<$  or  $\subseteq$ , respectively. Equations, inequations, and inclusions are also referred to as *atoms*. Atoms and their negations are called *literals*, and the disjunction of literals are called *clauses*. We use capital letters, such as  $C$  or  $D$ , to denote clauses, and use the comma to denote disjunction.

**2.1.3 Substitutions and unifiers.** A *substitution*  $\sigma = \langle x_1 \mapsto t_1, \dots, x_n \mapsto t_n \rangle$  is a map from a finite set  $\{x_1, \dots, x_n\} \subseteq \mathcal{X}$  of variables to  $\mathcal{T}_\Sigma(\mathcal{X})$ , and can be extended as a morphism to a map from terms to terms, from literals to literals, and from clauses to clauses. A *unifier* of two terms  $s$  and  $t$  is a substitution  $\sigma$  such that  $\sigma(s) = \sigma(t)$ . It is a *most general unifier (mgu)*, if for every other unifier  $\sigma'$  there exists a substitution  $\tau$  such that  $\sigma' = \sigma \cdot \tau$ .

**2.1.4 Term orderings.** An irreflexive and transitive binary relation  $\succ$  over a set  $T$  is called an *ordering*. In the specific case of orderings on terms (i.e.,  $T = \mathcal{T}_\Sigma(\mathcal{X})$ ), it is a *reduction ordering* if it is also well-founded (i.e., there are no infinite sequences  $t_1 \succ t_2 \succ \dots$  of elements of  $T$ ), monotonic (i.e.,  $s \succ t$  implies  $u[s]_p \succ u[t]_p$ ), and stable under substitutions (i.e.,  $s \succ t$  implies  $\sigma(s) \succ \sigma(t)$ ). *Path orderings* define reduction orderings over a given well-founded ordering on the symbols of the signature —the *precedence*— depending on the paths of the tree structure of terms. Actually they define *simplification orderings*, i.e., reduction orderings that satisfy the subterm property  $t \succ t|_p$ , with  $p \neq \lambda^1$ . An example of path ordering is the *lexicographic path ordering*.

An ordering on terms can be extended to an ordering on literals and on clauses, by considering literals as multisets of terms, and clauses as multisets of literals, and taking multiset extensions of the term ordering. A literal  $l$  is said to be *maximal* with respect to a clause  $C$  if, for all  $l'$  in  $C$ ,  $l \succcurlyeq l'$ , and it is said to be *strictly maximal* with respect to  $C$  if, for all  $l'$  in  $C$ ,  $l \succ l'$ . See (Dershowitz, 1987) for an extended survey on orderings.

**2.1.5 Rewrite relation.** A binary relation  $\rightarrow$  is called a *rewrite relation* if  $s \rightarrow t$  implies  $u[\sigma(s)]_p \rightarrow u[\sigma(t)]_p$ , for all terms  $s, t$ , and  $u$ , subterm position  $p$ , and substitution  $\sigma$ . We denote with  $\xrightarrow{+}$  the transitive closure and with  $\xrightarrow{*}$  the reflexive-transitive closure of rewrite relation  $\rightarrow$ . If the rewrite relation is also symmetric then we will denote it with  $\longleftrightarrow$ , and with  $\xleftrightarrow{+}$  and  $\xleftrightarrow{*}$  its transitive and its reflexive-transitive closure, respectively.

## 2.2 Equational Reasoning

**2.2.1 Proving by rewriting.** Most of the effort made in order to deal efficiently with special relations in the context of resolution-based theorem proving has focused mainly on the equality relation, because of its importance in a large number of domains. The use of rewrite techniques in resolution-based theorem proving appears to be one of the most promising approaches developed so far.

The use of rewrite techniques in automated theorem proving has its origins in the completion procedure stated by Knuth and Bendix (1970) for solving the word problem in equational theories. Equations are used as one-way rewrite rules, namely in the direction in which they simplify terms. The completion process adds to the initial set of equations  $E$  sufficiently many new ones, by computing *critical pairs*, in order to obtain a *convergent* (i.e., terminating and Church-Rosser) rewrite system. Convergent rewrite systems, when they can be obtained, provide a decision algorithm for the word problem in  $E$ , based on the computation of normal forms, since they exist and are unique in each equivalence class of the congruence defined by the set of equations  $E$ . If a set of equations  $E$  can be completed into a finite and convergent rewrite system  $R$ , then this system  $R$  is actually a finite encoding of the whole congruence closure defined

---

<sup>1</sup>Position  $\lambda$  is the root position.

by  $E$ , which is infinite in general: all possible proofs of equational consequences of  $E$  can be represented as *rewrite proofs*. Furthermore, if rewrite rules are kept as interreduced as possible, having in  $R$  the minimal number of rewrite rules required to represent the congruence closure, then the rewrite system  $R$  is *canonical*, and therefore unique up to renaming of variables (Dershowitz et al., 1988).

But since the word problem in equational theories is not decidable in general, a convergent rewrite system can not always be obtained. On one hand, the completion procedure may fail, because of the impossibility to orient an equation as a simplifying rewrite rule. On the other hand, it may not terminate. Failure can be avoided by *ordered* (or *unfailing*) completion (Lankford, 1975; Hsiang and Rusinowitch, 1987; Bachmair et al., 1989). Because of the ‘unfailing’ property of ordered completion, we can see it as a refutational deduction process. During this process we may distinguish two main kinds of deductions: those based on the generation of new equations by computation of critical pairs, which participate in the process of refutation, and those that simplify equations, which therefore avoid the deduction of unnecessary rules for the purpose of refutation. This distinction between productive deduction and simplifying deduction is a key aspect in theorem proving and hence also in its rewrite approach.

**2.2.2 Paramodulation.** The effort to extend completion to general first-order clauses has introduced the use of well-founded orderings to resolution-based theorem proving. In the theorem proving context, the desire to encode the equality predicate into the logical language led to the *paramodulation* inference rule stated by Robinson and Wos (1969):

$$\text{Paramodulation: } \frac{C, s \approx t \quad D, u[s']_p \approx v}{\sigma(C), \sigma(D), \sigma(u[t]_p) \approx \sigma(v)}$$

where  $\sigma$  is a most general unifier of  $s$  and  $s'$ . Paramodulation, together with resolution and factoring, was proved to be complete, whenever the reflexivity and functional-reflexivity axioms were added to the initial set of clauses. The paramodulation rule was significantly improved by Brand (1975) and Peterson (1983) by proving that the functional-reflexivity axioms were unnecessary, and that paramodulation through variables was not required. But even under these restrictions, paramodulation is difficult to control, because it generates the complete congruence closure of equality. Since convergent rewrite systems avoid the generation of such congruence closure, researchers focused on extending rewrite techniques to paramodulation.

**2.2.3 Superposition.** The generalization of completion to full first-order clauses led to the *superposition calculus* (Bachmair and Ganzinger, 1990; Bachmair and Ganzinger, 1994b), which, in essence, is a generalization of the critical pair generation of the Knuth-Bendix completion procedure. But this extension required the development of more powerful techniques for the completeness proofs. Hsiang and Rusinowitch (1991) proved the refutational completeness of



ordered paramodulations using the *transfinite semantic tree* method. Bachmair (1991) used *proof orderings* in order to obtain similar results. But it was with the *model construction* method of Bachmair and Ganzinger when the refutational completeness of an inference system based on strict superposition was proved (Bachmair and Ganzinger, 1990; Bachmair and Ganzinger, 1994b). At the same time, the model construction method provides a powerful, abstract redundancy notion, which generalizes the simplification methods of Knuth-Bendix completion. Model construction also allowed one to prove refutational completeness of inference calculi based on strict superposition with more restrictions, like the use of basic strategies and ordering constrained clauses (Nieuwenhuis and Rubio, 1995; Bachmair et al., 1995), which reduces the search space to be explored. The productive deduction of non-redundant clauses based on superposition is known as *saturation*, and it generalizes, in some sense, the process of completion towards canonical rewrite systems of equational theories.

## 2.3 Reasoning with Transitive Relations

**2.3.1 Chaining.** Most of the research done so far focused mainly on the equality relation, but there have been also some attempts to include transitive relations other than equality into the logical language. Slagle (1972) was the first to propose an inference system for theories with equality, orderings, and sets, based on the *chaining* inference rule, which essentially is the equivalent of the paramodulation rule for arbitrary transitive relations:

$$\text{Chaining: } \frac{C, u < s \quad D, t < v}{\sigma(C), \sigma(D), \sigma(u) < \sigma(v)}$$

where  $\sigma$  is a most general unifier of  $s$  and  $t$ .

But completeness of such inference systems requires the functional-reflexive axioms and chaining through variables. By investigating special transitive relations, such as dense total orderings without endpoints, Bledsoe and Hines (1980) developed techniques for eliminating certain occurrences of variables from formulae, avoiding the prolific chaining through variables. Bledsoe, Kunen, and Shostak (1985), and also Hines (1992), achieved completeness results for particular such systems of restricted chaining. Manna and Waldinger proposed subterm chaining methods for general clauses in the presence of monotonicity and antimonotonicity (Manna and Waldinger, 1986), but their calculus was proved to be incomplete (Manna and Waldinger, 1992).

But, like ordering restrictions on the paramodulation rule led to the superposition calculus, which avoids the generation of the whole congruence closure of equality, ordering restrictions on the chaining inference rule avoid the generation of such transitive closure, too. Therefore, rewrite systems can be used in order to provide decision procedures for the word problem in some theories with arbitrary transitive relations. This was done for the first time by Levy and Agustí (1993), who developed *bi-rewrite systems* (Levy and Agustí, 1996). Bachmair and Ganzinger later generalized bi-rewrite systems to theorem proving with full

first-order clauses with arbitrary transitive relations. They introduced the *ordered chaining calculus*, which generalizes superposition to cope with arbitrary transitive relations (1994c).

**2.3.2 Bi-rewriting.** Let us now see Levy and Agustí's bi-rewriting technique in more detail, because of its importance for the research accomplished in this thesis. Let  $\subseteq$  denote a reflexive, monotonic, but non-symmetric transitive relation, which we will refer to as *inclusion*. Given a finite set of inclusions  $I$ , we say that  $s \xrightarrow[\subseteq]{} t$  if there exist an inclusion  $u \subseteq v \in I$  and a substitution  $\sigma$ , such that  $\sigma(u)$  is a subterm of  $s$ , and  $t$  is the term resulting from replacing  $\sigma(u)$  with  $\sigma(v)$  in term  $s$ , or vice versa. Intuitively, we say that  $s \xrightarrow[\subseteq]{}^* t$  if we are able to obtain term  $t$  from  $s$  by replacing terms with 'bigger' terms, using instances of inclusions in  $I$ . The same can be done in the reverse direction, from 'bigger' to 'smaller' terms<sup>2</sup>. Given two terms  $s$  and  $t$ , the *word problem* in a theory  $I$  is to decide whether  $I \models s \subseteq t$ . By the equivalent of Birkhoff's theorem for inclusions it is known that  $I \models s \subseteq t$  if and only if  $s \xrightarrow[\subseteq]{}^* t$ , and in this case we say that the inclusion  $s \subseteq t$  is *provable* in  $I$ . But it is an arduous task to automatically prove the validity of an inclusion  $s \subseteq t$  by application of all possible replacements on subterms of  $s$  until term  $t$  is obtained. It is impractical to build a decision procedure for decidable inclusion theories based on this methodology, because of the huge search space that may be generated. Even further, because of the possible existence of infinite branches that may be explored, the procedure will not terminate if  $I \not\models s \subseteq t$ .

But we can avoid all potentially infinite branches by using inclusions to replace subterms by smaller ones with respect to a well-founded ordering on terms, i.e., we use them as *rewrite rules*. Given a finite set of rewrite rules  $R$ , we say that a rule  $l \rightarrow r \in R$  reduces term  $s$  to  $t$ , and write  $s \rightarrow t$ , if a subterm  $u$  of  $s$ , which is an instance of the left-hand side  $l$ , is replaced by the corresponding instance of the right-hand side  $r$ , obtaining a simpler term  $t$ . If  $t$  is a term such that no rewrite rule of  $R$  applies to it, we say that it is an *irreducible* term with respect to  $R$ , and call it a *normal form*. Given a term  $t$ , we call a term  $s$ , such that  $t \xrightarrow{*} s$ , a term that is *reachable* from  $t$ .

**2.3.3 Remark.** Notice that, if we orient a set of inclusions following an ordering on terms, we get *two* independent rewrite systems, one consisting of those rewrite rules that rewrite terms into 'bigger' ones (with respect to the inclusion relation), and the other one consisting of those rewrite rules that rewrite terms into 'smaller' ones<sup>3</sup>. We distinguish the two separate rewrite relations by denoting the first one with  $\xrightarrow[\subseteq]{}^*$  and the second one with  $\xrightarrow[\supseteq]{}^*$ .

<sup>2</sup>Notice that the comparatives 'bigger' and 'smaller' result from the inclusion relation, not from the term ordering.

<sup>3</sup>Here we are supposing that all inclusions can be oriented. Unorientable inclusions will be considered later.

**2.3.4 Example.** Let us consider an inclusion theory defined by the following axioms:

$$\begin{aligned} f(a, x) &\subseteq x \\ f(x, c) &\subseteq x \\ b &\subseteq f(a, c) \end{aligned}$$

If we orient these inclusions, following an ordering on terms, we obtain the following two distinct rewrite systems:

$$\begin{aligned} R_{\subseteq} &= \left\{ \begin{array}{l} f(a, x) \xrightarrow{\subseteq} x \\ f(x, c) \xrightarrow{\subseteq} x \end{array} \right. \\ R_{\supseteq} &= \left\{ f(a, c) \xrightarrow{\supseteq} b \right. \end{aligned}$$

We say that these two rewrite systems form a *bi-rewrite system*  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$ .

**2.3.5 Bi-rewrite proof.** In such a system a *bi-rewrite proof* consists of two paths, one using rules of  $R_{\subseteq}$  and the other using rules of  $R_{\supseteq}$ , which join together in a common term:

$$s \xrightarrow[\subseteq]{*} \cdots \xrightarrow[\subseteq]{*} u \xleftarrow[\subseteq]{*} \cdots \xleftarrow[\subseteq]{*} t$$

There will be a bi-rewrite proof if the set of reachable terms from  $s$  with  $\xrightarrow[\subseteq]{*}$  and the set of reachable terms from  $t$  with  $\xleftarrow[\subseteq]{*}$  intersect in a nonempty set.

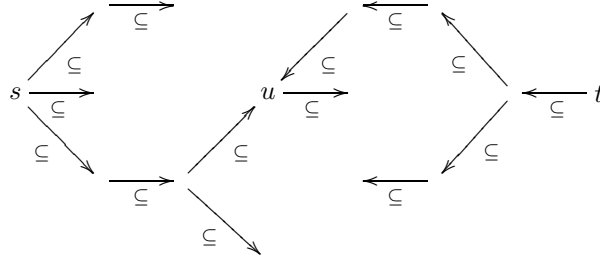
In order to have a decision algorithm for the word problem in a theory  $I$  we need the bi-rewrite system  $B$  to satisfy two properties: First, whenever for two terms  $s$  and  $t$ ,  $s \xleftarrow[\subseteq]{*} t$ , then a term  $u$  should exist, such that  $s \xrightarrow[\subseteq]{*} u$  and  $t \xrightarrow[\supseteq]{*} u$ . We denote the existence of such a term  $u$  by  $s \downarrow_{\subseteq} t$ . This is the *Church-Rosser property*. Second, and in order to avoid reducing terms  $s$  or  $t$  infinitely many times, we require that, applying one single rewrite rule, only finite many terms are reached (finite branching), and that only finite sequences of rewrites with rules in  $R_{\subseteq}$  (or  $R_{\supseteq}$ ) can be built. We say the bi-rewrite system  $B$  is *terminating*<sup>4</sup>. Termination can be obtained by the following theorem:

**2.3.6 Theorem.** A bi-rewrite system  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$  is terminating if the rewrite orderings  $\xrightarrow[\subseteq]{+}$  and  $\xrightarrow[\supseteq]{+}$  defined by  $R_{\subseteq}$  and  $R_{\supseteq}$  respectively are contained in a unique reduction ordering.

---

<sup>4</sup>Actually, for a decision algorithm, only quasi-termination suffices (Levy and Agustí, 1996)

**2.3.7 Decision algorithm.** Bi-rewrite systems fulfilling termination and the Church-Rosser property are *convergent*. The decision algorithm for the word problem is then straightforward: To check if  $s \xrightarrow{*} t$  we reduce  $s$  and  $t$ , applying rewrite rules of each rewrite system, until a common term is reached. Since the bi-rewriting system is terminating, only finite branches will be explored:



Like convergent rewrite systems are a finite representation of the congruence closure of equality, convergent bi-rewrite systems finitely encode the reflexive, transitive, and monotone closure of  $\subseteq$ . With convergent bi-rewrite systems, all possible consequences of a set of inclusions  $I$  using transitivity, reflexivity, and monotonicity are represented by a bi-rewrite proof.

An arbitrary bi-rewrite system, obtained by orienting the inclusions of a inclusion theory  $I$  is non-convergent in general. But, like in the equational case, there exist necessary and sufficient conditions for a terminating bi-rewrite system to be Church-Rosser, which were stated by Levy and Agustí, adapting the original results of Knuth and Bendix (1970). First of all, we give two definitions, and then we state the theorem that summarizes this result (see (Levy and Agustí, 1996) for further details):

**2.3.8 Definition.** Let  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$  be a bi-rewrite system, and let  $l \xrightarrow{\subseteq} r \in R_{\subseteq}$  and  $s[u]_p \xrightarrow{\supseteq} t \in R_{\supseteq}$  (or vice versa) be two rules, where  $u$  is not a variable. If  $\sigma$  is the most general unifier of  $l$  and  $u$ , then  $\langle \sigma(s[r]_p), \sigma(t) \rangle$  is called a *critical pair*.

**2.3.9 Definition.** Let  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$  be a bi-rewrite system, let  $l \xrightarrow{\subseteq} r$  be a rule of  $R_{\subseteq}$ , and let  $\sigma(s) \xrightarrow{\supseteq} \sigma(t)$  be an instance of a rule  $s \xrightarrow{\supseteq} t$  of  $R_{\supseteq}$  (or vice versa), where  $\sigma$  is a substitution such that, for some term  $v$  and some variable  $x$  at position  $p$  that appears more than once in  $s$ ,  $\sigma(x) = v[l]_q$  and  $\sigma(y) = y$ , if  $y \neq x$ . The critical pair  $\langle \sigma(s[v[r]_q]_p), \sigma(t) \rangle$  is called a *variable instance pair*<sup>5</sup>.

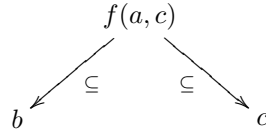
A critical pair or a variable instance pair  $\langle t, t' \rangle$  is said to be *convergent* if  $t \downarrow_{\subseteq} t'$  (or  $t \downarrow_{\supseteq} t'$ ), and *divergent* otherwise.

<sup>5</sup>The definition of variable instance pairs also appears in the context of rewrite systems modulo a congruence (see (Bachmair et al., 1986))

**2.3.10 Theorem.** *A terminating bi-rewrite system  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$  is Church-Rosser (and thus, convergent) if and only if there are no divergent critical pairs and variable instance pairs between the rules of  $R_{\subseteq}$  and the rules of  $R_{\supseteq}$ .*

Following the same ideas proposed by Knuth and Bendix (1970), one can attempt to complete a terminating, but non-confluent, bi-rewrite system by adding divergent critical pairs and divergent variable instance pairs as new rewrite rules to the systems  $R_{\subseteq}$  or  $R_{\supseteq}$ . The number of critical pairs among the rules of  $R_{\subseteq}$  and  $R_{\supseteq}$  is always finite. But from definition 2.3.9 of a variable instance pair, one sees that the overlap of term  $l$  on  $s$  is done *below* the variable position  $p$  of  $s$ , and thus unification always succeeds. Furthermore, term  $v$  is arbitrary, so that, if a variable instance pair exists between two rewrite rules, then there are infinite many of them. We will see later that this is one of the major drawbacks for a tractable generalization of rewrite techniques to arbitrary transitive relations: a completion procedure attempting to add infinite many variable instance pairs as new rewrite rules cannot manage them effectively.

**2.3.11 Example.** The bi-rewrite system of 2.3.4 is non-convergent because rules  $f(a, x) \xrightarrow[\subseteq]{} x$  and  $f(a, c) \xrightarrow[\supseteq]{} b$  yield a divergent critical pair:



Let us suppose that, in the term ordering,  $c \succ b$ . Since  $b \subseteq c$ , we obtain the following new rule:

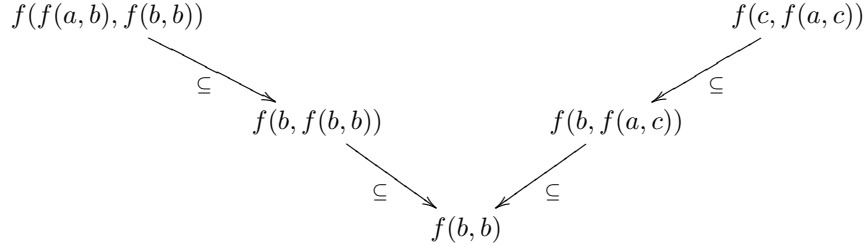
$$c \xrightarrow[\supseteq]{} b$$

A *Knuth-Bendix*-like completion procedure would add the critical pair  $\langle b, c \rangle$  as this new rewrite rule to the system. In this example, a convergent system can be built, because during the whole completion process no variable instance pairs among two rules arise. In fact, for this inclusion theory, the completion procedure would generate the following convergent bi-rewrite system:

$$R_{\subseteq} = \left\{ \begin{array}{ll} f(a, x) & \xrightarrow[\subseteq]{} x \\ f(x, c) & \xrightarrow[\subseteq]{} x \\ b & \xrightarrow[\subseteq]{} a \\ f(x, b) & \xrightarrow[\subseteq]{} x \end{array} \right.$$

$$R_{\supseteq} = \left\{ \begin{array}{ll} f(a, c) & \xrightarrow[\supseteq]{} b \\ c & \xrightarrow[\supseteq]{} b \end{array} \right.$$

Now we can prove the validity of inclusion  $f(f(a, b), f(b, b)) \subseteq f(c, f(a, c))$  by the following bi-rewrite proof:



**2.3.12 Remark.** Notice that  $f(b, b)$  is the only term reachable from  $f(f(a, b), f(b, b))$  with  $\xrightarrow[\subseteq]{*}$  which is also reachable from  $f(c, f(a, c))$  with  $\xrightarrow[\supseteq]{*}$ . But  $f(b, b)$  is not an irreducible term with respect to  $R_{\subseteq}$ , since  $f(b, b) \xrightarrow[\subseteq]{} b$ . This shows that, unlike the equational case, normal forms do not play any role in the search for a bi-rewrite proof.

**2.3.13 Completion modulo a theory.** The process of completion generates new critical pairs or variable instance pairs, and they need to be oriented in order to add them as new rewrite rules to the bi-rewrite system. Thus we must define a suitable reduction ordering on terms as to orient all these possible pairs.

Of course, like in equational theories, not for every given inclusion theory, and for every reduction ordering we define, the completion procedure terminates successfully, yielding a convergent bi-rewrite system for the inclusion theory. There exist theories for which the completion procedure will keep generating new critical pairs and never stop. But completion can also terminate unsuccessfully. It may abort without finding a convergent bi-rewrite system, though one exists. This happens, for example, when it finds a critical pair which cannot be oriented with the reduction ordering at hand. Sometimes this is solved by using stronger reduction orderings (which orient more pairs), but there are pairs, which no reduction ordering can orient. A typical example is the commutativity-like axiom:

$$f(x, y) \subseteq f(y, x)$$

It is obvious that no orientation of the inclusion preserves the termination property, because endless sequences of reductions can always be built:

$$f(x, y) \xrightarrow[\subseteq]{} f(y, x) \xrightarrow[\subseteq]{} f(x, y) \xrightarrow[\subseteq]{} \dots$$

Different methods were developed to overcome this problem. In the equational case Lankford and Ballantyne (1977) extended rewriting to rewriting modulo a set of equations. These set of equations are usually structural axioms, such

as the commutativity and the associativity axiom. Peterson and Stickel (1981) extended the unification mechanism used in the generation of critical pairs to a more general unification modulo associativity and commutativity, which Jouannaud and Kirchner (1986) extended to the general case. Both approaches treat the troublesome equations implicitly, not as rewrite rules. These ideas were generalized to bi-rewrite systems, too (Levy and Agustí, 1996).

**2.3.14 The role of symmetry in bi-rewrite systems.** If the arbitrary transitive relation we reason with is, in addition to reflexive and monotonic, also symmetric, then we are dealing with a congruence relation. Bi-rewrite systems based on equality happen to be standard rewrite systems as known from the equational case, because, when orienting equations following a given reduction ordering, we only obtain one single rewrite system, not two.

**2.3.15 Example.** Let us take the theory of groups, defined by the following set of equations:

$$\begin{aligned} e * x &\approx x \\ x^{-1} * x &\approx e \\ (x * y) * z &\approx x * (y * z) \end{aligned}$$

These equations are oriented leading to the following rewrite system:

$$\begin{aligned} e * x &\xrightarrow{\approx} x \\ x^{-1} * x &\xrightarrow{\approx} e \\ (x * y) * z &\xrightarrow{\approx} x * (y * z) \end{aligned}$$

The symmetry of the equality relation makes an equality to be equivalent to two inclusions (since the relation defined by the inclusions is also an equivalence relation), and thus we can define the theory of groups by the following set of inclusions:

$$\begin{aligned} e * x &\subseteq x \\ e * x &\supseteq x \\ x^{-1} * x &\subseteq e \\ x^{-1} * x &\supseteq e \\ (x * y) * z &\subseteq x * (y * z) \\ (x * y) * z &\supseteq x * (y * z) \end{aligned}$$

By orienting them with respect to a reduction ordering we obtain the following bi-rewrite system:

$$R_{\subseteq} = \left\{ \begin{array}{ll} e * x & \xrightarrow{\subseteq} x \\ x^{-1} * x & \xrightarrow{\subseteq} e \\ (x * y) * z & \xrightarrow{\subseteq} x * (y * z) \end{array} \right.$$

$$R_{\supseteq} = \left\{ \begin{array}{ll} e * x & \xrightarrow{\supseteq} x \\ x^{-1} * x & \xrightarrow{\supseteq} e \\ (x * y) * z & \xrightarrow{\supseteq} x * (y * z) \end{array} \right.$$

**2.3.16 Remark.** Notice that each former equation appears as a rewrite rule in both rewrite systems. Therefore, the generation of critical pairs, which is done by looking for overlaps between two rules, one of each rewrite system, is equivalent to look for overlaps among the rules of one unique equational rewrite system. Every overlap can be done twice, yielding each time two new inclusions. The result is resumed in the standard definition of critical pair:

**2.3.17 Definition.** Let  $R$  be a term rewrite system, and let  $l \xrightarrow{\approx} r$  and  $s[u]_p \xrightarrow{\approx} t$  be two rules in  $R$ , where  $u$  is not a variable. If  $\sigma$  is a most general unifier of  $l$  and  $u$ , then  $\langle \sigma(s[r]_p), \sigma(t) \rangle$  is called a *critical pair*.

**2.3.18 Remark.** It is easy to check that, because of symmetry, all variable instance pairs are convergent, and therefore do not need to be generated. Thus, no overlap on and below variable positions is required.

**2.3.19 Unique normal form.** Symmetry plays an important role in term rewriting, because, when reasoning with equivalence relations, we deal with the notion of equivalence class. Since we do not have two separate rewrite systems any more, critical pairs are computed by overlapping left-hand sides of rules of one single rewrite system. Thus, if the rewrite system is convergent, each term not only has an irreducible term, the *normal form*, but this normal form is also unique for each of them. Rewriting is done within an equivalence class, and all its members share the same normal form. A decision procedure for the word problem in equational theories, based on convergent rewriting systems, is much simpler than with arbitrary transitive relations, since backtracking can be avoided. We just compute the normal forms of both terms whose equality we want to validate, and check them for syntactic identity. This is an important difference of bi-rewrite systems with respect to equational rewrite systems: since rewriting is not done within equivalence classes, the notion of unique normal form becomes meaningless.

For a more exhaustive study of rewrite systems, see the surveys of Dershowitz and Jouannaud (1990), Huet (1980), Plaisted (1993), Klop (1992), and, more recently, of Baader and Nipkow (1998)

## 2.4 Rewrite Techniques in Theorem Proving

As mentioned in 2.2.1, in order to avoid failure during the completion process, completion itself can act as a deduction process for proving theorems, by



*unfailing completion.* The same principle can be generalized for the unfailing completion of bi-rewrite systems. But let us emphasize these notions by looking at the equational case.

Huet suggested to use the completion procedure not only to obtain convergent rewrite systems, but also to prove theorems of equational theories (Huet, 1981). He showed that, whenever the completion process does not fail and all newly generated equations are orientable, it is a semi-decision procedure for the word problem of equational theories.

**2.4.1 Theorem proving by completion.** Let  $E$  be a set of equations, and  $R$  the set of rewrite rules obtained by orienting the equations of  $E$ . Suppose we want to check if  $s \xrightarrow{*} t$ . We first reduce  $s$  and  $t$  to their normal forms with respect to  $R$ , obtaining  $s'$  and  $t'$ , respectively. If they are identical,  $s \approx t$  is provable in  $E$ . Otherwise, we run the completion procedure on  $R$ , following a suitable fairness criterion<sup>6</sup>, finding divergent critical pairs and adding them as new rewrite rules to  $R$ , until a new system  $R'$  is obtained, for which  $s'$  or  $t'$  are reducible. Again, we compute the normal forms of  $s$  and  $t$  (now with respect to  $R'$ ) and check them for identity. We repeat this process until we reach two identical normal forms—and thus the equation  $s \approx t$  is provable in  $E$ —or we do not find more divergent critical pairs, obtaining a convergent rewrite system with distinct normal forms for  $s$  and  $t$ —and thus  $s \approx t$  is not provable in  $E$ . Since the word problem is semi-decidable, it may also happen that, though  $s \approx t$  is not provable in  $E$ , the completion process never ends, unable to yield a convergent rewrite system. Notice that the basic idea is to progressively make divergent critical pairs convergent, until we obtain a rewrite proof. The same principle can be generalized for arbitrary transitive relations by progressively making critical pairs and variable instance pairs converge, such that we eventually get a bi-rewrite proof.

**2.4.2 Unfailing completion.** Unfortunately, the completion procedure may fail to orient a critical pair as a rewrite rule and the process is prone to abort. This makes standard Knuth-Bendix completion inappropriate for theorem proving purposes. Bachmair, Dershowitz, and Plaisted (1989) overcame this situation with a variant of completion, called *unfailing completion*, which is refutationally complete for equational theories.

In unfailing completion, first of all, we attempt to complete a set of equations rather than a set of rewrite rules, because we want to manage unorientable equations. Even if there is no convergent rewrite system for a given equational theory, it may still be possible to construct a system of equations that possesses a weaker Church-Rosser property, namely a Church-Rosser property only on *ground* (variable free) terms. Such a system defines unique ground normal forms, which suffices for most purposes, including refutational theorem proving, as we will see in 2.4.5. In this approach, the notion of rewriting is refined as follows:

---

<sup>6</sup>The fairness criterion guarantees that sooner or later the completion process generates all possible critical pairs.

**2.4.3 Definition.** Given a reduction ordering  $\succ$ ,  $s$  is rewritten to  $t$  and we denote it by  $s \xrightarrow[\approx]{\succ} t$ , if  $s \xleftrightarrow[\approx]{\approx} t$  by applying some equation  $u \approx v$  in  $E$  with a substitution  $\sigma$ , for which  $\sigma(u) \succ \sigma(v)$ . We call  $\sigma(u) \approx \sigma(v)$  an *orientable instance* of the equation  $u \approx v$ .

**2.4.4 Remark.** Notice that rewriting is now performed with equations rather than with rewrite rules. Since we want the system to be ground Church-Rosser, the unfailing completion process adds to the system those equations that may have *as instance* a divergent critical pair. These divergent critical pairs must be generated by the overlap of *orientable instances* of equations.

**2.4.5 Refutation by completion.** Let  $\hat{s} \approx \hat{t}$  be the equation obtained by replacing each variable in  $s$  and  $t$  with a unique Skolem constant. If the given reduction ordering  $\succ$  is complete on  $\mathcal{T}_{\Sigma \cup K}(\mathcal{X})$ , where  $K$  is the set of Skolem constants occurring in  $\hat{s}$  and  $\hat{t}$ , then unfailing completion is refutationally complete: If  $s \approx t$  is provable in  $E$ , then  $\hat{s} \approx \hat{t}$  is also provable in  $E$ , and hence unfailing completion can be used as a semi-decision procedure to find a ground rewrite proof of  $\hat{s} \approx \hat{t}$ . Conversely if  $\hat{s} \approx \hat{t}$  is provable in some  $E'$  obtained during the unfailing completion process, then  $s \approx t$  is provable in  $E$ . The idea of completion as a refutationally complete deduction process, can also be generalized to full first-order clauses, as we show next.

**2.4.6 Ordered chaining.** Like with paramodulation in first-order logic with equality (see 2.2.2), there have been attempts to use additional inferences for resolution-based theorem provers in order to avoid the transitivity axiom of arbitrary transitive relations. As mentioned in 2.3.1, Slagle introduced for these purposes the chaining inference rule. Chaining can be seen as the generalization of paramodulation for arbitrary transitive relations, but in this case without chaining on subterms, because the transitive relation needs not to be monotonic. Like with paramodulation, chaining has the drawback that it explicitly generates the transitive closure of the binary relation. As mentioned previously, there is an extra cost in this generalization, since chaining through variables is now needed, in general, unlike with paramodulation.

But like ordering restrictions on the paramodulation inference led to the superposition calculus—which in essence generalizes the critical pair generation of the unfailing variant of the Knuth-Bendix completion procedure—, so ordering restrictions on the chaining inference rule profit of the advantages of bi-rewriting techniques, avoiding the generation of the whole transitive closure by using bi-rewrite proofs to validate an inequation. So the ordered chaining proof calculus, introduced by Bachmair and Ganzinger (1994c), is based on the chaining inference with additional ordering restrictions on terms and literals.

Figure 2.1 shows the proof calculus, which is refutationally complete for Horn theories with transitive relation  $<$ . A complete calculus for full first-order theories with such transitive relation requires some additional inference rules (see (Bachmair and Ganzinger, 1994c) for further details).

Ordered positive chaining:

$$\frac{C, u < s \quad D, t < v}{\sigma(C), \sigma(D), \sigma(u) < \sigma(v)} \quad \left\{ \begin{array}{l} \sigma = mgu(s, t) \\ \sigma(u) < \sigma(s) \text{ strictly maximal wrt. } \sigma(C) \\ \sigma(t) < \sigma(v) \text{ strictly maximal wrt. } \sigma(D) \\ \sigma(u) \neq \sigma(s) \\ \sigma(v) \neq \sigma(t) \end{array} \right.$$

Ordered negative chaining:

$$\frac{C, u \not< s \quad D, t < v}{\sigma(C), \sigma(D), \sigma(v) \not< \sigma(s)} \quad \left\{ \begin{array}{l} \sigma = mgu(u, t) \\ \sigma(u) \not< \sigma(s) \text{ maximal wrt. } \sigma(C) \\ \sigma(t) < \sigma(v) \text{ strictly maximal wrt. } \sigma(D) \\ \sigma(s) \neq \sigma(u) \\ \sigma(v) \neq \sigma(t) \\ \sigma(s) \neq \sigma(v) \end{array} \right.$$

$$\frac{C, u \not< s \quad D, t < v}{\sigma(C), \sigma(D), \sigma(u) \not< \sigma(t)} \quad \left\{ \begin{array}{l} \sigma = mgu(s, v) \\ \sigma(u) \not< \sigma(s) \text{ maximal wrt. } \sigma(C) \\ \sigma(t) < \sigma(v) \text{ strictly maximal wrt. } \sigma(D) \\ \sigma(u) \neq \sigma(s) \\ \sigma(t) \neq \sigma(v) \\ \sigma(u) \neq \sigma(t) \end{array} \right.$$

Ordered resolution:

$$\frac{C, \neg A \quad D, B}{\sigma(C), \sigma(D)} \quad \left\{ \begin{array}{l} \sigma = mgu(A, B) \\ \sigma(A) \text{ maximal wrt. } \sigma(C) \\ \sigma(B) \text{ strictly maximal wrt. } \sigma(D) \end{array} \right.$$

**Figure 2.1:** Ordered chaining proof calculus

**2.4.7 Saturation.** Recall unfailing completion as a semi-decision procedure for theorem proving. Like in the equational case, the generation of new inequations from critical pairs and variable instance pairs can be seen as an ordered version of the chaining inference rule. In this context, like in the equational case, the process of completion is known as *saturation*. In analogy to a completion procedure that attempts to produce a convergent bi-rewrite system, in which all critical pairs (and variable instance pairs, if the transitive relation is also monotonic) are convergent (have a bi-rewrite proof), the saturation process attempts to provide us with a set of clauses in which all inferences are *redundant*. We say that the set of clauses is saturated, i.e., *closed up to redundancy*. Notice that this is the criterion in order to finish the process of completion, or saturation respectively. Like during the completion process rewrite rules are kept as interreduced as possible, during saturation redundant clauses are deleted and redundant inferences avoided. Although we will often come back to the notion of redundancy, we define this notion in more detail later, within the context of

putting all these concepts into practice.

As we will see next, Bachmair and Ganzinger (1994c) proved the refutational completeness of several inference calculi based on ordered chaining for Horn clauses and for general first-order clauses with arbitrary transitive relations with the help of bi-rewrite systems.

**2.4.8 Completeness of ordered chaining calculi.** As explained in 2.3.2, Levy and Agustí (1993) gave the sufficient and necessary conditions to identify those critical pairs that need to converge in order to build a *Knuth-Bendix*-like completion procedure for inclusion theories. We have seen that, by completion, we may get a convergent bi-rewrite system, which is a finite representation of the transitive closure of the binary relation. Bachmair and Ganzinger used these techniques at a more abstract level, in the context of their *model construction* method.

Bachmair and Ganzinger (1990) introduced this very useful method for proving refutational completeness of inference systems based on saturation, and which also provides us a very elegant way to define *redundancy*, as we will see later. It consists in constructing a Herbrand interpretation  $I$  from a *saturated* set of clauses  $N$  in such a way that, whenever  $N$  is satisfiable,  $I$  is a model of  $N$ . But whenever  $N$  is unsatisfiable, the minimal counter-example showing that  $I$  is not a model of  $N$  is the empty clause. This means that, in some step of the saturation process, the empty clause is added to the set of clauses. Note that the minimal counter-example is the smallest clause in  $N$  (with respect to the extension to an ordering on clauses of the reduction ordering  $\succ$ ) that is false in  $I$ .

The construction of this Herbrand interpretation is done inductively over a given total reduction ordering on ground clauses. For simplifying purposes, here we consider clauses without transitive relations: Starting with the empty interpretation  $I = \emptyset$ , we repeat the following steps: First, we take the smallest possible instance  $C$  of a clause of  $N$  that is false in  $I$ . Then we take a positive literal  $A$  of  $C$  that meets some ‘desired’ conditions<sup>7</sup>, and add it to  $I$ . A clause  $C$  with such a literal is called a *productive* clause. Obviously,  $C$  is true in the new interpretation  $I \cup \{A\}$ . The result of repeating these steps is an interpretation with the properties explained above.

**2.4.9 Example.** Consider the following ground clauses, which are statements about some property  $P$  of the naturals in Peano notation, and which is a saturated set, in the sense explained before<sup>8</sup>. The clauses are listed in

<sup>7</sup>See (Bachmair and Ganzinger, 1990) for how these conditions are determined.

<sup>8</sup>This example is taken from (Bachmair and Ganzinger, 1994d).

Clause set	Interpretation
$\boxed{P(0)}$	$P(0)$
$P(0) \vee P(\text{succ}(0))$	
$P(\text{succ}(0)) \vee \boxed{P(\text{succ}(\text{succ}(0)))}$	$P(\text{succ}(\text{succ}(0)))$
...	...
$P(\text{succ}^{2n-1}(0)) \vee \boxed{P(\text{succ}^{2n}(0))}$	$P(\text{succ}^{2n}(0))$
$P(\text{succ}^{2n}(0)) \vee P(\text{succ}^{2n+1}(0))$	
...	...

**Figure 2.2:** Model construction

increasing order:

$$\begin{aligned}
&P(0) \\
&P(0) \vee P(\text{succ}(0)) \\
&P(\text{succ}(0)) \vee P(\text{succ}(\text{succ}(0))) \\
&\dots \\
&P(\text{succ}^{2n-1}(0)) \vee P(\text{succ}^{2n}(0)) \\
&P(\text{succ}^{2n}(0)) \vee P(\text{succ}^{2n+1}(0)) \\
&\dots
\end{aligned}$$

Figure 2.2 shows how the interpretation would be constructed. The framed literals represent literals of productive clauses that meet the ‘desired’ condition (in this simple example, they must be the strictly maximal literals of their respective clauses). The constructed interpretation is the union of all literals appearing under the column **Interpretation**, and it is a model of this set of clauses. Notice that clause  $P(0) \vee P(\text{succ}(0))$  is not productive, because it is true in the interpretation  $\{P(0)\}$  constructed so far.

In the case of full first-order clauses with arbitrary transitive relations, productive literals will be ordered inequations, i.e., rewrite rules of a bi-rewrite system. Therefore, the constructed interpretation will be represented by a convergent bi-rewrite system, which finitely encodes the transitive closure of the binary relation.

**2.4.10 Redundancy.** The model construction method gives a very simple way to define the notion of redundancy, introduced before: A ground clause  $C$  is *redundant* in the saturated set  $N$  of clauses if it is non-productive, and thus does not contribute to the model construction. This means that other clauses produce the necessary literals, so that clause  $C$  is true in the final interpretation. A non-ground clause is redundant if all its ground instances are. It is easy, now, to give a sufficient condition for redundancy: A ground clause  $C$  is redundant in  $N$  if there exist ground instances  $D_1, \dots, D_n$  of clauses in  $N$ , such that  $D_1, \dots, D_n \models C$ , and  $C \succ D_i$ , for all  $i = 1 \dots n$ . This definition captures the redundancy, for example, of tautologies, subsumed clauses, and condensed

clauses, as expected.

A ground inference is redundant when either one of its premises is redundant or else the conclusion does not contribute to the model construction. A non-ground inference is redundant if all its ground instances are. As before, we can give a sufficient condition for redundancy: A ground inference with premises  $C$  and  $D$  and conclusion  $B$  is redundant ( $C$  being the maximal premise), whenever there exist ground instances  $D_1, \dots, D_n$  such that  $D_1, \dots, D_n \models B$ , and  $C \succ D_i$ , for all  $i = 1 \dots n$ . This redundancy notion generalizes the idea of confluent critical pairs, i.e., those pairs that have already a rewrite proof, and thus need not to be added as a new equations to the set of equations.

**2.4.11 Example.** With this notion it is easy to prove the redundancy of superposition on variables<sup>9</sup>: If we have an ordering on terms, such that ground terms  $u \succ v$  and  $f(u, u) \succ g(u)$ , then the superposition inference

$$\frac{u \approx v \quad f(x, x) \approx g(x)}{f(u, v) \approx g(u)}$$

is redundant. It is obvious that the only possible ground instance of this inference, namely

$$\frac{u \approx v \quad f(u, u) \approx g(u)}{f(u, v) \approx g(u)}$$

has as maximal premise  $f(u, u) \approx g(u)$ . Now we have that

$$u \approx v, f(v, v) \approx g(v) \models f(u, v) \approx g(u)$$

and  $f(u, u) \approx g(u) \succ u \approx v$ , and  $f(u, u) \approx g(u) \succ f(v, v) \approx g(v)$ .

## 2.5 Putting it into Practice

Theorem provers typically consist of two components: On one hand, we require a deductive inference system that is used to generate new formulae, such as the chaining inference rules given in figure 2.1. On the other hand, we need also techniques for simplifying formulae and eliminating redundancies, because we want a theorem prover to be as efficient as possible, without generating superfluous clauses or exploring blind allays of the search space.

In the context of the saturation of a set of first-order clauses, the theorem prover needs to implement powerful *redundancy provers*, which allow to drastically cut down the search space of the theorem prover, by maintaining only the minimum number of necessary clauses, keeping them as small as possible with respect to the ordering on clauses, and by avoiding those inferences that do not contribute to the saturation of the set.

These redundancy provers are expected to detect, for example, tautologies, subsumption within clauses, condensation within clauses, and are also expected

---

<sup>9</sup>This example is taken from (Bachmair and Ganzinger, 1994d).

to simplify clauses as much as possible, for example, by demodulation. Redundancy provers also should detect when a set is already saturated and no more inferences need to be done. Without the use of powerful redundancy provers the saturation of a set of clauses is practically impossible. We will see that several important drawbacks when handling arbitrary transitive relations appear at this point.

**2.5.1 The Saturate theorem prover.** All the theoretical results discussed so far have been put into practice in the **Saturate** theorem prover, implemented by Nivela and Nieuwenhuis (1993), and further developed by Ganzinger (Ganzinger et al., 1995). The system includes other additional constraints introduced by Nieuwenhuis and Rubio to the superposition inference rule, like the use of *basic strategies* and of *constrained clauses* (Nieuwenhuis and Rubio, 1995; Bachmair et al., 1995), which prune even more the required search space. Special emphasis was put on the implementation of powerful redundancy provers; on them lies the efficiency of the theorem prover. Thus **Saturate** checks, for example, for tautology, subsumption, and condensation. It also simplifies clauses by demodulation and makes case analysis by constrained rewriting. Redundant inferences are checked by clausal rewriting. All these redundancy provers are covered by the notion of redundancy introduced by Bachmair and Ganzinger, and are based on looking for *rewrite proofs*.

**2.5.2 Drawbacks.** We have pointed out several differences that appear when applying rewrite techniques to other transitive relations in addition to equality. So, instead of dealing with a single rewrite relation, we have to handle a bi-rewrite system. There is no rewriting within equivalence classes, making unique normal forms, on which equational rewrite systems are based, meaningless. Consequently, the order of application of rewrite rules is now significant for finding a rewrite proof. We also pointed out in 2.3.2 that for the completion of bi-rewrite systems we need to generate variable instance pairs by computing critical pairs between instances of rules.

Although Levy and Agustí, and Bachmair and Ganzinger proved the completeness of their respective inference systems, and though Ganzinger put them into practice extending the **Saturate** theorem prover to arbitrary transitive relations, the differences with respect to the equational case appear to cause many problems for an efficient application of these rewrite techniques, as we will see in the following paragraphs.

**2.5.3 Redundancy provers and rewrite proofs.** Though the model construction method provides a powerful abstract notion of redundancy—which is necessary for the implementation of efficient theorem provers—without a corresponding efficient decision procedure yielding bi-rewrite proof for inequations, we will hardly be able to have efficient redundancy provers.

The notion of redundancy derives from the model construction method, and is also applicable to clauses and inferences with arbitrary transitive relations.

Most existing techniques for redundancy proving in the equational case are based on rewrite proofs, but recall that, in order to find bi-rewrite proofs based on a bi-rewrite system, we need to search among all possible paths that can be built starting from each term. This is an important drawback for developing efficient redundancy provers in these kind of theories, and hence for efficient theorem proving.

**2.5.4 Variable chaining.** Brand (1975) made an important refinement to the paramodulation inference by proving that no paramodulation through variable positions is required for completeness. We have seen, in 2.4.10 and 2.4.11, that this is captured in the redundancy notion based on the model construction, since inferences through variable position are redundant. Unfortunately, this is false with chaining, since chaining through variables is necessary for completeness. This is an important drawback, because unification with variables always succeeds, and therefore the inference is very prolific. So, in order to make the theorem proving derivation efficient, we have to avoid as much variable chainings as possible. Fortunately, the ordering restrictions of ordered chaining evade many of them: The variable  $x$  must be a maximal term in the premise, and therefore  $x$  cannot occur in a negative literal and also must be *unshielded* in the clause. We say that a variable is unshielded if it only occurs as an argument of the transitive relation, but not as an argument of any other predicate or function symbol. In addition, when the variable  $x$  is linear, that is, it only occurs once in the premise, the inference is also redundant, because in these cases the conclusion is subsumed by one of the premises. For further details, we refer to the work of Bachmair and Ganzinger (1994c).

Another way to restrict the prolific variable chaining is by studying special transitive relations, for which these kind of chainings are redundant. In dense total orderings without endpoints, for example, Bledsoe and Hines (1980) developed a technique to eliminate variables from formulae. Bachmair and Ganzinger (1994a) showed that the variable elimination technique in these theories is a simplification rule captured by the notion of redundancy, in the sense explained above. So, after applying variable elimination to a clause, it becomes redundant, and can be eliminated. In this way variable chaining through variables can be completely avoided.

**2.5.5 Chaining below variables.** A quite problematic case, in the sense of practicability, is when we handle functions that are monotonic or antimonotonic with respect to the transitive relation, but that are not symmetric: This is the case if for some function  $f$

$$a < b \implies f(a) < f(b) \quad \text{or} \quad a < b \implies f(a) > f(b) .$$

In these cases *variable subterm chaining* is necessary, which leads to many problems if completeness results are desired.

In 2.3.2, we pointed out that computing the critical pairs between rules of each rewrite relation, as in the Knuth-Bendix completion, was not sufficient in



order to guarantee the bi-rewrite system to be Church-Rosser. Levy and Agustí proved that *extended* critical pairs need to be computed (Levy and Agustí, 1996). The definition of an extended critical pair is slightly different and more restrictive as the one of a variable instance pair, given in definition 2.3.9:

**2.5.6 Definition.** Let  $B = \langle R_{\subseteq}, R_{\supseteq} \rangle$  be a bi-rewrite system, and let  $l \xrightarrow{\subseteq} r$  and  $s \xrightarrow{\supseteq} t$  be two rules of  $R_{\subseteq}$  and of  $R_{\supseteq}$ , respectively (or vice versa), such that  $x$  is a variable at position  $p$  that occurs more than once in  $s$ , and  $v$  is an arbitrary term with  $l$  as subterm at position  $q$ , and without any occurrence of  $x$ . Furthermore,  $l \xrightarrow{\supseteq}^* r$ . If  $\sigma$  is such that  $\sigma(x) = v[l]_q$  and  $\sigma(y) = y$ , if  $y \neq x$ , then  $\langle \sigma(s[v[r]_q]_p), \sigma(t) \rangle$  is called an *extended critical pair*.

Levy and Agustí extended the notion Church-Rosser known of equational rewrite systems to bi-rewrite systems in order to define a completion procedure for bi-rewrite systems based on theories of reflexive and monotonic transitive relations (inclusions). Notice that if  $l \xrightarrow{\supseteq}^* r$ , then the extended critical pair converges, as happens when the transitive relation is also symmetric (see 2.3.14).

**2.5.7 Example.** For example with rewrite rules:

$$a \xrightarrow{\supseteq} b \quad (2.1)$$

$$f(x, x) \xrightarrow{\subseteq} x \quad (2.2)$$

where  $x$  occurs twice in  $f(x, x)$ . Take, for example,  $\sigma$  to substitute  $x$  with  $g(a)$ ; then  $\langle f(g(b), g(a)), g(a) \rangle$  is an extended critical pair between rule (2.1) and (2.2).

**2.5.8 Remark.** Notice that the term  $v$  in definition 2.5.6 can be any term built with function symbols of the signature, whenever it has  $a$  as subterm. Therefore, there are infinite many extended critical pairs between rules (2.1) and (2.2), and hence the completion process needs to add infinite many new rewrite rules to the bi-rewrite system. They can be represented by the following rule schema

$$f(C[b], C[a]) \xrightarrow{\subseteq} C[a] ,$$

where  $C[ ]$  can be seen as an arbitrary *context*. The generation of extended critical pairs can then be seen as the conclusion of an inference that chains below variables:

$$\frac{b \subseteq a \quad f(x, x) \subseteq x}{f(C[b], C[a]) \subseteq C[a]}$$

Note that  $a$  is unified with some term *below* variable  $x$ , and therefore context  $C[ ]$  may be arbitrary.

**2.5.9 Towards a completion procedure.** Computation of extended critical pairs can be avoided by using only left-linear rewrite system, i.e., rewrite system where all left-hand sides of rules have only single occurrences of variables; but this is too strong a restriction. Though it is impossible, in practice, to add infinite many new rewrite rules to a rewrite systems, Levy and Agustí (1996) attempted to build a completion procedure based on rule schemas, which encode the infinite number of new rewrite rules. They were able to give canonical bi-rewrite systems for the inclusion theory of the union, and of non-distributive and distributive lattices. This demonstrates the power of bi-rewriting as theorem proving technique, since Freese, Ježek, and Nation proved that no finite and convergent associative-commutative term rewriting system exists for deciding the word problem in the theory of free lattices (Freese et al., 1993). Based on the idea of handling extended critical pairs with rule schemas, Levy proposed in (Levy, 1993) to attack this problem by means of using second-order bi-rewrite systems in order to handle the contexts mentioned above as second-order variables. In this case, the inference requires the unification of terms with context variables, i.e., of linear second-order terms. Levy proposed a semi-decision procedure for context unification, and together with Villaret, he conjectures that context unification is decidable (Levy and Villaret, 1999).

## 2.6 Recent Advances

We have given an overview of the most important achievements accomplished by the theorem proving community in generalizing the term rewriting technique from equational reasoning to inequational reasoning. But these techniques still stay with binary relations that are transitive. Recent work in this direction includes also Inverardi's rewriting systems for preorder theories (Inverardi, 1995). Struth (1997) specifically investigated the use of non-symmetric term rewriting for solving the word problem for free lattices, giving a formal completeness proof of the decision algorithm originally put forth by Levy (1995).

Probably the most general approach was done by Bachmair and Ganzinger, who extended their ordered chaining inference mechanism to reason in first-order theories with multiple transitive relations (Bachmair and Ganzinger, 1998); but their calculus avoids general monotonicity axioms in order to evade prolific variable subterm chaining inferences. Another quite general approach is Kriaučiukas and Walicki's application of term rewriting techniques for non-deterministic specifications with set relations (Kriaučiukas and Walicki, 1995; Kriaučiukas and Walicki, 1996), since they cope with three binary relations, one of them non-transitive.

All this research, that we have surveyed here with more or less detail, served as starting point for us to attempt to further generalize the term rewriting technique in order to deal with multiple completely arbitrary binary relations at once, and including very general notions of monotonicity and antimonotonicity. In the next part, we lay down the framework in which we develop these techniques.

## Part II

# Specification and Proof with Special Relations



## Chapter 3

# A Logic of Special Relations

We want to identify the logic that captures the role played by special relations within algebraic specifications. But we are going to tailor this logic having in mind that we are actually interested in a proof calculus anchored on a general notion of term rewriting along binary relations, which we are going to develop in detail in chapter 4. Therefore, many design decisions that we take in proposing this *logic of special relations* will be fully understood once we have discussed our generalization of the term rewriting technique. But, for the moment, we will at least motivate some of these decisions by means of an example.

### 3.1 Specifying with Special Relations

**3.1.1 Equational type logic.** Take, for instance, specifications based on equational type logic (Manca et al., 1990). Recall that equational type logic extends many-sorted equational logic by treating sorts semantically. Its axioms are Horn clauses involving equations  $t \approx t'$  as well as type assignments  $t : t'$ . Therefore, its proof calculus includes, in addition to the inference rules for Horn logic (tautology, monotonicity, substitution, and modus ponens), also inference rules capturing the particular properties of equality and type assignment, in order to reason with them. Thus, we need rules coping with the reflexivity, symmetry, transitivity, and replacement properties of equality, as well as rules coping with typing equals and equating types (see figure 3.1).

But observe that the TRANSITIVITY, TYPING EQUALS, and EQUATING TYPES inference rules of figure 3.1 are actually a combination of the following COMPOSITION and PARTIAL ORDER rules:

$$\text{COMPOSITION: } \frac{t \alpha t' \quad t' \beta t''}{t \alpha; \beta t''}$$

$$\text{PARTIAL ORDER: } \frac{t \alpha t'}{t \beta t'} \quad \text{whenever } \alpha \sqsubseteq \beta$$

They capture together the following properties of special relations  $\approx$  and  $:$ ,

1. REFLEXIVITY: For each term $t$	$\overline{t \approx t}$
2. SYMMETRY:	$\frac{t \approx t'}{t' \approx t}$
3. TRANSITIVITY:	$\frac{t \approx t' \quad t' \approx t''}{t \approx t''}$
4. REPLACEMENT: For each $n$ -ary function symbol $f$	$\frac{t_1 \approx t'_1 \cdots t_n \approx t'_n}{f(t_1, \dots, t_n) \approx f(t'_1, \dots, t'_n)}$
5. TYPING EQUALS:	$\frac{t \approx t' \quad t : u}{t' : u}$
6. EQUATING TYPES:	$\frac{u \approx u' \quad t : u}{t : u'}$

**Figure 3.1:** Inference rules coping with equations and type assignment

expressed as relation-algebra expressions:

$$\begin{aligned} \approx; \approx &\sqsubseteq \approx \\ :: \approx &\sqsubseteq : \\ \approx; : &\sqsubseteq : \end{aligned}$$

Recall that the symbol  $;$  denotes composition of relations, and  $\sqsubseteq$  a partial order over relations.

Furthermore, the REFLEXIVITY and SYMMETRY inference rules of figure 3.1 are also a combination of the following IDENTITY and CONVERSE rules, each of them plus PARTIAL ORDER as before:

$$\text{IDENTITY: } \frac{}{t \text{ } I' \text{ } t} \text{ for each term } t$$

$$\text{CONVERSE: } \frac{t \alpha t'}{t' \tilde{\alpha} t}$$

They capture then the following properties of special relation  $\approx$ :

$$\begin{aligned} I' &\sqsubseteq \approx \\ \approx &\sqsubseteq \approx \end{aligned}$$

where the symbol  $1'$  denotes the identity relation, and  $\smile$  denotes reversion of relations.

**3.1.2 A logic of special relations.** In general, our logic of special relations is going to deal with positive Horn formulae with predicates restricted to binary relations. Properties of these relations are determined by a suitable relation-algebra structure, which is taken into account by the proof calculus. Consequently, its inference rules cope with identity relations, composition of relations, and reversion of relations. Later, in chapter 4, we will see that reversion is essential when relations are non-symmetric, since when rewriting is governed by the purpose of doing it in the direction that syntactically simplifies terms, this can be in either way of the binary relation. The possibility of doing a replacement within the structure of a term is built into the entailment of our logic by means of monotonicity properties for specific argument positions of function symbols. Relationships existing between special relations are captured by a partial order in this relation-algebra structure.

**3.1.3 Relation-algebra expressions.** We use relation algebra to express properties of special relations because of its expressiveness. This will become clearer when introducing the semantics of the logic in section 3.3. But for computational purposes—since we are interested in term rewriting—we do not take advantage of the whole expressive power of relation algebra; we already mentioned that we basically need composition, identity, and reversion. Of course, by extending to wider fragments of relation algebra we would be able to capture, within our framework, more expressive specification paradigms.

**3.1.4 Overview.** The following sections formally introduce the logic of special relations. We first define its syntax giving special attention to the notion of *polarity*, which will play a central role when term rewriting techniques are studied in chapter 4. Next, we study its semantics, first, giving an intuitive set-valued interpretation of the syntax, but then moving on towards a more abstract presentation within the realm of category theory, to be able to capture disparate specification paradigms. For the sake of simplicity, we only treat the unconditional fragment of the logic, but we end this chapter with a brief discussion on how the framework extends to conditional sentences.

## 3.2 Syntax

**3.2.1 Signatures.** Signatures of the logic of special relations are tuples  $\Omega = (\mathcal{S}^*, \Sigma)$ , where

- $\mathcal{S}^* = (S^*, ;, 1', \smile, \sqsubseteq)^1$  forms a partially ordered free monoid with an anti-involution generated over a set  $S$  of special binary relation symbols, i.e.,

---

<sup>1</sup>We are following the notation of relation algebra given in (Maddux, 1991).

for all  $\alpha, \beta, \gamma \in S^*$ ,

$$\alpha; (\beta; \gamma) = (\alpha; \beta); \gamma$$

$$1'; \alpha = \alpha$$

$$\check{\check{\alpha}} = \alpha$$

$$\check{1}' = 1'$$

$$(\alpha; \beta)^\circ = \check{\beta}; \check{\alpha}$$

$$\alpha \subseteq \beta \implies \check{\alpha} \subseteq \check{\beta}$$

$$\alpha \subseteq \beta \implies \gamma; \alpha \subseteq \gamma; \beta$$

- $\Sigma$  is a ranked alphabet of function symbols, which may be monotonic or antimonotonic in their argument positions with respect to a pair of special relation symbols of  $S$ .

**3.2.2 Remark.** Notice the difference between sets  $S$ ,  $S^*$ , and  $\mathcal{S}^*$ .  $S$  is a finite set of special relations, the generators of the whole monoid structure. The domain of this structure is  $S^*$ , i.e., all the relations we obtain by composing and reversing the special relations in  $S$ , including an identity relation  $1'$ . Finally,  $\mathcal{S}^*$  denotes the monoid structure itself, with its neutral element, its multiplication operator, its involution operator, and its partial order.

**3.2.3 Polarity.** We are going to treat monotonicity and antimonotonicity as inherent features of the signature's function symbols, in the same sense as their arities. For this purpose we use the notion of *polarity*, inspired by Manna and Waldinger's work on special-relation rules (Manna and Waldinger, 1986; Manna and Waldinger, 1992). For example, let  $|x|$  denote the cardinality function applied to the set  $x$ . We have that for all  $x, y$ ,

$$x \subseteq y \implies |x| \leq |y|,$$

i.e., the cardinality function is monotonic in its unique argument position. We will say that its argument position has *positive* polarity (or is positive) with respect to  $(\subseteq, \leq)$ . In another example, let  $x \setminus y$  denote set difference between sets  $x$  and  $y$ . We have that for all  $x, y, z$ ,

$$x \subseteq y \implies z \setminus y \subseteq z \setminus x,$$

i.e., the set difference function is antimonotonic in its second argument. We will say that its second argument position has *negative* polarity (or is negative) with respect to  $(\subseteq, \subseteq)$ .

When we say that an argument position is positive (or negative) we do not exclude the possibility that it has both polarities. In general, when an argument



position has some polarity (either positive, negative, or both) we will just say that it is *polarized*.

Without loss of generality, in the rest of this paper we will mainly refer to positive polarities of argument position, since if a position has negative polarity with respect to a pair of relations, we express this polarity as a positive one in the following way: For any argument position  $i$  of any function symbol  $f$  in  $\Sigma$ ,

- the  $i$ -th argument position of  $f$  is negative with respect to  $(\alpha, \beta)$  if and only if it is positive with respect to  $(\check{\alpha}, \beta)$ ,
- the  $i$ -th argument position of  $f$  is negative with respect to  $(\alpha, \beta)$  if and only if it is positive with respect to  $(\alpha, \check{\beta})$ .

We also extend polarities with respect to composite relations and with respect to the identity relation in the following way: For any argument position  $i$  of any function symbol  $f$  in  $\Sigma$ ,

- if the  $i$ -th argument position of  $f$  is positive with respect to both,  $(\alpha, \beta)$  and  $(\alpha', \beta')$ , then it is also positive with respect to  $(\alpha; \alpha', \beta; \beta')$ ,
- the  $i$ -th argument position of  $f$  is positive with respect to  $(1', 1')$ .

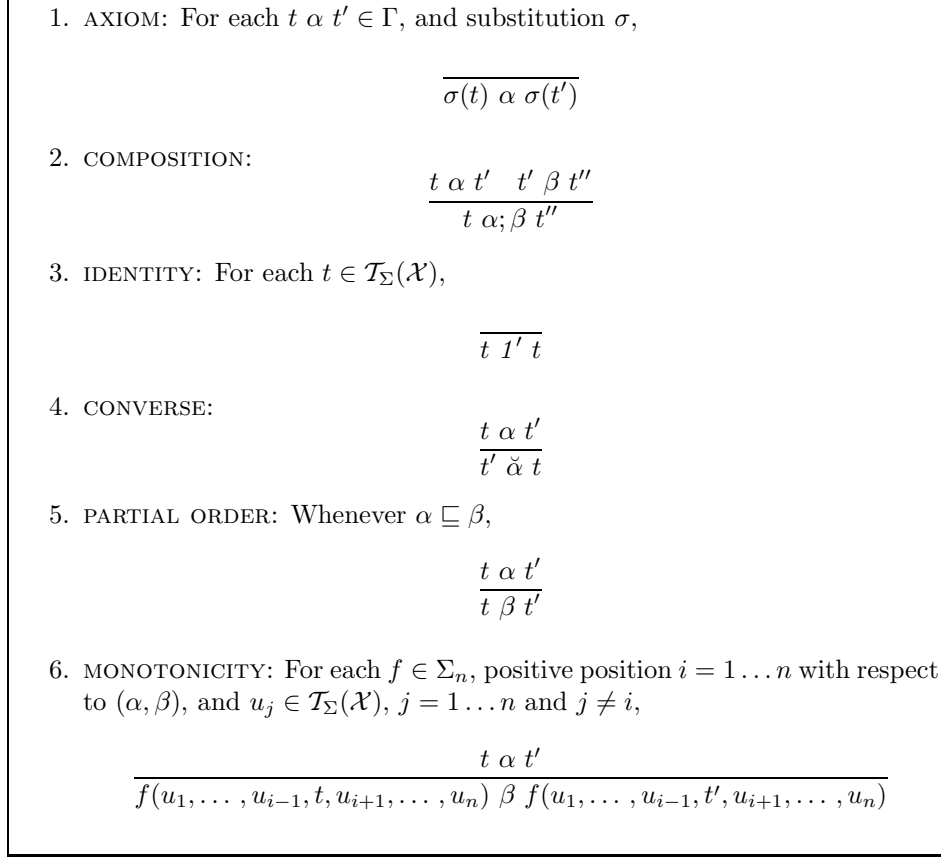
There exists an obvious relationship between polarities of argument positions and the partial order relation  $\sqsubseteq$  on relations. We need therefore to guarantee that the polarities given in a signature indeed satisfy this relationship. Thus we define a notion of correct signature:

**3.2.4 Definition.** A signature  $(S^*, \Sigma)$  is said to be *correct* if for any argument position  $i$  of any function symbol  $f$  in  $\Sigma$ , and any relations  $\alpha$ ,  $\beta$ , and  $\gamma$  in  $S$  we have that

- if  $\gamma \sqsubseteq \beta$  and the  $i$ -th argument position of  $f$  is positive with respect to  $(\alpha, \gamma)$ , then it is also positive with respect to  $(\alpha, \beta)$ ,
- if  $\alpha \sqsubseteq \gamma$  and the  $i$ -th argument position of  $f$  is positive with respect to  $(\gamma, \beta)$ , then it is also positive with respect to  $(\alpha, \beta)$ .

In the rest of this thesis we assume that all signatures are correct.

**3.2.5 Sentences.** Let  $\Omega = (S^*, \Sigma)$  be a signature. As usual,  $\mathcal{T}_\Sigma(\mathcal{X})$  denotes the set of first-order  $\Sigma$ -terms over a denumerable set  $\mathcal{X}$  of variables. We write  $t(x_1, \dots, x_m)$  for a term in  $\mathcal{T}_\Sigma(\{x_1, \dots, x_m\})$ , whenever we need to make its variables explicit. Sometimes we will abbreviate sequences of the form  $x_1, \dots, x_m$  with  $\bar{x}^m$ , and we will drop the superscript  $m$  when it is clear from context.  $\Omega$ -sentences are expressions  $s \alpha t$ , where  $s, t \in \mathcal{T}_\Sigma(\mathcal{X})$  and  $\alpha \in S^*$ . A *substitution*  $\sigma = \langle x_1 \mapsto t_1, \dots, x_n \mapsto t_n \rangle$  is a map from a finite subset of variables  $\{x_1, \dots, x_n\} \subseteq \mathcal{X}$  to terms, and can be uniquely extended to a map from terms to terms and from sentences to sentences. A *theory presentation* is a pair  $T = (\Omega, \Gamma)$ , where  $\Omega$  is a signature and  $\Gamma$  is a set of  $\Omega$ -sentences, also called *axioms*.



**Figure 3.2:** Inference rules of the logic of special relations

**3.2.6 Remark.** As already mentioned, in order to ease the following presentation, we will only consider atomic sentences, i.e., when sentences are unary positive Horn clauses. In section 3.4 we hint at how the logic can be generalized to deal with conditional sentences.

**3.2.7 Entailment.** Let  $T = (\Omega, \Gamma)$  be a theory presentation, we define the entailment  $\Gamma \vdash \varphi$  of an  $\Omega$ -sentence  $\varphi$  from a set of  $\Omega$ -sentences  $\Gamma$ , by means of the *inference rules* given in figure 3.2.

### 3.3 Semantics

We first start giving the intuitive set-valued semantics, and then move on towards a more abstract presentation within the realm of category theory, in order to embrace a large variety of models of specification, ranging from type algebras (Manca et al., 1990) to rewriting logic's ' $\mathcal{R}$ -systems' (Meseguer, 1992).

**3.3.1 Definition.** Given a signature  $(S^*, \Sigma)$ , a (set-valued) *interpretation* is a  $\Sigma$ -algebra  $\mathcal{A}$ , together with an assignment to each  $\alpha \in S^*$  of a set  $\llbracket \alpha \rrbracket \subseteq \mathcal{A} \times \mathcal{A}$ , such that for all  $\alpha, \beta \in S^*$ ,  $a, b \in \mathcal{A}$ :

1.  $(a, b) \in \llbracket \alpha; \beta \rrbracket$  if and only if there exists  $c \in \mathcal{A}$  such that  $(a, c) \in \llbracket \alpha \rrbracket$  and  $(c, b) \in \llbracket \beta \rrbracket$
2.  $(a, a) \in \llbracket I' \rrbracket$
3.  $(a, b) \in \llbracket \check{\alpha} \rrbracket$  if and only if  $(b, a) \in \llbracket \alpha \rrbracket$
4.  $\llbracket \alpha \rrbracket \subseteq \llbracket \beta \rrbracket$  if and only if  $\alpha \sqsubseteq \beta$

**3.3.2 Satisfaction.** We say that an interpretation satisfies a sentence  $t \alpha t'$  if and only if, for each assignment  $\rho : \mathcal{X} \rightarrow \mathcal{A}$ , we have  $(\llbracket t \rrbracket_\rho, \llbracket t' \rrbracket_\rho) \in \llbracket \alpha \rrbracket$ , where  $\llbracket \cdot \rrbracket_\rho$  is the unique  $\Sigma$ -homomorphism extending  $\rho$ .

**3.3.3 Remark.** Notice that we can make the logic deal with many-sorted signatures  $(M, S^*, \Sigma)$ , where  $M$  is a set of sort symbols, by defining interpretations as many-sorted  $(M, \Sigma)$ -algebras, and having the assignment to each heterogeneous relation  $\alpha : s_1 \leftrightarrow s_2 \in S^*$ —where  $s_1$  and  $s_2$  denote sorts—of a set  $\llbracket \alpha \rrbracket \subseteq \llbracket s_1 \rrbracket \times \llbracket s_2 \rrbracket$ .

**3.3.4 Categorical approach.** In order to capture a large variety of models of specification, we are going to endow the logic with a more abstract model theory in the realm of category theory.

By looking at the previous interpretations as structures valued in the category **Rel** of sets and relations, we may express models of a many-sorted theory presentation  $((M, S^*, \Sigma), \Gamma)$  without referring to set elements. We make abuse of notation by using the symbol of a function  $f : A \rightarrow B$  also as the symbol for the binary relation  $f \subseteq A \times B$  it denotes, such that  $(a, b) \in f$  if and only if  $b = f(a)$ , i.e.,  $f = \{(x, f(x)) \mid x \in A\}$ . This abuse can be extended by induction over the structure of a term in the obvious way.

An interpretation is given by an assignment to each sort symbol  $s$  of a **Rel**-object  $\llbracket s \rrbracket$ , together with an assignment to each function symbol  $f : s_1 \cdots s_n \rightarrow s \in \Sigma$  of a **Rel**-arrow

$$\llbracket f \rrbracket : \llbracket s_1 \rrbracket \times \cdots \times \llbracket s_n \rrbracket \rightarrow \llbracket s \rrbracket ,$$

which obviously has to be a function, and of each relation symbol  $\alpha : s_1 \leftrightarrow s_2 \in S^*$  of a **Rel**-arrow

$$\llbracket \alpha \rrbracket : \llbracket s_1 \rrbracket \rightarrow \llbracket s_2 \rrbracket ,$$

such that

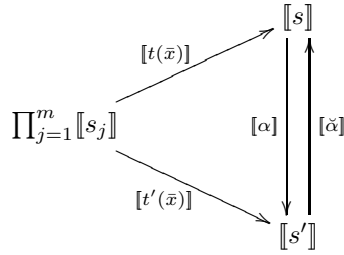
1.  $\llbracket \alpha; \beta \rrbracket = \llbracket \beta \rrbracket \cdot \llbracket \alpha \rrbracket$  (composition of arrows)
2.  $\llbracket I'_s \rrbracket = id_{\llbracket s \rrbracket}$  (identity arrows)

3.  $\llbracket \check{\alpha} \rrbracket = \llbracket \alpha \rrbracket^\circ$  (conversion of arrows)
4.  $\llbracket \alpha \rrbracket \subseteq \llbracket \beta \rrbracket$  if and only if  $\alpha \sqsubseteq \beta$

**3.3.5 Satisfaction of sentences.** What does the satisfaction of a sentence of the form

$$t(x_1, \dots, x_m) \alpha t'(x_1, \dots, x_m)$$

actually mean? We will dispose of arrows as pictured in the following diagram:



We have seen above that  $t(\bar{x}) \alpha t'(\bar{x})$  is true if and only if, for each assignment  $\rho : \mathcal{X} \rightarrow \mathcal{A}$ , we have  $(\llbracket t(\bar{x}) \rrbracket_\rho, \llbracket t'(\bar{x}) \rrbracket_\rho) \in [\alpha]$ . That is,  $\forall u \in \prod_{j=1}^m \llbracket s_j \rrbracket$

$$\begin{aligned} & (\llbracket t \rrbracket(u), \llbracket t' \rrbracket(u)) \in [\alpha] \\ \equiv & \\ & (u, \llbracket t' \rrbracket(u)) \in \{(u, v) \mid (\llbracket t \rrbracket(u), v) \in [\alpha]\} \\ \equiv & \\ & (u, \llbracket t' \rrbracket(u)) \in \{(u, v) \mid \exists w (u, w) \in \llbracket t \rrbracket \wedge (w, v) \in [\alpha]\} \end{aligned}$$

and, by definition of composition, this is equivalent to

$$\forall (u, v) \in \llbracket t' \rrbracket \quad (u, v) \in [\alpha] \cdot \llbracket t \rrbracket .$$

Analogously,  $\forall u \in \prod_{j=1}^m \llbracket s_j \rrbracket$

$$\begin{aligned} & (\llbracket t \rrbracket(u), \llbracket t' \rrbracket(u)) \in [\alpha] \\ \equiv & \\ & (\llbracket t \rrbracket(u), u) \in \{(w, u) \mid (w, \llbracket t' \rrbracket(u)) \in [\alpha]\} \\ \equiv & \\ & (u, \llbracket t \rrbracket(u)) \in \{(u, w) \mid (\llbracket t' \rrbracket(u), w) \in [\check{\alpha}]\} \\ \equiv & \\ & (u, \llbracket t \rrbracket(u)) \in \{(u, w) \mid \exists v (u, v) \in \llbracket t' \rrbracket \wedge (v, w) \in [\check{\alpha}]\} \end{aligned}$$

and, by definition of composition, this is equivalent to

$$\forall (u, w) \in \llbracket t \rrbracket \quad (u, w) \in [\check{\alpha}] \cdot \llbracket t' \rrbracket .$$

Consequently,

$$t(\bar{x}) \alpha t'(\bar{x}) \text{ is satisfied} \quad \equiv \quad \llbracket t(\bar{x}) \rrbracket \subseteq [\alpha] \cdot \llbracket t(\bar{x}) \rrbracket \quad \equiv \quad \llbracket t(\bar{x}) \rrbracket \subseteq [\check{\alpha}] \cdot \llbracket t'(\bar{x}) \rrbracket .$$

Since both inclusions are equivalent, it suffices to state only one of them.

**3.3.6 Monotonicity.** We have not seen yet how models capture monotonicity and antimonotonicity properties of function symbols. Given a function symbol  $f$  in  $\Sigma$ , we need that, when its  $i$ -th argument position is positive with respect to a pair of relations  $(\alpha, \beta)$ , then the following implication should hold, for terms  $t$  and  $t'$ :

$$(\llbracket t \rrbracket, \llbracket t' \rrbracket) \in \llbracket \alpha \rrbracket \Rightarrow (\llbracket f \rrbracket(\dots, \overset{i}{\llbracket t \rrbracket}, \dots), \llbracket f \rrbracket(\dots, \overset{i}{\llbracket t' \rrbracket}, \dots)) \in \llbracket \beta \rrbracket \quad (3.1)$$

Let us look first at the case of a monotone one-argument function  $f \in \Sigma_1$ . This means, that given two terms  $t, t' \in \mathcal{T}_\Sigma(\{x_1, \dots, x_m\})$

$$\llbracket t'(\bar{x}) \rrbracket \subseteq \llbracket \alpha \rrbracket \cdot \llbracket t(\bar{x}) \rrbracket \Rightarrow \llbracket f \rrbracket \cdot \llbracket t'(\bar{x}) \rrbracket \subseteq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket \cdot \llbracket t(\bar{x}) \rrbracket .$$

This is true whenever

$$\llbracket f \rrbracket \cdot \llbracket \alpha \rrbracket \subseteq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket . \quad (3.2)$$

The previous statement expresses actually how a model captures monotonicity of  $f$ . This is pictured in the following semi-commutative diagram:

$$\begin{array}{ccccc} & & \llbracket s \rrbracket & \xrightarrow{\llbracket f \rrbracket} & \llbracket s \rrbracket \\ & \nearrow \llbracket t(\bar{x}) \rrbracket & \downarrow \llbracket \alpha \rrbracket & \subseteq & \downarrow \llbracket \beta \rrbracket \\ \prod_{j=1}^m \llbracket s_j \rrbracket & \subseteq & & & \\ & \searrow \llbracket t'(\bar{x}) \rrbracket & \downarrow \llbracket f \rrbracket & & \downarrow \llbracket f \rrbracket \\ & & \llbracket s \rrbracket & \xrightarrow{\llbracket f \rrbracket} & \llbracket s \rrbracket \end{array}$$

Monotonicity in the  $i$ -th argument of a many-argument function  $f$  is semantically captured in a way analogous to the one-argument case, stated in inclusion (3.2). We proceed by currying  $f$  in all its arguments, but  $i$ -th one, obtaining  $\text{curry}_i(f)$ , and proceed as with a monotone one-argument function. We thus can express monotonicity category-theoretically by saying that

$$\text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \llbracket \alpha \rrbracket) \subseteq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket , \quad (3.3)$$

where  $\text{curry}_i(\llbracket f \rrbracket) : \prod_{j \neq i} \llbracket s_j \rrbracket \rightarrow \llbracket s \rrbracket^{[s_i]}$  is the unique arrow such that

$$\text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \text{id}_{[s_i]}) = \llbracket f \rrbracket .$$

This is pictured in the following semi-commutative diagram:

$$\begin{array}{ccc} \prod_{j=1}^m \llbracket s_j \rrbracket & \xrightarrow{\llbracket f \rrbracket} & \llbracket s \rrbracket \\ \downarrow \text{curry}_i(\llbracket f \rrbracket) \times \llbracket \alpha \rrbracket & \subseteq & \downarrow \llbracket \beta \rrbracket \\ \llbracket s \rrbracket^{[s_i]} \times \llbracket s_i \rrbracket & \xrightarrow{\text{apply}} & \llbracket s \rrbracket \end{array}$$

To prove that inclusion (3.3) indeed captures monotonicity, we reason

$$\begin{aligned}
& \llbracket t' \rrbracket \subseteq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \\
\Rightarrow & \quad \{ \llbracket t_1 \rrbracket, \dots, \llbracket t_{i-1} \rrbracket, \llbracket t_{i+1} \rrbracket, \dots, \llbracket t_n \rrbracket \text{ are total functions} \} \\
& \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t' \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \subseteq \langle \llbracket t_1 \rrbracket, \dots, \llbracket \alpha \rrbracket \cdot \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \\
\Rightarrow & \quad \{ \text{monotonicity of composition} \} \\
& \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t' \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \subseteq \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket \alpha \rrbracket \cdot \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle.
\end{aligned}$$

Furthermore, since

$$\begin{aligned}
& \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket \alpha \rrbracket \cdot \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \\
= & \quad \{ \text{exponentials} \} \\
& \text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \text{id}_{\llbracket s_i \rrbracket}) \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket \alpha \rrbracket \cdot \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \\
= & \quad \{ \text{products}^2 \} \\
& \text{apply} \cdot \langle \text{curry}_i(\llbracket f \rrbracket) \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket t_{i-1} \rrbracket, \llbracket t_{i+1} \rrbracket, \dots, \llbracket t_n \rrbracket \rangle, \llbracket \alpha \rrbracket \cdot \overset{i)}{\llbracket t \rrbracket} \rangle \\
= & \quad \{ \text{products}^2 \} \\
& \text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \llbracket \alpha \rrbracket) \cdot \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \\
\subseteq & \quad \{ \text{inclusion (3.3)} \} \\
& \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle,
\end{aligned}$$

it follows that

$$\llbracket t' \rrbracket \subseteq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \Rightarrow \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t' \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle \subseteq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \overset{i)}{\llbracket t \rrbracket}, \dots, \llbracket t_n \rrbracket \rangle$$

which is the category-theoretical equivalent of implication (3.1).

We did the previous discussion for set-valued interpretations —structures valued in the particular category **Rel** of sets and relations. The models, though, can be generalized to valuations in categories in general, provided they have the additional structure we need. For this reason, we use notions borrowed from the category theory of relations, *allegories*, thoroughly studied in (Freyd and Scedrov, 1990).

**3.3.7 Allegories.** Allegories are to binary relations between sets as categories are to functions between sets. They are, therefore, the generalization of category **Rel** to any kind of object. We already mentioned in the introduction to this chapter that, for computational purposes, we chose not to use the whole expressive power provided by relation algebra. Consequently, we will not use all the expressiveness provided by the theory of allegories. Therefore, we define *quasi-allegories*, a slightly simplified definition of allegories than the original one

---

<sup>2</sup>We use the following property of products:  $(f \times g) \cdot \langle p, q \rangle = \langle f \cdot p, g \cdot q \rangle$

given by Freyd and Scedrov (1990), but whose main intuitions remain essentially the the same. See also (Bird and de Moor, 1997) for a short introduction to allegories.

**3.3.8 Definition.** A *quasi-allegory* is a category together with

1. a unary operation on arrows, assigning to each arrow  $\alpha : S \rightarrow T$  its converse arrow  $\alpha^\circ : T \rightarrow S$  such that
  - $(\alpha^\circ)^\circ = \alpha$
  - for each object  $S$ ,  $id_S^\circ = id_S$
  - given arrows  $\alpha : S \rightarrow T$  and  $\beta : T \rightarrow U$ ,  $(\beta \cdot \alpha)^\circ = \alpha^\circ \cdot \beta^\circ$
2. a partial order  $\leq$  over arrows such that, whenever for two parallel arrows  $\alpha, \beta : S \rightarrow T$  we have that  $\alpha \leq \beta$ , then
  - $\alpha^\circ \leq \beta^\circ$
  - given an arrow  $\gamma : T \rightarrow U$ ,  $\gamma \cdot \alpha \leq \gamma \cdot \beta$

**3.3.9 Definition.** We say that an arrow  $f : S \rightarrow T$  in a quasi-allegory

- is *entire*, if  $id_S \leq f^\circ \cdot f$ ,
- is *simple*, if  $f \cdot f^\circ \leq id_T$ ,
- is a *function*, when it is entire and simple.

For a quasi-allegory  $\mathbf{A}$ , we denote its subcategory of functions by  $Fun(\mathbf{A})$ . For example,  $Fun(\mathbf{Rel})$  is the category **Set**.

**3.3.10 Models.** We now dispose of all the ingredients to define the models in the logic of special relations within the abstract theory of allegories:

**3.3.11 Definition.** Given a theory presentation  $T = (\Omega, \Gamma)$ , with many-sorted signature  $\Omega = (M, S^*, \Sigma)$ , a *model* of  $T$  consists of a quasi-allegory  $\mathbf{A}$ , for which its subcategory of functions  $Fun(\mathbf{A})$  is Cartesian closed, together with an assignment

1. to each sort  $s \in M$  of an object  $\llbracket s \rrbracket$
2. to each  $f : s_1 \cdots s_n \rightarrow s \in \Sigma$  of a function  $\llbracket f \rrbracket : \llbracket s_1 \rrbracket \times \cdots \times \llbracket s_n \rrbracket \rightarrow \llbracket s \rrbracket$
3. to each  $\alpha : s_1 \leftrightarrow s_2 \in S^*$  of an arrow  $\llbracket \alpha \rrbracket : \llbracket s_1 \rrbracket \rightarrow \llbracket s_2 \rrbracket$

such that

1.  $\llbracket \alpha; \beta \rrbracket = \llbracket \beta \rrbracket \cdot \llbracket \alpha \rrbracket$
2.  $\llbracket 1'_s \rrbracket = id_{\llbracket s \rrbracket}$

3.  $\llbracket \check{\alpha} \rrbracket = \llbracket \alpha \rrbracket^\circ$
4.  $\llbracket \alpha \rrbracket \leq \llbracket \beta \rrbracket$  whenever  $\alpha \sqsubseteq \beta$

and, if the  $i$ -th position of  $f$  is positive with respect to  $(\alpha, \beta)$ , then

$$\text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \llbracket \alpha \rrbracket) \leq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket$$

where  $\text{curry}_i(\llbracket f \rrbracket) : \prod_{j \neq i} \llbracket s_j \rrbracket \rightarrow \llbracket s \rrbracket^{[s_i]}$  is the unique arrow such that

$$\text{apply} \cdot (\text{curry}_i(\llbracket f \rrbracket) \times \text{id}_{[s_i]}) = \llbracket f \rrbracket \quad ;$$

and, for each sentence  $t \alpha t'$  in  $\Gamma$ , we have that

$$\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket^3 \quad .$$

We write  $\Gamma \models t \alpha t'$ .

It suffices to require only one inequation,  $\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket$ , because the other one,  $\llbracket t \rrbracket \leq \llbracket \alpha \rrbracket^\circ \cdot \llbracket t' \rrbracket$ , is equivalent:

**3.3.12 Proposition.** *Given two functions  $f : A \rightarrow B$  and  $g : A \rightarrow C$  and an arrow  $\alpha : B \rightarrow C$ , then  $g \leq \alpha \cdot f$  if and only if  $f \leq \alpha^\circ \cdot g$ .*

PROOF:

$$\begin{aligned} g \leq \alpha \cdot f &\Rightarrow g^\circ \cdot g \leq g^\circ \cdot \alpha \cdot f \\ &\Rightarrow \text{id}_A \leq g^\circ \cdot \alpha \cdot f \\ &\Rightarrow f^\circ \leq g^\circ \cdot \alpha \cdot f \cdot f^\circ \\ &\Rightarrow f^\circ \leq g^\circ \cdot \alpha \\ &\Rightarrow f \leq \alpha^\circ \cdot g \end{aligned}$$

$$\begin{aligned} f \leq \alpha^\circ \cdot g &\Rightarrow f^\circ \cdot f \leq f^\circ \cdot \alpha^\circ \cdot g \\ &\Rightarrow \text{id}_A \leq f^\circ \cdot \alpha^\circ \cdot g \\ &\Rightarrow g^\circ \leq f^\circ \cdot \alpha^\circ \cdot g \cdot g^\circ \\ &\Rightarrow g^\circ \leq f^\circ \cdot \alpha^\circ \\ &\Rightarrow g \leq \alpha \cdot f \end{aligned}$$

□

**3.3.13 Soundness and completeness.** We now proof that the inference system of figure 3.2 is sound with respect to the semantics we have given to the syntax of our logic. Unfortunately, we do not have yet a proof that the logic is complete. We suspect that probably we have endowed the logic with too expressive a semantics, in comparison to its restricted syntactic language, and that we may have to polish our definition of quasi-allegory in order to get

---

<sup>3</sup>The value of a term in a model is defined inductively over the structure of the term by following the usual conventions of categorical logic, namely as a morphism from the product object determined by the sorts of the variables, to the object determined the value of the term.



also a complete logic. We believe that completeness will be achieved, once we have established the relationship of our logic of special relations with Bruno, Gadducci, and Montanari's closely related tile logic (Gadducci and Montanari, 1996; Bruni et al., 1998) (see also the discussion in 9.1.4).

**3.3.14 Theorem.** *The logic is sound: Given a theory presentation  $T = (\Omega, \Gamma)$  and an  $\Omega$ -sentence  $t \alpha t'$ , then if  $\Gamma \vdash t \alpha t'$ , every model  $\mathbf{A}$  of  $T$  satisfies  $t \alpha t'$ , i.e.,  $\Gamma \models t \alpha t'$ .*

PROOF: We proof it, by showing that each inference rule of figure 3.2 is sound:

1. AXIOM: By monotonicity of composition,

$$\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \quad \text{implies} \quad \llbracket t' \rrbracket \cdot a \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \cdot a ,$$

for every arrow  $a$  with codomain the domain of  $\llbracket t \rrbracket$  and  $\llbracket t' \rrbracket$ .

2. COMPOSITION: By monotonicity of composition and transitivity of  $\leq$ ,

$$\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \quad \text{and} \quad \llbracket t'' \rrbracket \leq \llbracket \beta \rrbracket \cdot \llbracket t' \rrbracket \quad \text{imply} \quad \llbracket t'' \rrbracket \leq \llbracket \beta \rrbracket \cdot \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket$$

3. IDENTITY: For each  $t \in \mathcal{T}_\Sigma(\mathcal{X})$ ,

$$\llbracket t \rrbracket \leq id \cdot \llbracket t \rrbracket$$

4. CONVERSE: By proposition 3.3.12,

$$\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \quad \text{implies} \quad \llbracket t \rrbracket \leq \llbracket \alpha \rrbracket^\circ \cdot \llbracket t' \rrbracket$$

5. PARTIAL ORDER: By monotonicity of composition and transitivity of  $\leq$ ,

$$\llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \quad \text{and} \quad \llbracket \alpha \rrbracket \leq \llbracket \beta \rrbracket \quad \text{imply} \quad \llbracket t' \rrbracket \leq \llbracket \beta \rrbracket \cdot \llbracket t \rrbracket$$

6. MONOTONICITY: Reasoning as in 3.3.6,

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \alpha \rrbracket) \leq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket \quad \text{and} \quad \llbracket t' \rrbracket \leq \llbracket \alpha \rrbracket \cdot \llbracket t \rrbracket \quad \text{imply}$$

$$\llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket t' \rrbracket, \dots, \llbracket t_n \rrbracket \rangle \leq \llbracket \beta \rrbracket \cdot \llbracket f \rrbracket \cdot \langle \llbracket t_1 \rrbracket, \dots, \llbracket t \rrbracket, \dots, \llbracket t_n \rrbracket \rangle^4$$

□

## 3.4 Conditional Sentences

In the introduction to this chapter, we mentioned that we want our logic of special relations, in general, to deal with positive Horn formulae, but that, in this thesis, we will only consider the unconditional fragment of the logic, i.e., when sentences are positive atoms. Here, we briefly sketch how the logic is to be extended to cope with conditional sentences.

---

<sup>4</sup> $\langle \dots \rangle$  denotes the mediating arrow.

**3.4.1 Replacement.** If we want to deal with conditional sentences of the form

$$t \alpha t' \text{ if } u_1 \beta_1 v_1 \wedge \cdots \wedge u_n \beta_n v_n ,$$

where  $t, t', u_i$ , and  $v_i$  are terms in  $\mathcal{T}_\Sigma(\mathcal{X})$ , and  $\alpha$  and  $\beta_i$  are relations in  $S^*$ . we will need to reformulate the AXIOM deduction rule of our entailment system given in figure 3.2 to the REPLACEMENT deduction rule as follows:

REPLACEMENT:

For each sentence  $(t \alpha t' \text{ if } u_1 \beta_1 v_1 \wedge \cdots \wedge u_n \beta_n v_n) \in \Gamma$ , and for each substitution  $\sigma : \mathcal{X} \mapsto \mathcal{T}_\Sigma(\mathcal{X})$

$$\frac{\sigma(u_1) \beta_1 \sigma(v_1) \cdots \sigma(u_n) \beta_n \sigma(v_n)}{\sigma(t) \alpha \sigma(t')}$$

**3.4.2 Includer.** With respect to the semantics, and given a conditional sentence of the form

$$t(\bar{x}) \alpha t'(\bar{x}) \text{ if } u_1(\bar{x}) \beta_1 v_1(\bar{x}) \wedge \cdots \wedge u_n(\bar{x}) \beta_n v_n(\bar{x}) ,$$

we will need the models to satisfy  $t(\bar{x}) \alpha t'(\bar{x})$  whenever all  $u_i(\bar{x}) \beta_i v_i(\bar{x})$  are satisfied. In other words, for all functions  $f$  with codomain the domain of  $\llbracket u_i(\bar{x}) \rrbracket$  and  $\llbracket v_i(\bar{x}) \rrbracket$ , such that, whenever, for all  $i$ ,

$$\llbracket v_i(\bar{x}) \rrbracket \cdot f \leq \llbracket \beta_i \rrbracket \cdot \llbracket u_i(\bar{x}) \rrbracket \cdot f ,$$

then we have also that

$$\llbracket t'(\bar{x}) \rrbracket \cdot f \leq \llbracket \alpha \rrbracket \cdot \llbracket t(\bar{x}) \rrbracket \cdot f .$$

This is expressed categorically using a variant of the notion of *equalizer* that we call *includer*.

**3.4.3 Definition.** Given arrows  $\alpha, \beta : S \rightarrow T$  of a quasi-allegory, an *includer* of  $\alpha$  and  $\beta$  is an object  $I$  together with a function

$$\iota : I \rightarrow S$$

such that

1.  $\alpha \cdot \iota \leq \beta \cdot \iota$
2. given an arrow  $\kappa : K \rightarrow S$  such that  $\alpha \cdot \kappa \leq \beta \cdot \kappa$  there exists a unique arrow  $\lambda : K \rightarrow I$  such that  $\kappa = \iota \cdot \lambda$ .

**3.4.4 Example.** Let  $\alpha$  and  $\beta$  be two relations in **Rel** with common domain  $S$  and codomain  $T$ . Let  $I$  be the subset of  $S$  constructed as follows:

$$I = \{x \in S \mid \forall y \in T \quad (x, y) \in \alpha \Rightarrow (x, y) \in \beta\}$$

Then the inclusion function  $i : I \rightarrow S$ , which maps each element  $x \in I$  to the same  $x$  considered as an element of  $S$ , is an includer of  $\alpha$  and  $\beta$ .

**3.4.5 Satisfaction.** The partial order  $\leq$  over arrows of a quasi-allegory can also be seen as a collection of 2-cells of a 2-category. Thus, an includer is conceptually close to a subequalizer, originally introduced in (Lambek, 1979), generalizing equalizers of two functors requiring a natural transformation between functors instead of an identity of functors. We will use includers in the same sense as subequalizers are used in the semantics of conditional rewriting logic (Meseguer, 1992), namely as includers of *families* of pairs of arrows, such that a quasi-allegory  $\mathbf{A}$  will be a model of a given theory presentation involving conditional sentences, if for each sentence

$$t(\bar{x}^m) \alpha t'(\bar{x}^m) \text{ if } u_1(\bar{x}^m) \beta_1 v_1(\bar{x}^m) \wedge \cdots \wedge u_n(\bar{x}^m) \beta_n v_n(\bar{x}^m)$$

in  $\Gamma$ , we have

$$\llbracket t'(\bar{x}^m) \rrbracket \cdot \iota \leq \llbracket \alpha \rrbracket \cdot \llbracket t(\bar{x}^m) \rrbracket \cdot \iota.$$

where  $\iota$  is the includer of the family of arrows  $(\llbracket v_1(\bar{x}^m) \rrbracket, \llbracket \beta_i \rrbracket \cdot \llbracket u_1(\bar{x}^m) \rrbracket)_{1 \leq i \leq n}$ .

## 3.5 A Semantic Framework

We have laid down the required framework to explore the assumptions put forth in chapter 1. The next chapter introduces the general notion of term rewriting that actually motivated us to define this logic of special relations. The reason of many of the choices we have taken in developing the logic will become much clearer once we have worked through its rewriting-based proof calculus; and we will show the need of the abstract semantics based on quasi-allegories when demonstrating how, by means of conservative maps of logics, the logic indeed captures several specification frameworks, such as membership equational logic, rewriting logic, and specifications with set relations. Therefore, we will need to postpone the full understanding of the logic presented in this chapter, until we reach chapter 5, where we will shed light on its role as semantic framework.



## Chapter 4

# Rewriting Beyond Equality

The development of the general rewriting technique presented in this chapter originates from our belief that the basic properties of special relations can be nicely captured by term rewriting. We like to look at term rewriting from the following perspective:

1. rewriting replaces a term by another by applying an instance of a given rewrite rule;
2. rewriting profits of the term structure in order to replace only certain subterms;
3. rewriting is actually the result of a series of successive replacements.

When we rewrite a term, we usually replace equals by equals, but nothing hinders us to replace smaller by bigger, or supersets by subsets. In principle we can replace one term by another related by any binary relation. To rewrite a term we only need to keep track of the binary relation between the source and the target term. For replacing only part of a term —a subterm— we do not need to be able to generalize this operation to *any* subterm. When we replace equals by equals it has sense to replace any subterm of a term; but if other binary relations are involved, maybe only certain subterms make sense to be replaced. If our notion of term rewriting is to proceed like that, we will need some additional mechanism to control the replacement of subterms. Since we eventually perform several replacements one after the other, we will need to know how the source term and the target term are related, once all replacements have been performed.

In the previous chapter, we have tailored a logic of special relation to be able to handle with all these aspects, namely

1. arbitrary binary relations, and their reversions, for relating terms,
2. (anti)monotonicity of certain argument positions of function symbols by means of polarities, and

### 3. general composition of relations.

In this chapter, we look at theories in this logic of special relations as term rewriting systems. Therefore, we redefine the notions of rewrite relation and rewrite proof in order to cover our general approach, and we further explore the conditions term rewriting systems have to satisfy, in order to provide decision procedures for theoremhood. This brings us to work through the well-known notions of *termination* and *confluence*, and to see how they translate to our framework. The difficult terrain is entered when effective completion of non-confluent systems is desired. Recall the difficulties we already encountered in surveying the bi-rewriting technique in chapter 2. Indeed, abandoning the requirement of special relations to have nice properties like transitivity, symmetry and monotonicity makes it very difficult to deal with effective completion... but not impossible. We therefore analyze the conditions that signatures or theories have to satisfy, so that completion is tractable, and we find that the concept of polarity is very useful for this enterprise. Indeed, polarity becomes the key issue for controlling term rewriting in such general setting.

## 4.1 Rewriting Along Binary Relations

**4.1.1 Polarity of subterm positions.** Given a term  $t$ , let  $t|_p$  denote the subterm occurring at position  $p$ . When this occurrence is replaced with term  $s$ , we denote that by  $t[s]_p$ . The polarity of argument positions of functions can be easily extended to subterm positions  $p$  within a term  $t$  as follows:

### 4.1.2 Definition.

1. Position  $i$  in a term  $f(t_1, \dots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})$  is *positive* with respect to a pair of relations  $(\alpha, \beta)$ , if and only if the  $i$ -th argument position of  $f$  is positive with respect to  $(\alpha, \beta)$ .
2. For every term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$ , subterm positions  $p$  and  $q$ , and relations  $\alpha, \beta \in S^*$ ,  $p \cdot q$  in  $u$  is *positive* with respect to  $(\alpha, \beta)$  if and only if there exists a relation  $\gamma \in S^*$ , such that the polarity of  $q$  in  $u|_p$  with respect to  $(\alpha, \gamma)$  and the polarity of  $p$  in  $u$  with respect to  $(\gamma, \beta)$  are both positive.

**4.1.3 Remark.** Obviously this extension captures all the negative polarities through the relation existing between positive and negative polarities, as discussed in 3.2.3.

**4.1.4 Example.** Let  $(S^*, \Sigma)$  be a signature of the logic of special relations, with  $S = \{\subseteq, \leq\}$  and  $\Sigma = \{0, a, b, |_, _ + _, _ \setminus _\}$ <sup>1</sup> (three constants, one unary function symbol, and two binary function symbols), such that the following polarities hold:

---

<sup>1</sup>The symbol  $_$  is a placeholder for arguments.

- the first (and unique) argument position of  $|-|$  is positive with respect to  $(\subseteq, \leq)$
- the first and second argument positions of  $- + -$  are both positive with respect to  $(\leq, \leq)$
- the first argument position of  $- \setminus -$  is positive with respect to  $(\subseteq, \subseteq)$
- the second argument position of  $- \setminus -$  is negative with respect to  $(\subseteq, \subseteq)$ , and therefore it is positive with respect to  $(\subseteq, \supseteq)$ , or with respect to  $(\supseteq, \subseteq)$ <sup>2</sup>

In this case we have that the subterm position of  $b$  in term  $|a \setminus b| + 0$ , namely  $1 \cdot 1 \cdot 2$ , is positive with respect to  $(\supseteq, \leq)$ , because

1. the subterm position of  $b$  in  $a \setminus b$  (i.e.,  $2$ ) is positive with respect to  $(\supseteq, \subseteq)$ ,
2. the subterm position of  $a \setminus b$  in  $|a \setminus b|$  (i.e.,  $1$ ) is positive with respect to  $(\subseteq, \leq)$ , and
3. the subterm position of  $|a \setminus b|$  in  $|a \setminus b| + 0$  (i.e.,  $1$ ) is positive with respect to  $(\leq, \leq)$ .

**4.1.5 Rewrite rules.** The term rewriting approach to theorem proving in equational logic is based on using equations of a given theory presentation as rewrite rules, by imposing a specific directionality. Analogously, we may prove theorems of a given theory in our logic of special relations by considering its atomic formulae as rewrite rules, too. We do this, either considering an atomic formula  $s \alpha t$  as a rule from left to right, which we write  $s \xrightarrow{\alpha} t$ , or else from right to left, which we write  $s \xleftarrow{\alpha} t$ . Since sentences  $s \alpha t$  and  $t \check{\alpha} s$  are equivalent, we may also write  $t \xleftarrow{\check{\alpha}} s$  and  $t \xrightarrow{\check{\alpha}} s$ , respectively.

**4.1.6 Term rewriting system.** If, given a *theory presentation*  $((S^*, \Sigma), \Gamma)$  we interpret the axioms in  $\Gamma$  as rewrite rules in the sense explained in 4.1.5, then we may call  $\Gamma$  a *term rewriting system*, generalizing so the standard notion of term rewriting system (where rewrite rules are actual equations). Consequently, we redefine the notion of term rewriting as follows:

**4.1.7 Definition.** Given a term rewriting system  $\Gamma$ , a rewrite rule  $l \xrightarrow{\alpha} r$  in  $\Gamma$ , and a term  $s$ , we say that  $s$  *rewrites along  $\gamma$*  to  $t$ , written  $s \xrightarrow[\gamma, \Gamma]{\alpha} t$ , if there exist a relation  $\gamma$  in  $S^*$  and a substitution  $\sigma$ , such that  $\sigma(l) = s|_p$  for a subterm position  $p$  that is positive with respect to  $(\alpha, \gamma)$ , and  $t = s[\sigma(r)]_p$ . We call  $s \xrightarrow[\gamma, \Gamma]{\alpha} t$  also a *rewrite step*.

---

<sup>2</sup> $\supseteq$  is the reversion of  $\subseteq$ , i.e.,  $\supseteq \equiv \check{\subseteq}$ .

**4.1.8 Example.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation with  $(\mathcal{S}^*, \Sigma)$  the signature of example 4.1.4 and  $\Gamma = \{x \setminus y \subseteq x\}$ , such that we interpret the axiom as a rewrite rule

$$x \setminus y \xrightarrow[\subseteq]{} x .$$

We have that

$$|a \setminus b| + 0 \xrightarrow[\leq, \Gamma]{} |a| + 0 ,$$

because  $x \setminus y$  matches  $a \setminus b$ , and its subterm position  $1 \cdot 1$  in  $|a \setminus b| + 0$  is positive with respect to  $(\subseteq, \leq)$ .

**4.1.9 Notation.** In general, we will write  $s \xrightarrow[\Gamma]{} t$ , if there exists some relation  $\gamma$  in  $S^*$ , such that  $s \xrightarrow[\gamma, \Gamma]{} t$ , i.e.,

$$\xrightarrow[\Gamma]{} = \bigcup_{\gamma \in S^*} \xrightarrow[\gamma, \Gamma]{} .$$

**4.1.10 Remark.** Deviating from its standard definition, we will call  $\xrightarrow[\Gamma]{} a$  a *rewrite relation*. Usually a rewrite relation is defined as a binary relation over terms that is closed under both, context application (the ‘replacement property’) and substitutions (the ‘fully invariant property’). Our redefinition of rewrite relation differs from the standard one in that, according to definition 4.1.7,  $\xrightarrow[\Gamma]{} a$  satisfies a weaker ‘replacement property’, namely that the relation is closed under context application *only on positively polarized positions* with respect to a pair of relations.

**4.1.11 Notation.** Given rewrite relation  $\xrightarrow[\Gamma]{} a$  induced by term rewriting system  $\Gamma$ , we write  $\xrightarrow[\Gamma]^+ a$  and  $\xrightarrow[\Gamma]^* a$  for its transitive and reflexive-transitive closures, respectively. In particular, we write  $s \xrightarrow[\gamma, \Gamma]^+ t$  if there exist terms  $s_0, \dots, s_n \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\alpha_1, \dots, \alpha_n \in S^*$ ,  $n > 0$ , such that

$$s = s_0 \xrightarrow[\alpha_1, \Gamma]{} s_1 \xrightarrow[\alpha_2, \Gamma]{} s_2 \xrightarrow[\alpha_3, \Gamma]{} \dots \xrightarrow[\alpha_n, \Gamma]{} s_n = t \quad \text{and} \quad \alpha_1; \dots; \alpha_n \sqsubseteq \gamma ,$$

and we write  $s \xrightarrow[\gamma, \Gamma]{} t$  when  $n \geq 0$ . It is obvious that

$$\begin{aligned} \xrightarrow[\Gamma]^+ &= \bigcup_{\gamma \in S^*} \xrightarrow[\gamma, \Gamma]^+ \\ \xrightarrow[\Gamma]^* &= \bigcup_{\gamma \in S^*} \xrightarrow[\gamma, \Gamma]^* . \end{aligned}$$



We write  $s \xleftrightarrow[\gamma, \Gamma]{\quad} t$  if either  $s \xrightarrow[\gamma, \Gamma]{\quad} t$  or else  $s \xleftarrow[\gamma, \Gamma]{\quad} t$ , i.e.,

$$\xleftrightarrow[\gamma, \Gamma]{\quad} = \left( \xleftarrow[\gamma, \Gamma]{\quad} \cup \xrightarrow[\gamma, \Gamma]{\quad} \right) .$$

And, as before,

$$\xleftrightarrow[\Gamma]{\quad} = \bigcup_{\gamma \in S^*} \xleftrightarrow[\gamma, \Gamma]{\quad} ,$$

which is the symmetric closure of  $\xrightarrow[\Gamma]{\quad}$ . Analogously, we write  $s \xleftrightarrow[\gamma, \Gamma]{+} t$  if there exist terms  $s_0, \dots, s_n \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\alpha_1, \dots, \alpha_n \in S^*$ ,  $n > 0$ , such that

$$s = s_0 \xleftrightarrow[\alpha_1, \Gamma]{\quad} s_1 \xleftrightarrow[\alpha_2, \Gamma]{\quad} s_2 \xleftrightarrow[\alpha_3, \Gamma]{\quad} \cdots \xleftrightarrow[\alpha_n, \Gamma]{\quad} s_n = t \quad \text{and} \quad \alpha_1; \cdots; \alpha_n \sqsubseteq \gamma ,$$

and we write  $s \xleftrightarrow[\gamma, \Gamma]{*} t$  when  $n \geq 0$ . It follows that

$$\begin{aligned} \xleftrightarrow[\Gamma]{+} &= \bigcup_{\gamma \in S^*} \xleftrightarrow[\gamma, \Gamma]{+} \\ \xleftrightarrow[\Gamma]{*} &= \bigcup_{\gamma \in S^*} \xleftrightarrow[\gamma, \Gamma]{*} , \end{aligned}$$

are the transitive and reflexive-transitive closures of  $\xleftrightarrow[\Gamma]{\quad}$ , respectively.

We will drop the subscript  $\Gamma$  if the term rewriting system is clear from context.

## 4.2 Rewrite Proofs

We are interested in using the general notion of term rewriting along relations introduced in section 4.1 for proving if an atomic formula is or is not a theorem of a given theory in the logic of special relations. We know from standard equational reasoning that term rewriting systems need to be *convergent* in order to provide decision algorithms for the equational theory they embed. In this section, we are going to analyze how these properties translate to our general notion of term rewriting system introduced in 4.1.6. Of course, due to the additional generality and expressiveness of the logic of special relations, we expect the required analysis to be much more subtle than in the equational case.

Let us first see how the notion of proof by term rewriting is connected to the notion of entailment given in 3.2.7.

**4.2.1 Definition.** Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation in the logic of special relations. A *proof* of  $\Gamma \vdash s \gamma t$  is the sequence of rewrite steps in  $\Gamma$ —now considered as a term rewriting system— of the form

$$s = s_0 \xleftrightarrow[\alpha_1]{\quad} s_1 \xleftrightarrow[\alpha_2]{\quad} \cdots \xleftrightarrow[\alpha_n]{\quad} s_n = t$$

$n \geq 0$ , such that  $\alpha_1; \dots; \alpha_n \sqsubseteq \gamma$ , i.e.,  $s \xrightarrow[\gamma]{*} t$ . Recall that, when  $n = 0$ ,  $\alpha_1; \dots; \alpha_n = 1'$ .

It follows from the definition of the entailment relation  $\vdash$  given by the inference rules of figure 3.2 that there is a proof—as defined in 4.2.1—for every theorem of a given theory.

**4.2.2 Proposition.** *For any theory presentation  $((S^*, \Sigma), \Gamma)$ , terms  $s$  and  $t$  in  $\mathcal{T}_\Sigma(\mathcal{X})$ , and relation  $\gamma$  in  $S^*$ ,  $\Gamma \vdash s \gamma t$  if and only if  $s \xrightarrow[\gamma, \Gamma]{*} t$ .*

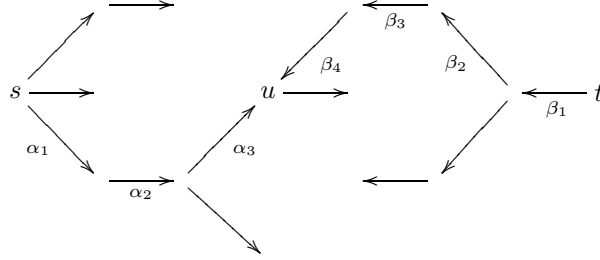
Since we are interested in exploiting term rewriting along relations in a computational way, we will look for sufficient conditions on term rewriting system  $\Gamma$ , such that every theorem of  $\Gamma$  can also be proved by a rewrite proof:

**4.2.3 Definition.** Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation in our logic of special relations. A *rewrite proof* of  $\Gamma \vdash s \gamma t$  is a proof of the particular form

$$s \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} u \xleftarrow{\beta_m} \dots \xleftarrow{\beta_1} t$$

$n, m \geq 0$ , such that  $\alpha_1; \dots; \alpha_n; \beta_m; \dots; \beta_1 \sqsubseteq \gamma$ , i.e.,  $s \xrightarrow[\alpha_1; \dots; \alpha_n]{*} u \xleftarrow[\beta_m; \dots; \beta_1]{*} t$ .

If every atom that is provable by a sequence of rewrite steps is also provable by a rewrite proof, then a decision procedure can be defined in a quite straightforward way: In order to prove  $\Gamma \vdash s \gamma t$ , we explore all the rewritings starting at terms  $s$  and  $t$  respectively until a common term  $u$  is reached and the composition of relations along which the rewritings took place is below relation  $\gamma$ :



Before we describe the exact procedure, we need to prove the following proposition, in order to define the notion of *reachability set*.

**4.2.4 Proposition.** *Let  $\check{S} = \{\check{\alpha} \mid \alpha \in S\}$ . For every relation  $\alpha \in S^*$ , there exist  $\alpha_1, \dots, \alpha_n \in S \cup \check{S}$ ,  $n \geq 0$ , such that  $\alpha = \alpha_1; \dots; \alpha_n$ .*

PROOF: By definition, since  $S^*$  is the domain of a free monoid with an anti-involution (see 3.2.1).  $\square$

**4.2.5 Definition.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation in our logic of special relation, and let  $s$  be a term in  $\mathcal{T}_\Sigma(\mathcal{X})$ . We call the set

$$OneStep_s = \bigcup_{p \xrightarrow[\alpha]{l} r} B \times \{s[\sigma(r)]_p\} ,$$

with  $\alpha = \alpha_1; \dots; \alpha_n$  ( $\alpha_i \in S \cup \check{S}$ , see proposition 4.2.4),  $\sigma(l) = s|_p$ , and

$$B = \{\beta_1; \dots; \beta_n \mid \beta_i \in S \cup \check{S} \text{ and } p \text{ in } s \text{ is positive with respect to } (\alpha_i, \beta_i)\} ,$$

the *one-step-reachability set* of  $s$ .

**4.2.6 Definition.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation in our logic of special relation, and let  $s$  be a term in  $\mathcal{T}_\Sigma(\mathcal{X})$ . We call the set  $Reach_s \subseteq S^* \times \mathcal{T}_\Sigma(\mathcal{X})$  the *reachability set* of  $s$ , whenever  $(\delta, v) \in Reach_s$  if and only if there exist terms  $s = s_0, s_1, \dots, s_n = v$  and relations  $\delta_1, \dots, \delta_n$ , such that

$$1. \forall i \in [1 \dots n] \quad (\delta_i, s_i) \in OneStep_{s_{i-1}}, \text{ and}$$

$$2. \delta = \delta_1; \dots; \delta_n ,$$

$$\text{i.e., } s = s_0 \xrightarrow[\delta_1]{\quad} s_1 \xrightarrow[\delta_2]{\quad} \dots \xrightarrow[\delta_n]{\quad} s_n = v.$$

**4.2.7 Decision algorithm.** In order to decide whether  $\Gamma \vdash s \gamma t$ , we will have to compute reachability sets  $Reach_s$  and  $Reach_t$  and check if there exists a term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\alpha, \beta \in S^*$ , such that  $(\alpha, u) \in Reach_s$ ,  $(\beta, u) \in Reach_t$ , and  $\alpha; \beta \sqsubseteq \gamma$ . Obviously, sets  $Reach_s$  and  $Reach_t$  need to be always finite, and the partial order  $\sqsubseteq$  must be decidable. For the finiteness of the reachability sets, the rewrite relation has to be *terminating*, so that no branch of the rewrite tree is infinite, and it has to be *finitely branching*, so that, by performing a rewrite step at a given term, we only need to explore a finite number of branches.

**4.2.8 Remark.** Notice that the decision algorithm described in 4.2.7 is *not* based on normal-form computation as in the case of equational term rewriting, but instead computes the whole rewrite trees starting from  $s$  and  $t$ .

**4.2.9 Termination and finite branching.** Termination and finite branching are defined in a similar way as in standard equational rewriting, and we refer to (Dershowitz and Jouannaud, 1990) and (Baader and Nipkow, 1998) for a deeper discussions concerning these subjects.

Here we will be only interested in general methods guaranteeing, that given a finite set of atomic sentences in our logic of special relations, it indeed forms a terminating and finitely branching term rewriting system. Therefore, we are going to orient rewrite rules following a well-founded ordering  $\succ$  on terms. We refer to 2.1.4 for the necessary definitions.

It is obvious that given a term rewriting system  $\Gamma$ , the rewrite relation  $\xrightarrow{\Gamma}$  is terminating if its transitive closure  $\xrightarrow{\Gamma}^+$  is included in a well-founded ordering. The only additional difficulty within our general framework is to prove finite branching, since, given a position  $p$  in a term  $s$  and a rewrite rule  $l \xrightarrow{\alpha} r$  in  $\Gamma$ , there may be, in principle, more than one possible rewrites of the form  $s \xrightarrow{\gamma} t$ , depending on how many relations  $\gamma$  satisfy that the polarity of subterm position  $p$  in  $s$  is positive with respect to  $(\alpha, \gamma)$ . We allow, therefore, only a finite number of special relations and of rewrite rules in the system.

**4.2.10 Proposition.** *Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation in our logic of special relations, where  $S$  and  $\Gamma$  are finite. Let  $\xrightarrow{\Gamma}$  be the rewrite relation defined by  $\Gamma$  interpreting its atomic sentences as rewrite rules, following a well-founded ordering on terms, and let  $s$  be a term in  $\mathcal{T}_{\Sigma}(\mathcal{X})$ . Rewrite relation  $\xrightarrow{\Gamma}$  is finitely branching, i.e., the set  $\text{OneStep}_s$  as defined in definition 4.2.5 is finite.*

PROOF:  $B$  is a finite set. It is isomorphic to a subset of  $\overbrace{(S \cup \check{S}) \times \cdots \times (S \cup \check{S})}^n$ , which is finite, because  $S$  is finite. Consequently,  $B \times \{s[\sigma(r)]_p\}$  is also finite, and, since there are a finite number of subterm positions  $p$  in  $s$  and of rewrite rules  $l \xrightarrow{\alpha} r \in \Gamma$ ,  $\text{OneStep}_s$  is finite, too.  $\square$

**4.2.11 Decidability of  $\sqsubseteq$ .** Recall that in 4.2.7 we have also pointed out that, in order to state a decision algorithm for theoremhood, in addition to termination and finite branching, we also need the partial order  $\sqsubseteq$  of our structure  $\mathcal{S}^*$  to be decidable. Later, in chapter 5, we will see how we define particular partial orders by giving a finite number of axioms of the form  $\delta_1; \cdots; \delta_n \sqsubseteq \gamma$ , where  $\delta_i, \gamma$  are relations in  $S \cup \check{S}$ , and  $n \geq 0$ . Whenever the partial order relation is defined in this way, every relation  $\alpha = \alpha_1; \cdots; \alpha_n$  ( $\alpha_i \in S \cup \check{S}$ , see proposition 4.2.4) lying below a given relation  $\beta \in S \cup \check{S}$  can be seen as a word of the language generated by the context-free grammar with production rules

$$U_{\gamma} \rightarrow U_{\delta_1} \cdots U_{\delta_n}$$

for each axiom  $\delta_1; \cdots; \delta_n \sqsubseteq \gamma$ , and

$$U_{\gamma} \rightarrow \gamma$$

for each relation  $\gamma \in S \cup \check{S}$ .  $U_{\gamma}$  and  $U_{\delta_i}$  stand for non-terminal symbols, while relation symbol  $\gamma$  stands for a terminal symbol of the grammar. Of course, this makes the partial order  $\sqsubseteq$  decidable.

If, in addition, the language generated by this context-free grammar turns out to be regular, as happens within the particular cases studied later in chapter 5, we can define a couple of finite state automata that recognize those compositions of relations lying below a particular relation, exploring the word (i.e.,

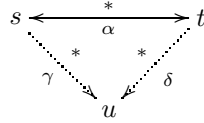
the compositions of relations) either from the left, or else from the right, as we explore the search space. We can thus prune those branches of the search space that do not lead to a composition of relations lying below the special relation of the atomic sentence to be proved by our decision algorithm.

**4.2.12 Soundness and completeness.** That the decision algorithm proposed in 4.2.7 is sound is a direct consequence of the definition of rewriting along relations given in definition 4.1.7. The completeness of the algorithm depends on the existence of a rewrite proof for each theorem of the theory, and this is a consequence of theorem 4.4.18 seen later.

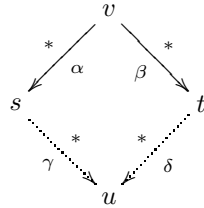
### 4.3 Confluence

If we want that, given a term rewriting system  $\Gamma$ , every atomic formula provable by a sequence of rewrite steps in  $\Gamma$  can also be proved by a rewrite proof, in addition to be finitely branching and terminating, the rewrite system needs to be *Church-Rosser*. Let us define what the Church-Rosser property, and its closely related properties of *confluence* and *local confluence*, for term rewriting along relations look like:

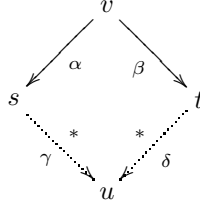
**4.3.1 Definition.** A term rewriting systems  $\Gamma$  is *Church-Rosser* if, for any pair of terms  $s, t \in \mathcal{T}_\Sigma(\mathcal{X})$  and relation  $\alpha \in S^*$ , such that  $s \xrightarrow[\alpha]{*} t$ , there exists a term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\gamma, \delta \in S^*$ , such that  $s \xrightarrow[\gamma]{*} u \xleftarrow[\delta]{*} t$ , and  $\gamma; \delta \sqsubseteq \alpha$ .



**4.3.2 Definition.** A term rewriting systems  $\Gamma$  is *confluent* if, for any three terms  $s, t, v \in \mathcal{T}_\Sigma(\mathcal{X})$  and pair of relations  $\alpha, \beta \in S^*$ , such that  $s \xleftarrow[\alpha]{*} v \xrightarrow[\beta]{*} t$ , there exists a term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\gamma, \delta \in S^*$ , such that  $s \xrightarrow[\gamma]{*} u \xleftarrow[\delta]{*} t$ , and  $\gamma; \delta \sqsubseteq \alpha; \beta$ .



**4.3.3 Definition.** A term rewriting systems  $\Gamma$  is *locally confluent* if, for any three terms  $s, t, v \in \mathcal{T}_\Sigma(\mathcal{X})$  and pair of relations  $\alpha, \beta \in S^*$ , such that  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$ , there exists a term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\gamma, \delta \in S^*$ , such that  $s \xrightarrow{\gamma}^* u \xleftarrow{\delta}^* t$ , and  $\gamma; \delta \sqsubseteq \alpha; \beta$ .

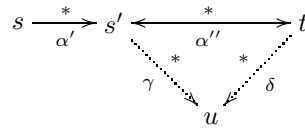


We call  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$  a *peak*, and say that it is *convergent* when it has a rewrite proof, and *divergent* otherwise. Therefore, a term rewriting systems is locally confluent when all peaks are convergent. The following propositions are true in our general setting. Their proofs are based on well-known proof techniques used for abstract reduction relations (see e.g., (Huet, 1980)).

**4.3.4 Proposition.** A term rewriting system  $\Gamma$  is Church-Rosser if and only if it is confluent.

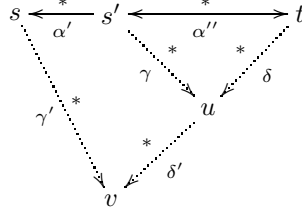
PROOF: The ‘only if’ direction is trivial. For the ‘if’ direction, we prove the proposition by induction on the length  $n$  of the proof  $s \xrightarrow{\alpha}^* t$ . If  $n = 0$ , the proposition holds vacuously. If  $n > 0$ , we have that  $s \xrightarrow{\alpha'} s' \xrightarrow{\alpha''}^* t$ , where  $\alpha'; \alpha'' \sqsubseteq \alpha$ . By the induction hypothesis, there exist  $u, \gamma, \delta$ , such that  $s' \xrightarrow{\gamma}^* u \xleftarrow{\delta}^* t$  and  $\gamma; \delta \sqsubseteq \alpha''$ .

1. If  $s \xrightarrow{\alpha'} s'$ , we have that  $s \xrightarrow{\alpha'; \gamma}^* u \xleftarrow{\delta}^* t$  and  $\alpha'; \gamma; \delta \sqsubseteq \alpha'; \alpha'' \sqsubseteq \alpha$ .



2. If  $s \xleftarrow{\alpha'} s'$ , by confluence, there exist  $v, \gamma', \delta'$ , such that  $s \xrightarrow{\gamma'}^* v \xleftarrow{\delta'}^* u$  and  $\gamma'; \delta' \sqsubseteq \alpha'; \gamma$ . Therefore, we have that  $s \xrightarrow{\gamma'}^* v \xleftarrow{\delta'; \delta}^* t$  and  $\gamma'; \delta'; \delta \sqsubseteq \gamma; \delta \sqsubseteq \alpha''$ .

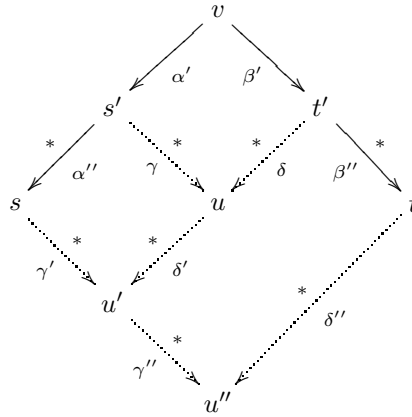
$$\alpha'; \gamma; \delta \sqsubseteq \alpha'; \alpha'' \sqsubseteq \alpha.$$



□

**4.3.5 Proposition.** *A terminating term rewriting system  $\Gamma$  is confluent if and only if it is locally confluent.*

PROOF: The ‘only if’ direction is trivial. For the ‘if’ direction, we prove the proposition by induction on the lengths  $n$  and  $m$  of the proofs  $s \xrightarrow[\alpha]{*} v$  and  $v \xrightarrow[\beta]{*} t$ , respectively. If  $n = 0$  or  $m = 0$ , the proposition holds vacuously. If  $n, m > 0$ , we have that  $s \xleftarrow[\alpha'']{*} s' \xleftarrow[\alpha']{*} v \xrightarrow[\beta']{*} t' \xrightarrow[\beta'']{*} t$  and  $\alpha''; \alpha' \sqsubseteq \alpha$  and  $\beta'; \beta'' \sqsubseteq \beta$ . By local confluence, there exist  $u, \gamma, \delta$ , such that  $s' \xrightarrow[\gamma]{*} u \xleftarrow[\delta]{*} t'$  and  $\gamma; \delta \sqsubseteq \alpha'; \beta'$ . By the induction hypothesis, there exist  $u', \gamma', \delta'$ , such that  $s \xrightarrow[\gamma']{*} u' \xleftarrow[\delta']{*} u$  and  $\gamma'; \delta' \sqsubseteq \alpha''; \gamma$ . Again, by the induction hypothesis, there exist  $u'', \gamma'', \delta''$ , such that  $u' \xrightarrow[\gamma'']{*} u'' \xleftarrow[\delta'']{*} t$  and  $\gamma''; \delta'' \sqsubseteq \delta'; \delta; \beta''$ . Therefore we have that  $s \xrightarrow[\gamma'; \gamma'']{*} u'' \xleftarrow[\delta'']{*} t$ , and  $\gamma'; \gamma''; \delta'' \sqsubseteq \gamma'; \delta'; \delta; \beta'' \sqsubseteq \alpha''; \gamma; \delta; \beta'' \sqsubseteq \alpha''; \alpha'; \beta'; \beta'' \sqsubseteq \alpha; \beta$ .



□

## 4.4 Critical Atoms

Recall that, in standard term rewriting, as a result of the Knuth-Bendix theorem (Knuth and Bendix, 1970), we may check for local confluence by checking for convergence of all *critical pairs* arising from non-trivial overlaps among left-hand sides of rewrite rules of the term rewriting system at hand. Recall that a non-trivial overlap is an overlap on non-variable subterm positions. Based on the validity of this theorem, one can attempt to *complete* a terminating but non-Church-Rosser term rewriting system to a Church-Rosser one, by adding non-convergent critical pairs as new rewrite rules to the system. It is, therefore, very desirable to see if the Knuth-Bendix theorem, or at least a variant of it, is also valid within the context of term rewriting along relations.

Many aspects of the following discussion have already been thoroughly studied within the context of standard rewriting (see for instance (Dershowitz and Jouannaud, 1990)), but it is worth to work through them again within our general framework of rewriting along relations, in order to highlight the subtleties we have to deal with now.

**4.4.1 Peak.** Given a theory presentation  $((\mathcal{S}^*, \Sigma), \Gamma)$ , let  $s, t, v$  be terms in  $\mathcal{T}_\Sigma(\mathcal{X})$ ,  $\alpha, \beta$  relations in  $\mathcal{S}^*$ , and let us consider  $\Gamma$  as a term rewriting system. A peak  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$  in  $\Gamma$  is the result of rewriting with two (not necessarily distinct) rewrite rules  $l_1 \xrightarrow{\alpha'} r_1$  and  $l_2 \xrightarrow{\beta'} r_2$  in  $\Gamma$  on (not necessarily distinct) subterm positions  $p$  and  $q$  in  $v$ . There are two different cases to consider:

1. either  $p \not\leq q$  and  $q \not\leq p$ , i.e., the rewrite rules do not overlap, but are applied on two disjoint subterm positions in  $v$ ,
2. or else  $p \leq q$  or  $q \leq p$ , i.e., the rewrite rules do overlap.<sup>3</sup>

Subterm positions  $p$  and  $q$  may have a common prefix, i.e., there exist an  $r$  such that  $p = r \cdot p'$  and  $q = r \cdot q'$ , and in the previous two cases it is desirable to neglect the fragment of the term above  $r$ , since it does not take part in the actual formation of the peak, it is just a context  $w[\ ]_r$  put around the terms. In standard term rewriting we can indeed strip off this context, because equality is a congruence, but in our general approach, the relations we deal with do not necessarily have such property. We need to proof explicitly that context application, when possible, preserves local confluence.

**4.4.2 Context application.** Context application preserving local confluence means that, if a peak  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$  converges, then, by applying a context  $w[\ ]_r$  around the terms involved, the resulting peak  $w[s]_r \xleftarrow{\alpha'} w[v]_r \xrightarrow{\beta'} w[t]_r$  converges, too.

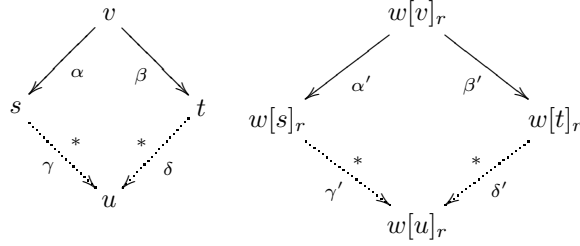
Though this is obviously true in standard term rewriting, its validity in the framework of our general notion of term rewriting is not that straightforward. A

---

<sup>3</sup>We refer to 2.1.1 for the exact meaning of the order  $\leq$  between subterm positions.



context  $w[\ ]_r$  can be applied around the terms if there exist relations  $\alpha', \beta' \in S^*$ , such that  $r$  in  $w$  is positive with respect to  $(\alpha, \alpha')$  and with respect to  $(\beta, \beta')$ . But then the resulting peak converges only if there also exist relations  $\gamma', \delta' \in S^*$ , such that  $r$  in  $w$  is positive with respect to  $(\gamma, \gamma')$  and with respect to  $(\delta, \delta')$ , and, in addition,  $\gamma'; \delta' \sqsubseteq \alpha'; \beta'$ .



Unfortunately, this is not always true in general, and we will need to put additional conditions on the polarity of our signature's function symbols for context application to preserve local confluence. In order to simplify the treatment of these conditions, we have already chosen to define the monotonicity and antimonotonicity properties of function symbols only with respect to special relations in  $S$  (see 3.2.1), so that the polarities of their argument positions with respect to general binary relations in  $S^*$  depend on their polarities with respect to special relations in  $S$  and on the partial order in  $S^*$ .

We now define the notion of *well-polarized* signatures in order to capture those special cases, for which context application is going to preserve local confluence:

**4.4.3 Definition.** A signature  $(S^*, \Sigma)$  in the logic of special relations is *well-polarized*, if for all function symbols  $f \in \Sigma$ , argument positions  $i$  in  $f$ , and pairs of relations  $\alpha, \beta \in S^*$ —where  $\alpha = \alpha_1; \dots; \alpha_n$ ,  $\alpha_j \in S \cup \check{S}$  for all  $j \in [1 \dots n]$  (see proposition 4.2.4)—we have that, if  $i$  in  $f$  is positive with respect to  $(\alpha, \beta)$ , then there exist relations  $\beta_1, \dots, \beta_n \in S \cup \check{S}$  such that  $i$  in  $f$  is positive with respect to  $(\alpha_j, \beta_j)$ , for all  $j \in [1 \dots n]$ , and  $\beta_1; \dots; \beta_n \sqsubseteq \beta$ .

Well-polarization is a condition put on argument positions of function symbols in signatures, but it can be extended to subterm positions of terms in  $\mathcal{T}_\Sigma(\mathcal{X})$ :

**4.4.4 Proposition.** Let  $(S^*, \Sigma)$  be a well-polarized signature, let  $s$  be a term in  $\mathcal{T}_\Sigma(\mathcal{X})$ , and let  $\alpha, \beta$  be a pair of relations in  $S^*$ —where  $\alpha = \alpha_1; \dots; \alpha_n$ ,  $\alpha_i \in S \cup \check{S}$ , for all  $i \in [1 \dots n]$  (proposition 4.2.4). If  $p$  in  $s$  is positive with respect to  $(\alpha, \beta)$ , then there exist relations  $\beta_1, \dots, \beta_n \in S \cup \check{S}$ , such that  $p$  in  $s$  is positive with respect to  $(\alpha_i, \beta_i)$ , for all  $i \in [1 \dots n]$ , and  $\beta_1; \dots; \beta_n \sqsubseteq \beta$ .

PROOF: By induction on the length of position  $p = d_1 \dots d_m$ ,  $m \geq 1$ ,  $d_m \in \mathbb{N}$ . Let  $f$  be the top-most function symbol of  $s$ , i.e.,  $s = f(s_1, \dots, s_k)$ , where  $s_i \in \mathcal{T}_\Sigma(\mathcal{X})$  for all  $i \in [1 \dots k]$ .

1. If  $m = 1$  then  $p = d$ ,  $d \in \mathbb{N}$ , the proposition is true by definition 4.1.2.
2. If  $m > 1$  then  $p = d \cdot p'$ ,  $d \in \mathbb{N}$ . By definition 4.1.2, there exists  $\gamma \in S^*$ , such that  $p'$  in  $s_d$  is positive with respect to  $(\alpha, \gamma)$ , and  $d$  in  $f$  is positive with respect to  $(\gamma, \beta)$ . By the induction hypothesis, there exist relations  $\gamma_1, \dots, \gamma_n \in S \cup \check{S}$ , such that  $p'$  in  $s_d$  is positive with respect to  $(\alpha_i, \gamma_i)$ , for all  $i \in [1 \dots n]$ , and  $\gamma_1; \dots; \gamma_n \sqsubseteq \gamma$ . Consequently,  $d$  in  $f$  is also positive with respect to  $(\gamma_1; \dots; \gamma_n, \beta)$ . Since the signature is well-polarized, there exist relations  $\beta_1, \dots, \beta_n \in S \cup \check{S}$ , such that  $d$  in  $f$  is positive with respect to  $(\alpha_i, \beta_i)$ , for all  $i \in [1 \dots n]$ , and  $\beta_1; \dots; \beta_n \sqsubseteq \beta$ . Finally, by definition 4.1.2,  $p$  in  $s$  is positive with respect to  $(\alpha_i, \beta_i)$ , for all  $i \in [1 \dots n]$ .

□

Indeed, when dealing with well-polarized signatures, local confluence is closed under context application:

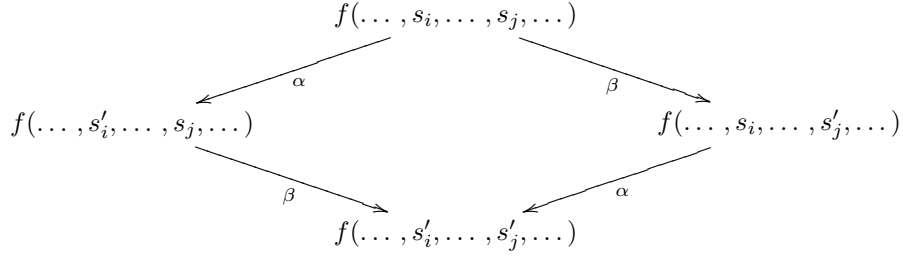
**4.4.5 Proposition.** *Let  $(S^*, \Sigma)$  be a well-polarized signature, let  $s, t, v$  be terms in  $\mathcal{T}_\Sigma(\mathcal{X})$ , and let  $\alpha, \beta$  be relations in  $S^*$ . If the peak  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$  converges, then, for each term  $w \in \mathcal{T}_\Sigma(\mathcal{X})$ , argument position  $r$  in  $w$ , and relations  $\alpha', \beta' \in S^*$  such that  $r$  in  $w$  is positive with respect to  $(\alpha, \alpha')$  and with respect to  $(\beta, \beta')$ , the peak  $w[s]_r \xleftarrow{\alpha'} w[v]_r \xrightarrow{\beta'} w[t]_r$  converges, too.*

PROOF: Since the peak  $s \xleftarrow{\alpha} v \xrightarrow{\beta} t$  converges, there exist a term  $u \in \mathcal{T}_\Sigma(\mathcal{X})$  and relations  $\gamma, \delta \in S^*$ , such that  $s \xrightarrow{\gamma}^* u \xleftarrow{\delta}^* t$  and  $\gamma; \delta \sqsubseteq \alpha; \beta$ . We have that  $r$  in  $w$  is positive with respect to  $(\alpha; \beta, \alpha'; \beta')$ , therefore,  $r$  in  $w$  is also positive with respect to  $(\gamma; \delta, \alpha'; \beta')$ . By proposition 4.4.4, there exist relations  $\gamma', \delta' \in S^*$  such that  $r$  in  $w$  is positive with respect to  $(\gamma, \gamma')$  and  $(\delta, \delta')$ . Consequently,  $w[s]_r \xrightarrow{\gamma'}^* w[u]_r \xleftarrow{\delta'}^* w[t]_r$ . Also by proposition 4.4.4,  $\gamma'; \delta' \sqsubseteq \alpha'; \beta'$ . □

**4.4.6 Remark.** A negative consequence of not taking into account the well-polarization of the signature will be shown in chapter 6, where we discuss Kriaučiukas and Walicki's proof calculus for specifications with set relations presented in (Kriaučiukas and Walicki, 1995). The calculus is based on an extension of rewrite techniques to deal with three special relations, but turns out *not* to be complete when combined with redundancy, because of the non-preservation of context application.

**4.4.7 Disjoint case.** Recall the formation of a peak given in 4.4.1. Knowing that we can strip off the context not involved in the rewritings that form the actual peak, let us now see the case when the peak is due to the application of two rewrite rules on positions  $p \not\leq q$  and  $q \not\leq p$ , i.e.,  $p = i.p'$  and  $q = j.q'$ , and  $i \neq j$ ,  $i, j \in \mathbb{N}$ .

Let  $f$  denote the top-most function symbol of  $v$ , and let  $s_i$  and  $s_j$  be the subterms of  $v$  on positions  $i$  and  $j$ . Let  $\alpha''$  and  $\beta''$  denote the relations along terms  $s_i$  and  $s_j$  are rewritten to  $s'_i$  and  $s'_j$ , due to the application of rewrite rules  $l_1 \xrightarrow{\alpha'} r_1$  and  $l_2 \xrightarrow{\beta'} r_2$ . By rewriting in  $f(\dots, s'_i, \dots, s_j, \dots)$  term  $s_j$  to  $s'_j$  with rule  $l_2 \xrightarrow{\beta'} r_2$  on position  $q$ , and by rewriting in  $f(\dots, s_i, \dots, s'_j, \dots)$  term  $s_i$  to  $s'_i$  with rule  $l_1 \xrightarrow{\alpha'} r_1$  on position  $p$ , we obtain the common term  $f(\dots, s'_i, \dots, s'_j, \dots)$ .



But for the peak to converge, we also need that  $\beta; \alpha \sqsubseteq \alpha; \beta$ . Consequently, peaks due to disjoint overlaps converge, whenever those binary relations commute, with respect to which specific argument positions in specific function symbols are positive. Before we go on stating the exact conditions our signature needs to satisfy, we need to first introduce a stronger notion of well-polarization.

**4.4.8 Definition.** A signature  $(\mathcal{S}^*, \Sigma)$  in the logic of special relations is *strongly well-polarized* if, for all function symbols  $f \in \Sigma$ , argument positions  $i$  in  $f$ , and pairs of relations  $\alpha, \beta \in \mathcal{S}^*$ —where  $\alpha = \alpha_1; \dots; \alpha_n$ ,  $\alpha_j \in S \cup \dot{S}$ , for all  $j \in [1 \dots n]$  (see proposition 4.2.4)—, we have that, if  $i$  in  $f$  is positive with respect to  $(\alpha, \beta)$ , then there exist relations  $\beta_1, \dots, \beta_n \in S \cup \dot{S}$ , such that  $i$  in  $f$  is positive with respect to  $(\alpha_j, \beta_j)$ , for all  $j \in [1 \dots n]$ , and  $\beta_1; \dots; \beta_n = \beta$ .

**4.4.9 Remark.** Notice that the difference with respect to definition 4.4.3 is that  $\beta_1, \dots, \beta_n = \beta$  for strong well-polarization, instead of  $\beta_1, \dots, \beta_n \sqsubseteq \beta$ .

**4.4.10 Definition.** A signature  $(\mathcal{S}^*, \Sigma)$  in the logic of special relations is *well-commuting* if, for all function symbols  $f \in \Sigma_n$ ,  $n \in \mathbb{N}$ , argument positions  $i, j \in [1 \dots n]$ ,  $i \neq j$ , and special relations  $\alpha, \alpha', \beta, \beta' \in S \cup \dot{S}$ , we have that, whenever both, the  $i$ -th and  $j$ -th argument positions of  $f$  are positive with respect to  $(\alpha', \alpha)$  and  $(\beta', \beta)$ , respectively, then  $\alpha; \beta \sqsubseteq \beta; \alpha$ .

Well-commutation is a condition put on special relations and their conversions, but it can be extended to general binary relations in  $\mathcal{S}^*$ , if the signature is also strongly well-polarized:

**4.4.11 Proposition.** *Let  $(S^*, \Sigma)$  be a strongly well-polarized and well-commuting signature. If, for all function symbols  $f \in \Sigma_n$ ,  $n \in \mathbb{N}$ , argument positions  $i, j \in [1 \dots n]$ ,  $i \neq j$ , and special relations  $\alpha, \alpha', \beta, \beta' \in S^*$ , we have that, whenever both, the  $i$ -th and  $j$ -th argument positions of  $f$  are positive with respect to  $(\alpha', \alpha)$  and  $(\beta', \beta)$ , respectively, then  $\alpha; \beta \sqsubseteq \beta; \alpha$ .*

PROOF: Let  $\alpha' = \alpha'_1; \dots; \alpha'_n$  and  $\beta' = \beta'_1; \dots; \beta'_m$ ,  $m, n \geq 0$ , such that, for all  $k \in [1 \dots n]$  and  $l \in [1 \dots m]$ ,  $\alpha_k, \beta_l \in S \cup \bar{S}$  (proposition 4.2.4). The proof is by induction over  $n$  and  $m$ .

1. If  $n = 0$  then  $1'; \beta \sqsubseteq \beta; 1'$  (analogously for  $m = 0$ ).
2. If  $n, m > 0$  then let  $\bar{\alpha}' = \alpha'_2; \dots; \alpha'_n$  and  $\bar{\beta}' = \beta'_2; \dots; \beta'_m$ . By strong well-polarization there exist relations  $\alpha_1, \beta_1 \in S \cup \bar{S}$  and  $\bar{\alpha}, \bar{\beta} \in S^*$  with  $\alpha = \alpha_1; \bar{\alpha}$  and  $\beta = \beta_1; \bar{\beta}$ , such that
  - the  $i$ -th argument position in  $f$  is positive with respect to  $(\alpha'_1, \alpha_1)$  and with respect to  $(\bar{\alpha}', \bar{\alpha})$
  - the  $j$ -th argument position in  $f$  is positive with respect to  $(\beta'_1, \beta_1)$  and with respect to  $(\bar{\beta}', \bar{\beta})$

Therefore,

$$\alpha_1; \beta_1 \sqsubseteq \beta_1; \alpha_1 \quad ,$$

and, by the induction hypothesis,

$$\begin{aligned} \alpha_1; \bar{\beta} &\sqsubseteq \bar{\beta}; \alpha_1 \\ \bar{\alpha}; \beta_1 &\sqsubseteq \beta_1; \bar{\alpha} \\ \bar{\alpha}; \bar{\beta} &\sqsubseteq \bar{\beta}; \bar{\alpha} \quad . \end{aligned}$$

Consequently,

$$\begin{aligned} \alpha; \beta &= \alpha_1; \bar{\alpha}; \beta_1; \bar{\beta} \\ &\sqsubseteq \alpha_1; \beta_1; \bar{\alpha}; \bar{\beta} \\ &\sqsubseteq \beta_1; \alpha_1; \bar{\beta}; \bar{\alpha} \\ &\sqsubseteq \beta_1; \bar{\beta}; \alpha_1; \bar{\alpha} \\ &= \beta; \alpha \quad . \end{aligned}$$

□

**4.4.12 Proposition.** *Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation, such that  $(S^*, \Sigma)$  is a strongly well-polarized and well-commuting signature. All peaks due to the application of two rewrite rules in  $\Gamma$  on disjoint subterm positions converge.*

PROOF: Let  $s \xleftarrow[\alpha]{} v \xrightarrow[\beta]{} t$  be a peak due to the application of two rewrite rules  $r_1 \xleftarrow[\alpha']{} l_1$  and  $l_2 \xrightarrow[\beta']{} r_2$  in  $\Gamma$  on two disjoint subterm positions  $p = i.p'$  and  $q = j.q'$  of  $v$ ,  $i \neq j$ , and let  $f$  be the top-most function symbol of term  $v$ . We have that  $s = v[\sigma_1(r_1)]_p$  and  $t = v[\sigma_2(r_2)]_q$ , where  $\sigma_1$  and  $\sigma_2$  are the unifiers of  $v|_p$  with  $l_1$ , and  $v|_q$  with  $l_2$ , respectively. It follows that subterm positions  $p$  and  $q$  in  $v$  are positive with respect to  $(\alpha', \alpha)$  and  $(\beta', \beta)$ , respectively, and therefore there exist relations  $\alpha'', \beta'' \in S^*$ , such that the  $i$ -th and  $j$ -th argument positions of  $f$  are positive with respect to  $(\alpha'', \alpha)$  and  $(\beta'', \beta)$ , respectively. Furthermore, since positions  $p$  and  $q$  are disjoint, there exist  $u = s[\sigma_2(r_2)]_q = t[\sigma_1(r_1)]_p$ , such that  $s \xrightarrow[\beta]{} u \xleftarrow[\alpha]{} t$ . In addition, by proposition 4.4.11,  $\alpha; \beta \sqsubseteq \beta; \alpha$ , which means that the peak converges. Finally, by proposition 4.4.5, context application preserves local confluence.  $\square$

**4.4.13 Remark.** Notice that, since the condition in proposition 4.4.12 requires  $\beta; \alpha \sqsubseteq \alpha; \beta$  for *all* argument positions  $i, j \in [1 \dots n]$  in  $f$ ,  $i \neq j$ —for which, of course, the suitable polarities hold—, we will also need that  $\alpha; \beta \sqsubseteq \beta; \alpha$ , and therefore, we are actually requiring that  $\beta; \alpha = \alpha; \beta$ .

**4.4.14 Overlap case.** Recall from 4.4.1 that the overlap case arises when  $p \leq q$  or  $q \leq p$ . Without any loss of generality, we may consider only the case when  $q \leq p$ , and again we strip off the context, so that  $q = \lambda$ , which means that  $v = \sigma(l_2)$ ,  $\sigma$  being the most general unifier of  $l_2|_p$  and  $l_1$ . When term  $l_2|_p$  is not a variable, then the pair of terms  $\langle s, t \rangle$  is usually called a *critical pair*, which in standard rewriting is the only case we need to check for convergence. Unfortunately, when dealing with non-symmetric relations (recall the discussion on bi-rewriting in chapter 2)—and obviously that happens in our general approach of rewriting along general binary relations—we also need to check so called *variable instance pairs* for confluence. Variable instance pairs are formed when position  $p$  is a variable position in  $l_2$ , or when it is *below* a variable position in  $l_2$ . This latter means that for a variable position  $r$  in  $l_2$  we have that  $r \leq p$ , i.e., rule  $l_1 \xrightarrow[\alpha']{} r_1$  overlaps on an *instance* of rule  $l_2 \xrightarrow[\beta']{} r_2$  and on a subterm position of an instantiated variable of  $l_2$ .

**4.4.15 Terminology.** In standard equational rewriting we talk about critical or variable instance *pairs*, and the two terms are always related by equality. But in our general approach we also need to make explicit the binary relation in  $S^*$  that relates these two terms. For this reason, and though ‘critical pair’ and ‘variable instance pair’ are well established in the terminology of rewriting, we prefer to talk in our framework of *critical atoms* and *variable instance atoms*. Let us define them formally.

**4.4.16 Definition.** Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation, where  $\Gamma$  is considered a term rewriting system. If  $l \xrightarrow[\alpha]{} r$  and  $s \xrightarrow[\beta]{} t$  are two rewrite rules in

$\Gamma$  and  $\alpha$ ,  $\beta$ , and  $\gamma$  in  $S^*$  are relations, such that  $p$  is a positive position with respect to  $(\check{\alpha}, \gamma)$  of a non-variable subterm of  $s$ , and  $\sigma$  is the most general unifier of  $s|_p$  and  $l$ , then the atomic formula  $\sigma(s[r]_p) \gamma; \beta \sigma(t)$  is a *critical atom*.

**4.4.17 Definition.** Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation, where  $\Gamma$  is considered a term rewriting system. If  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$  are two rewrite rules in  $\Gamma$  and  $\alpha$ ,  $\beta$ , and  $\gamma$  in  $S^*$  are relations, such that  $p$  is a position of a variable subterm  $x$  of  $s$ ,  $\sigma$  is a substitution, such that  $\sigma(x)$  has  $l$  as subterm at position  $q$ , but  $\sigma(y) = y$ , for all  $y \neq x$ , position  $p \cdot q$  in  $\sigma(s)$  is a positive position with respect to  $(\check{\alpha}, \gamma)$ , then the critical atom  $\sigma(s)[r]_{p \cdot q} \gamma; \beta \sigma(t)$  is a *variable instance atom*.

**4.4.18 Theorem.** Let  $((S^*, \Sigma), \Gamma)$  be a theory presentation, such that  $(S^*, \Sigma)$  is a strongly well-polarized and well-commuting signature. The rewrite system  $\Gamma$  is locally confluent if and only if all critical and variable instance atoms have a rewrite proof.

PROOF: The ‘only if’ direction is trivial. For the ‘if’ direction, if we have a peak due to the application of two rewrite rules on disjoint position, then we have a rewrite proof by proposition 4.4.12. Otherwise, the peak is due to the overlap of two rewrite rules generating either a critical atom or a variable instance atom. Since we assume that such atoms have rewrite proofs, and by proposition 4.4.5 context application preserves the existence of such rewrite proofs, the peak must have a rewrite proof, too.  $\square$

## 4.5 Towards Effective Completion

As usual, a *Knuth-Bendix*-like completion procedure would attempt to complete a terminating, but non-confluent, term rewriting system to a confluent one, by adding divergent critical and variable instance atoms as new rewrite rules to the system. But several important differences to standard equational term rewriting appear, which are very significant for the practicability of the term rewriting approach to perform deductions with binary relations, specially if the signature includes functions that are monotonic or antimonotonic with respect to pairs of relations.

We already know from the bi-rewriting technique (see chapter 2) that, by moving from symmetric to non-symmetric transitive relations, two rewrite rules may give rise to infinite many variable instance atoms. This is a severe drawback for the practicability of the term-rewriting approach, but even more discouraging is the fact that, as long as we stick with the straightforward definition of critical atom given in 4.4.16, also a non-variable overlap of two rewrite rules may generate infinite many critical atoms, as the following example shows:

**4.5.1 Example.** Let  $(\mathcal{S}^*, \Sigma)$  be a signature of the logic of special relations, with  $S = \{\alpha, \beta, \gamma\}$  and  $\Sigma = \{a, b, c, f(-)\}$  (three constants and one unary function symbol), such that the first (and unique) argument position of  $f$  is positive with respect to  $(\alpha, \gamma)$ . Let us suppose that  $\alpha$  is reflexive, so that  $\mathcal{S}^*$  is the minimal structure satisfying  $1' \sqsubseteq \alpha$ . Let now  $\Gamma = \{f(a) \xrightarrow[\beta]{\alpha} c, a \xrightarrow[\alpha]{\beta} b\}$ . Since  $\alpha \sqsubseteq \alpha; \alpha$  but  $\gamma \not\sqsubseteq \gamma; \gamma$  we have that the argument position of  $f$  is positive with respect to  $(\alpha, \gamma; \dots; \gamma)$ , and consequently the two rewrite rules of  $\Gamma$  generate the following infinite number of critical atoms

$$f(b) \underbrace{\gamma; \dots; \gamma; \beta}_n c ,$$

where  $n \in \mathbb{N}$ . Of course, having monotonicity in an argument of a function with respect to the pair formed by a reflexive and a non-reflexive relation seems a quite ‘pathological’ situation, one that rarely will arise in practical specification, but without additional restrictions, we may have to deal with it. In fact, the problem lies in that this signature is not well-polarized.

**4.5.2 Dimensions of infinity.** Recall definition 4.4.17 of variable instance atoms. The possibly infinite number of them, formed by rewrite rules  $l \xrightarrow[\alpha]{\beta} r$  and  $s \xrightarrow[\beta]{\alpha} t$ , was due to the fact that we can choose infinite possible substitutions  $\sigma$  instantiating the variable  $x$  in  $s$  with a term having  $l$  as subterm. This is only one *dimension of infinity*; it does not exist for critical atoms. But in our general framework, with the introduction of relational expressions relating the two terms resulting from a critical peak, a second *dimension of infinity* arises, namely when, because of monotonicity properties of function symbols and the partial order over relations, infinite many relations between the two terms formed by the critical peak need to be taken into account. This happened, for instance, in example 4.5.1.

Fortunately, by restricting to well-polarized signatures, we can limit the number of relations that need to be taken into account in the formation of critical and variable instance atoms, without, therefore, falsifying theorem 4.4.18, as we will show next.

**4.5.3 Critical and variable instance atoms revisited.** With the following more accurate definitions of critical and variable instance atoms, we can limit the number of relations that have to be considered in their generation. Thus we are able to control one of the two dimensions of infinity discussed in 4.5.2. Later, in 4.5.8, we discuss how to get to grips with the other dimension of infinity, so that effective *Knuth-Bendix*-like completion may become possible.

**4.5.4 Definition.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation, where  $\Gamma$  is considered a term rewriting system. Let  $l \xrightarrow[\alpha]{\beta} r$  and  $s \xrightarrow[\beta]{\alpha} t$  be two rewrite rules in  $\Gamma$ , and let  $\alpha$ ,  $\beta$ , and  $\gamma$  be relations in  $\mathcal{S}^*$ , such that  $\alpha = \alpha_1; \dots; \alpha_n$  and

$\gamma = \gamma_1; \dots; \gamma_n$ , for  $\alpha_i, \gamma_i \in S \cup \check{S}$  (proposition 4.2.4). If, for all  $i \in [1 \dots n]$ ,  $p$  is a positive position with respect to  $(\check{\alpha}_i, \gamma_i)$  of a non-variable subterm of  $s$ , and  $\sigma$  is the most general unifier of  $s|_p$  and  $l$ , then the atomic formula  $\sigma(s[r]_p) \gamma; \beta \sigma(t)$  is a *critical atom*.

**4.5.5 Definition.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation, where  $\Gamma$  is considered a term rewriting system. Let  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$  be two rewrite rules in  $\Gamma$ , and let  $\alpha, \beta$ , and  $\gamma$  be relations in  $\mathcal{S}^*$ , such that  $\alpha = \alpha_1; \dots; \alpha_n$  and  $\gamma = \gamma_1; \dots; \gamma_n$ , for  $\alpha_i, \gamma_i \in S \cup \check{S}$  (proposition 4.2.4). If  $p$  is a position of a variable subterm  $x$  of  $s$ ,  $\sigma$  is a substitution such that  $\sigma(x)$  has  $l$  as subterm at position  $q$ , but  $\sigma(y) = y$ , for all  $y \neq x$ , and for all  $i \in [1 \dots n]$ , position  $p \cdot q$  in  $\sigma(s)$  is a positive position with respect to  $(\check{\alpha}_i, \gamma_i)$ , then the critical atom  $\sigma(s)[r]_{p \cdot q} \gamma; \beta \sigma(t)$  is a *variable instance atom*.

**4.5.6 Finiteness.** The finiteness of the number of relations to be considered between two terms arising from critical and variable instance atoms, as given in definitions 4.5.4 and 4.5.5, follows from the finiteness of set  $S$  of special relations, by reasoning analogously as done in proving the finiteness of set  $B$  given in definition 4.2.5 in the proof of proposition 4.2.10. That, with the new definitions, theorem 4.4.18 still holds, is guaranteed by the following lemma:

**4.5.7 Lemma.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation, such that  $(\mathcal{S}^*, \Sigma)$  is a strongly well-polarized and well-commuting signature. If all critical and variable instance atoms, as defined in 4.5.4 and 4.5.5, have a rewrite proof, so do all the critical and variable instance atoms, as defined in 4.4.16 and 4.4.17.

PROOF: Let  $s \gamma; \beta t$  be a critical or variable instance atom, as first defined in 4.4.16 and 4.4.17, formed from the overlap of two rewrite rules on a subterm position  $p$  of term  $s$ , because  $p$  is a positive position with respect to  $(\check{\alpha}, \gamma)$ . By proposition 4.2.4, there exist  $\alpha_1, \dots, \alpha_n \in S \cup \check{S}$ , such that  $\alpha = \alpha_1; \dots; \alpha_n$ , and by strong well-polarization of the signature, there exist relations  $\gamma_1, \dots, \gamma_n \in S \cup \check{S}$ , such that, for all  $i \in [1 \dots n]$ ,  $p$  in  $s$  is positive with respect to  $(\check{\alpha}_i, \gamma_i)$ , and  $\gamma = \gamma_1; \dots; \gamma_n$ . We are assuming that all critical and variable instance atoms, as defined in 4.5.4 and 4.5.5, have a rewrite proof, thus, in particular, we have that  $s \gamma_1; \dots; \gamma_n; \beta t$  has one, too, and since  $\gamma_1; \dots; \gamma_n \sqsubseteq \gamma$ , it follows that  $s \gamma; \beta t$  converges.  $\square$

**4.5.8 Infinite many variable instance atoms.** Because of the double dimension of infinity arising with variable instance atoms, dealing with them is infeasible in practice, even if, by well-polarization, one of the dimensions is controlled. Therefore, we may consider rewriting on subterm positions only when variable instance atoms are generally unnecessary. In the literature, you can find two ways to avoid the generation of new rewrite rules from variable instance atoms:



1. We may restrict the special relations of our signature to those satisfying only specific properties, such as symmetry and transitivity. We may allow the overlap on and below variable positions only in rules rewriting along these kind of relations. This is the approach followed by Bachmair and Ganzinger (1994a; 1998) in order to tame prolific variable subterm chaining inferences.
2. We may consider only specific kind of axioms in our theories, or certain algebraic structures, in which variable instance atoms are always convergent. Some cases have been studied by Levy and Agustí (1996), and Struth (1997).

A third and unexplored approach is presented here, in 4.5.11. It relies on exploiting the notion of polarity of argument positions of specific function symbols. Actually, this approach does not look under which conditions variable instance atoms are unnecessary, but when two rewrite rules only generate a *finite amount* of variable instance atoms.

Before going any further, let us first see when a variable instance atom is convergent in general, that is, when the two rules forming the peak through such an atom are also the rules that make it converge:

**4.5.9 Proposition.** *Let  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$  be two rewrite rules in  $\Gamma$  that form the variable instance atom  $\sigma(s)[r]_{p \cdot q} \gamma; \beta \sigma(t)$ , as defined in 4.5.5. Let  $n$  be the number of occurrences of variable  $x$  in  $s$ ,  $n \geq 1$ . One occurrence is in position  $p$ ; let  $p_1, \dots, p_{n-1}$  denote the other  $n-1$  positions where  $x$  occurs. Let  $m$  be the number of occurrences of  $x$  in  $t$ ,  $m \geq 0$ , and let  $p'_1, \dots, p'_m$  denote the positions in  $t$  where it occurs. If*

1. *there exist relations  $\gamma_1, \dots, \gamma_{n-1}$  in  $S^*$  such that positions  $p_i \cdot q$  in  $\sigma(s)[r]_{p \cdot q}$  are positive with respect to  $(\alpha, \gamma_i)$ ,  $i = 1 \dots n-1$ ,*
2. *there exist relations  $\delta_1, \dots, \delta_m$  in  $S^*$  such that positions  $p'_j \cdot q$  in  $\sigma(t)$  are positive with respect to  $(\beta, \delta_j)$ ,  $j = 1 \dots m$ , and*
3. *there exist permutations  $P$  and  $Q$  of sequences  $(1, \dots, n-1)$  and  $(1, \dots, m)$  respectively, such that*

$$\gamma_{P_1}; \dots; \gamma_{P_{n-1}}; \beta; \delta_{Q_1}; \dots; \delta_{Q_m} \sqsubseteq \gamma; \beta,$$

*then the variable instance atom  $\sigma(s)[r]_{p \cdot q} \gamma; \beta \sigma(t)$  is convergent.*

PROOF: Under these conditions, we can use rule  $l \xrightarrow{\alpha} r$  to rewrite each of the  $n-1$  occurrences of  $l$  in positions  $p_i \cdot q$  in term  $\sigma(s)[r]_{p \cdot q}$  along relations  $\gamma_i$ . Choosing the particular sequence of rewrite steps determined by the permutation  $P$ , we eventually rewrite term  $\sigma(s)[r]_{p \cdot q}$  to term  $\sigma'(s)$  along relation  $\gamma_{P_1}; \dots; \gamma_{P_{n-1}}$ , where  $\sigma'$  is the substitution such that  $\sigma'(x) = \sigma(x)[r]_q$  and  $\sigma'(y) = y$ , for

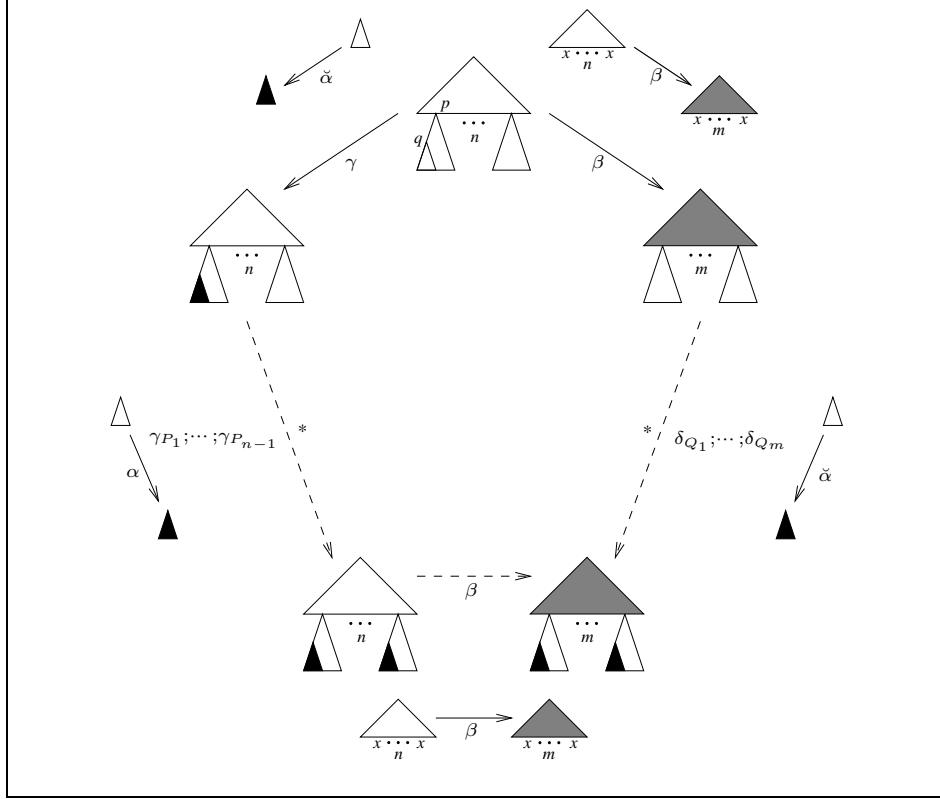


Figure 4.1: Convergent variable instance atom

all  $y \neq x$ . Analogously, under these conditions, we can use rule  $l \xrightarrow{\alpha} r$  to rewrite each of the  $m$  occurrences of  $l$  in positions  $p'_j \cdot q$  in term  $\sigma(t)$  along relations  $\delta_j$ . Again, choosing the particular sequence of rewrite steps determined by permutation  $Q$ , we eventually rewrite term  $\sigma(t)$  to term  $\sigma'(t)$  along relation  $(\delta_{Q_1}; \dots; \delta_{Q_m})^\vee$ . With rule  $s \xrightarrow{\beta} t$  we can now rewrite  $\sigma'(s)$  to  $\sigma'(t)$  along relation  $\beta$ , and since  $\gamma_{P_1}; \dots; \gamma_{P_{n-1}}; \beta; \delta_{Q_1}; \dots; \delta_{Q_m} \sqsubseteq \gamma; \beta$ , for these particular permutations, the variable instance atom converges (see figure 4.1).  $\square$

**4.5.10 Remark.** The conditions for a variable instance atom to be convergent are far too strong, and, in general, a completion procedure should add these atoms as new rewrite rules to the term rewriting system.

**4.5.11 Effectively dealing with variable instance atoms.** We have already mentioned, in 4.5.8, that we wanted to look under which conditions only a finite amount of variable instance atoms need to be considered by a completion procedure. This corresponds to cope with the other dimension of infinity (see 4.5.2) not yet treated.

Recall definition 4.4.17 or 4.5.5 of variable instance atom. There will be a finite number of such atoms between two given rewrite rules  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$ , if and only if there exist a finite number of substitutions  $\sigma$  satisfying that

1.  $\sigma(x)$  has term  $l$  as subterm at position  $q$ , and
2. position  $p \cdot q$  in  $\sigma(s)$  is positive with respect to  $(\check{\alpha}, \gamma)$ .

Basically, we need to limit the number of possible substitutions, and we are going to do this by exploiting the notion of polarity, so that position  $p \cdot q$  in  $\sigma(s)$  is positive with respect to  $(\check{\alpha}, \gamma)$  only for a finite number of terms  $\sigma(s)$ . We will need to define a *polarization relation* on which we put conditions, so that we eventually cut down the number of variable instance atoms to a finite amount:

**4.5.12 Definition.** A signature  $\Omega = (\mathcal{S}^*, \Sigma)$  determines a binary relation  $\triangleleft \subseteq (S \cup \check{S}) \times (S \cup \check{S})$  as follows: For every  $\alpha, \beta$  in  $S \cup \check{S}$ ,  $\alpha \triangleleft \beta$  if and only if there exists a function symbol  $f$  in  $\Sigma_n$ ,  $n \in \mathbb{N}$ , and an argument position  $i \in [1 \dots n]$ , such that  $i$  in  $f$  is positive with respect to  $(\alpha, \beta)$ . We will call the transitive closure of  $\triangleleft$  the signature's *polarization relation*.

**4.5.13 Proposition.** Let  $((\mathcal{S}^*, \Sigma), \Gamma)$  be a theory presentation in the logic of special relations, considering  $\Gamma$  as a term rewriting system. If its polarization relation is irreflexive (i.e., it is an ordering), then there can be only a finite number of variable instance atoms between two arbitrary rewrite rules  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$  in  $\Gamma$ .

PROOF: Let  $\alpha = \alpha_1; \dots; \alpha_n$  and  $\gamma = \gamma_1; \dots; \gamma_n$ , where  $\alpha_i, \gamma_i \in S \cup \check{S}$ . Two rules  $l \xrightarrow{\alpha} r$  and  $s \xrightarrow{\beta} t$  form the *variable instance atom*

$$\sigma(s)[r]_{p \cdot q} \gamma; \beta \sigma(t)$$

if and only if position  $p \cdot q$  in  $\sigma(s)$  is positive with respect to  $(\check{\alpha}_i, \gamma_i)$ , for all  $i \in [1 \dots n]$ .

Taking an arbitrary  $i \in [1 \dots n]$ ,  $p \cdot q$  in  $\sigma(s)$  is a positive position with respect to  $(\check{\alpha}_i, \gamma_i)$ , if and only if there exists  $\delta_i \in S \cup \check{S}$ , such that variable position  $p$  in  $s$  is positive with respect to  $(\delta_i, \gamma_i)$  and position  $q$  in  $\sigma(x)$  is positive with respect to  $(\check{\alpha}_i, \delta_i)$ .

Since  $S \cup \check{S}$  is finite, there exists only a finite number of such  $\delta_i$ . Thus the finiteness of the amount of variable instance atoms only depends on the number of terms  $\sigma(x) = t[l]_q \in \mathcal{T}_{\Sigma}(\mathcal{X})$ , such that  $q$  in  $t$  is positive with respect to  $(\check{\alpha}_i, \delta_i)$ . Let  $q = d_1 \dots d_n$  and

$$t =$$

$$f_1(x_1, \dots, x_{d_1-1}, f_2(\dots f_n(y_1, \dots, y_{d_n-1}, l, y_{d_n+1} \dots, y_m) \dots), x_{d_1+1}, \dots, x_k)$$

where  $d_j \in \mathbb{N}$ ,  $f_j \in \Sigma$ , and  $x_k, y_k \in \mathcal{X}$ . Notice that we construct  $t$  by taking the most general substitution  $\sigma$  assigning to  $x$  a term constructed with function symbols  $f_j$ , and with subterm  $l$  at  $q$ , so we subsume all other variable instance pairs constructed the same way. Position  $q$  in  $t$  is positive with respect to  $(\check{\alpha}_i, \delta_i)$  if and only if there exist a sequence of relations

$$\check{\alpha}_i = \rho_0, \dots, \rho_n = \delta_i \quad \rho_j \in S \cup \check{S} \ ,$$

such that  $d_j$  in  $f_j$  is positive with respect to  $(\rho_{j-1}, \rho_j)$ , for all  $j \in [1 \dots n]$ , which means that

$$\check{\alpha}_i = \rho_0 \triangleleft \dots \triangleleft \rho_n = \delta_i \ .$$

But, since the polarization relation is irreflexive, there exists only a finite number of sequences  $\rho_0, \dots, \rho_n$  satisfying such property, which means that there exists only a finite number of terms  $t[l]_q$ , for which  $q$  in  $t$  is positive with respect to  $(\check{\alpha}_i, \delta_i)$ . Consequently, there can be only a finite number of variable instance atoms between two rewrite rules.  $\square$

## 4.6 Discussion

Throughout this chapter we have shown the technical complexity involved in the generalization of the term rewrite technique to arbitrary binary relations and arbitrary monotonicity properties of function symbols. Confluence results are not translated in a straightforward way, but are much more subtle, so that the concept of well-polarized signature becomes necessary. But the most important drawbacks arise when we want to effectively deal with the completion of non-confluent term rewrite systems to confluent ones. In addition to the already known dimension of infinity in variable instance atoms, resulting from overlapping below variable positions, a new dimension of infinity arises, since infinite relations between terms formed from critical peaks need to be taken into account, too. Fortunately, we are able to control both dimensions of infinity by exploiting again the notion of polarity, with well-polarized signatures and irreflexive polarization relations.

Well-polarization turns out to be a quite natural property. For instance, it forces a function symbol to be polarized with respect to a pair of transitive relations, or else with respect to a pair of non-transitive relations. Thus, it avoids awkward polarities like with respect to a transitive and a non-transitive relation, or with respect to a reflexive and a non-reflexive transitive relation (recall example 4.5.1). On the other hand, an irreflexive polarization relation is not that natural, and it rules out many interesting theories. For instance, polarizations with respect to a relation with itself (standard monotonicity) makes a polarization relation to be reflexive.

From the perspective gained by our detailed analysis of where and when the term rewriting technique for general binary relations starts to get out of control, we want to discuss why several particular cases of term rewriting, such as standard equational term rewriting and bi-rewriting, work so well as deduction mechanisms.

**4.6.1 Equational rewriting.** The standard equational case arises when the unique special relation is *equality* ( $\approx$ ), a reflexive, symmetric, transitive binary relation with respect to which all argument positions of all function symbols are positively polarized. In this case, a variable instance atom formed by rewrite rules  $l \xrightarrow{\approx} r$  and  $s \xrightarrow{\approx} t$  is always convergent, since the three conditions of proposition 4.5.9 are satisfied:

1. all position  $p_i \cdot q$  are positive with respect to  $(\approx, \approx)$
2. all positions  $p'_i \cdot q$  are positive with respect to  $(\approx, \approx)$
3.  $\underbrace{\approx; \dots; \approx}_{n-1}; \approx; \underbrace{\approx; \dots; \approx}_m \sqsubseteq \approx; \approx$

The well-known consequence is that a Knuth-Bendix completion process does not need to generate variable instance atoms, since all of them will be convergent.

**4.6.2 Bi-rewriting.** The bi-rewriting case arises when the unique special relation is inclusion ( $\subseteq$ ), i.e., a reflexive, but non-symmetric, transitive binary relation with respect to which all argument positions of all function symbols are positively polarized, too. In this case, a variable instance atom formed by rewrite rule  $l \xrightarrow{\supseteq} r$  and  $s \xrightarrow{\subseteq} t$  is not always convergent, since only the last two conditions of proposition 4.5.9 are satisfied in general, but not the first:

1. all position  $p_i \cdot q$  are *not* necessarily positive with respect to  $(\supseteq, \subseteq)$
2. all positions  $p'_i \cdot q$  are positive with respect to  $(\subseteq, \subseteq)$
3.  $\underbrace{\subseteq; \dots; \subseteq}_{n-1}; \subseteq; \underbrace{\subseteq; \dots; \subseteq}_m \sqsubseteq \subseteq; \subseteq$

But, when rewrite rule  $s \xrightarrow{\subseteq} t$  is left-linear (i.e., variable  $x$  only occurs once in  $s$ , thus  $n = 1$ ), then the first condition of proposition 4.5.9 is trivially satisfied. In this case, a Knuth-Bendix completion process does not need to look for variable instance atoms, either.

**4.6.3 The roles of symmetry, monotonicity, and transitivity.** We mentioned, in 4.5.8, that one way to avoid the generation of new rewrite rules from variable instance atoms was to restrict the special relations of our signature to those satisfying only specific properties. In 4.6.1 and 4.6.2 we have shown two examples where we proceed in this way.

Thus, if in proposition 4.5.9 the relation  $\alpha$  along which rewrite rule  $l \xrightarrow{\alpha} r$  rewrites is symmetric, i.e.,  $\check{\alpha} = \alpha$ , then we might use this same rule to rewrite term  $\sigma(s)[r]_{p \cdot q}$  to  $\sigma'(s)$ . In addition we need also that the positions  $p_i \cdot q$  are polarized.

When we say that the relation  $\alpha$  is monotonic, we think of all argument positions of all function symbols of our signature being positive with respect to  $(\alpha, \alpha)$ . In this case it turns out that

$$\forall i \in [1 \dots n] \quad \gamma_i = \alpha \quad \text{and} \quad \forall j \in [1 \dots m] \quad \delta_j = \check{\alpha} \ ,$$

so that the permutations  $P$  and  $Q$  do not matter at all. If  $\alpha$  is transitive, too, i.e.,  $\alpha; \alpha \sqsubseteq \alpha$ , then the number of occurrences of variable  $x$  in  $s$  and in  $t$  does not matter either, because then  $\sigma(t) \xrightarrow[\alpha]{} \sigma'(t)$  and  $\sigma(s)[r]_{p,q} \xrightarrow[\alpha]{} \sigma'(s)$ , independently of the number of occurrences  $n$  and  $m$ .

Monotonicity alone causes many problems. Observe that the condition of having an irreflexive polarization relation excludes standard monotonicity, i.e., the polarity of a function symbol with respect to one relation with itself, and this is of course very restrictive for practical specification. But monotonicity and transitivity of  $\alpha$ , both combined with symmetry, allow us to consider variable instance atoms for which the following order among relation has to hold:

$$\alpha; \beta; \alpha \sqsubseteq \alpha; \beta$$

In the case of left-linear rewrite rules, it would be the sentence

$$\beta; \alpha \sqsubseteq \alpha; \beta \ .$$

Of course, such property holds when  $\alpha$  and  $\beta$  are the same relation.

## 4.7 Extensions of Term Rewriting

The obvious question arising at this point is how this general notion of rewriting along binary relations translates to other extensions of term rewriting, such as conditional rewriting and rewriting modulo theories, and how these techniques may be applied to resolution-based theorem proving through saturation techniques. Of course, all these extensions are of central importance, specially for the research aims we motivated in chapter 1. Furthermore, it could also happen that, by means of extending the general term rewriting technique presented in this thesis, we may be able to cope with the problems arising when reasoning with general binary relations in addition to equality.

**4.7.1 Linearization.** Consider, for example, the bi-rewriting case, i.e., a theory in our logic with one unique special relation, namely inclusion. In 4.6.2 we showed that, when rewrite rules are left-linear, then all critical atoms generated by an overlap below a variable position—all variable instance atoms—converge. In particular, recall example 2.5.7 and remark 2.5.8, where we saw how rewrite rules

$$f(x, x) \xrightarrow[\subseteq]{} x \tag{4.1}$$

$$a \xrightarrow[\supseteq]{} b \tag{4.2}$$

generate infinite many new rules of the form

$$f(C[b], C[a]) \xrightarrow{\subseteq} C[a] ,$$

where  $C[\ ]$  is an arbitrary context.

At first sight, one might think that such situation could be avoided by linearizing rule (4.1) containing the repeated variable  $x$ , replacing the rule by the following two conditional left-linear rewrite rules

$$f(y, x) \xrightarrow{\subseteq} x \quad \text{if } y \subseteq x \quad (4.3)$$

$$f(x, y) \xrightarrow{\subseteq} x \quad \text{if } y \subseteq x \quad (4.4)$$

because now, any inclusion of the form  $f(C[b], C[a]) \subseteq C[a]$  can be proved by conditional rewriting, no matter what the context  $C$  is. For example, let  $C \equiv f(\cdot, a)$ . We can perform the rewrite step

$$f(f(b, a), f(a, a)) \xrightarrow{\subseteq} f(a, a)$$

applying rule (4.3) if we can proof the condition  $f(b, a) \subseteq f(a, a)$ . Indeed, this can be done applying rule (4.2).

Unfortunately, linearization alone does not save us of having to generate critical atoms by overlapping rewrite rules below variable positions. Look at the following counter-example.

**4.7.2 Example.** We add a third rewrite rule to the two rewrite rules above:

$$\begin{array}{lcl} f(x, x) & \xrightarrow{\subseteq} & x \\ a & \xrightarrow{\supseteq} & b \\ a & \xrightarrow{\supseteq} & c \end{array}$$

After linearizing we obtain the following system

$$f(y, x) \xrightarrow{\subseteq} x \quad \text{if } y \subseteq x \quad (4.5)$$

$$f(x, y) \xrightarrow{\subseteq} x \quad \text{if } y \subseteq x \quad (4.6)$$

$$a \xrightarrow{\supseteq} b \quad (4.7)$$

$$a \xrightarrow{\supseteq} c \quad (4.8)$$

These rewrite rules do not generate critical atoms due to non-variable overlaps (standard critical atoms). But the system is not complete, because though  $f(b, c) \subseteq a$  is a theorem in the underlying inclusion theory, it is not provable

with a rewrite proof: We cannot apply neither rule (4.5) nor rule (4.6), because we cannot proof their respective conditions:

$$\begin{array}{ll} f(b, c) \xrightarrow[\subseteq]{} c & \text{only if } b \subseteq c \\ f(b, c) \xrightarrow[\subseteq]{} b & \text{only if } c \subseteq b \end{array}$$

**4.7.3 Contextual critical atoms.** Overlaps on variable positions are still necessary for generating critical atoms, even if the rules are left-linear, because the conditions make them in general non-convergent. For instance, rules (4.5) and (4.8) form critical atom by unifying  $a$  with  $x$ , thus generating the following contextual critical atom:

$$f(y, c) \subseteq a \quad \text{if } y \subseteq a$$

But in order to make this critical atom converge, by applying the same rules that generated it, we need that

if we can proof the condition of rule (4.5) with the substitution of  $x$  by  $a$  (which is the left-hand side of rule (4.8)),

then we should be able to proof condition of rule (4.5) with the substitution of  $x$  by  $c$  (which is the right-hand side of rule (4.8)).

Unfortunately, this does not hold: From  $y \subseteq a$  and  $a \supseteq c$  we cannot imply  $y \subseteq c$ . Again, if the binary relation was symmetric, for example taking equality ( $=$ ) instead of inclusion ( $\subseteq$ ), the above implication would hold, and critical atom generation, overlapping on variable positions, would not be necessary, because all contextual critical atoms would always converge.

**4.7.4 Saturation.** In the example above we avoid the problem of generating critical atoms due to the overlap on variable positions by joining all rules obtained from linearization into one unique conditional rule:

$$f(y, z) \subseteq x \quad \text{if } y \subseteq x \wedge z \subseteq x ,$$

But then it is obvious that conditional rewriting techniques are not applicable any more, and a general saturation calculus based on chaining, at least for Horn theories, should be taken under consideration. If this will lead us to obtain some significant gain over dealing with context variables is not clear at first sight.

From example 4.7.2 we also conclude that we might be able to proof the inclusion  $f(b, c) \subseteq a$  with the linearized system, if the binary relation was total (which would mean that either  $b \subseteq c$  or  $c \subseteq b$ , which is not the case with inclusion). Thus, maybe linearization together with conditional rewriting suffice for theories with total orderings. Actually, Bachmair and Ganzinger presented ordered chaining calculi for theories with total orderings (Bachmair and Ganzinger,



1994a), but with the additional restriction of having an ordering without endpoints, and considering only non-monotonic function symbols.

Of course, all these ideas deserve a thorough and detailed analysis in the future, not only for the special relation  $\subseteq$ , but also for general binary relations as studied in this thesis. Instead, in the next part, we are going to use the tools developed in these last two chapters —the logic of special relations and the notion of term rewriting along binary relations— to study computational issues of particular specification frameworks. We believe that the logic and its term rewriting based calculus suite well for describing the pragmatics of special relations in specification and reasoning. We shall see its potential on some examples. But we want to emphasize that we will not use the full expressiveness of the logic and the term rewriting mechanism, but only part of it. This already suffices to highlight the gain achieved with this new reasoning tool.



## Part III

# Computations in Specification Frameworks



## Chapter 5

# Specification Frameworks with Special Relations

We have defined a logic of special relations and have explored the practicability of a term rewriting based proof calculus, in order to lay down a framework for investigating several other frameworks of algebraic specification. Since we want to look at several logics of specification as particular instances of our logic of special relations, so that later, in chapter 6, we can analyze how the general calculus of term rewriting along special relations translates to each particular framework, we first need to show that our logic of special relation indeed suites well as a semantic framework.

Therefore, we give a general axiomatization of our logic of special relations, but also of membership equational logic (Bouhoula et al., 1997a), rewriting logic (Meseguer, 1992), and specifications with set relations (Kriaučiukas and Walicki, 1995), and we show that there exist *conservative maps of logics*, as defined in (Meseguer, 1989), between the latter three and the logic of special relations. The existence of these maps supports the validity of our logic of special relations as a general framework, and we claim that the role played by special relations in specifications is better highlighted in our logic than, for instance, in first-order logic. In addition, and since in chapter 4 we have described a rewriting-based calculus, by interpreting theories in the logic of special relations as term rewriting systems, we will be able to analyze, in chapter 6, computational issues of these three specification paradigms from such general term rewriting perspective.

Our framework can be made general enough to accommodate many other specification paradigms, with the purpose of studying rewriting based proof calculi for them. For instance, unified algebras (Mosses, 1989), equational type logic (Manca et al., 1990), or the calculus of refinements (Levy, 1995) could, in principle, be mapped to our logic of special relations, or to minor extensions of it, too, though a careful and detailed study is necessary.

## 5.1 General Logics

Having logics rigorously axiomatized allows one to study how different logics are related, or at least some specific components of these logics, such as their entailment systems and proof calculi. We can then speak of a space of logics, where maps preserve the essential properties, such as deduction, logical consequence, structure of proofs, initiality of models, and completeness of its entailment system.

In the following, we resume the basic ideas underlying the general axiomatization of logics, in order to cover their basic components: syntax, entailment, satisfaction, and proof. For further details, we refer to (Meseguer, 1989).

**5.1.1 Entailment system.** An *entailment system* describes, given a signature  $\Omega$ , the entailment relation between a set of  $\Omega$ -sentences  $\Gamma$  and a  $\Omega$ -sentence  $\varphi$ , denoted  $\Gamma \vdash_{\Omega} \varphi$ .

**5.1.2 Definition.** An *entailment system* is a tuple  $(\mathbf{Sign}, sen, \vdash)$ , such that

- **Sign** is a category of signatures;
- $sen : \mathbf{Sign} \rightarrow \mathbf{Set}$  is a functor assigning to each  $\Omega$  a set of  $\Omega$ -sentences;
- $\vdash$  is a function assigning to each  $\Omega$  a binary relation  $\vdash_{\Omega} \subseteq \mathcal{P}(sen(\Omega)) \times sen(\Omega)$ , such that,
  1. for all  $\varphi \in sen(\Omega)$ ,  $\{\varphi\} \vdash_{\Omega} \varphi$  (reflexivity),
  2. if  $\Gamma \vdash_{\Omega} \varphi$  and  $\Gamma \subseteq \Gamma'$  then  $\Gamma' \vdash_{\Omega} \varphi$  (monotonicity),
  3. if  $\Gamma \vdash_{\Omega} \varphi$  and  $\Gamma \cup \{\varphi\} \vdash_{\Omega} \psi$  then  $\Gamma \vdash_{\Omega} \psi$  (transitivity),
  4. if  $\Gamma \vdash_{\Omega} \varphi$ , then for all signature morphism  $H : \Omega \rightarrow \Omega'$  of **Sign**,  $sen(H)(\Gamma) \vdash_{\Omega'} sen(H)(\varphi)$  ( $\vdash$ -translation).

**5.1.3 Definition.** Given a category of signatures **Sign**, its category **Th** of *theories* has as objects  $T = (\Omega, \Gamma)$ , where  $\Omega$  is a signature, and  $\Gamma \subseteq sen(\Omega)$ . A theory morphism  $H : (\Omega, \Gamma) \rightarrow (\Omega', \Gamma')$  is a signature morphism  $H : \Omega \rightarrow \Omega'$ , such that  $sen(H)(\Gamma) \subseteq \Gamma'$ <sup>1</sup>.

**5.1.4 Institution.** Within this approach, the model-theoretic aspect is captured by the notion of *institution* of Goguen and Burstall (Goguen and Burstall, 1984). An *institution* describes, given a signature  $\Omega$ , the logical consequence relation  $\models$  between an  $\Omega$ -model  $\mathbf{Mod}(\Omega)$  and an  $\Omega$ -sentence  $\varphi$ , denoted  $\mathbf{Mod}(\Omega) \models \varphi$ .

---

<sup>1</sup>This is not the most general definition Meseguer gives for **Th**; I'm actually only interested in *axiom-preserving* theory morphisms.

**5.1.5 Definition.** An *institution* is a tuple  $(\mathbf{Sign}, sen, \mathbf{Mod}, \models)$ , such that

- $\mathbf{Sign}$  is a category of signatures;
- $sen : \mathbf{Sign} \rightarrow \mathbf{Set}$  is a functor assigning to each  $\Omega$  a set of  $\Omega$ -sentences;
- $\mathbf{Mod} : \mathbf{Sign}^{op} \rightarrow \mathbf{Cat}$  is a contravariant functor assigning to each  $\Omega$  a category of  $\Omega$ -models;
- $\models$  is a function assigning to each  $\Omega$  a binary relation  $\models_\Omega \subseteq |\mathbf{Mod}(\Omega)| \times sen(\Omega)$ , such that, for all signature homomorphisms  $H : \Omega \rightarrow \Omega'$  and models  $M' \in |\mathbf{Mod}(\Omega')|$ ,

$$\mathbf{Mod}(H)(M') \models_\Omega \varphi \text{ if and only if } M' \models_{sen(H)(\varphi)} .$$

**5.1.6 Logic.** A *logic* covers both, the provability and the model-theoretic sides. It is therefore the combination of an entailment system and an institution, with the additional notions of *soundness* and *completeness* relating both parts:

**5.1.7 Definition.** A *logic* is a tuple  $(\mathbf{Sign}, sen, \mathbf{Mod}, \vdash, \models)$ , where

- $(\mathbf{Sign}, sen, \vdash)$  is an entailment system;
- $(\mathbf{Sign}, sen, \mathbf{Mod}, \models)$  is an institution,

satisfying the following *soundness* property: For each signature  $\Omega$  of  $\mathbf{Sign}$  and  $\varphi \in sen(\Omega)$ ,

$$\Gamma \vdash_\Omega \varphi \text{ implies } \Gamma \models_\Omega \varphi .$$

In addition, if  $\Gamma \models_\Omega \varphi$  implies  $\Gamma \vdash_\Omega \varphi$ , the logic is *complete*.

**5.1.8 Map of logics.** Recall that the purpose of axiomatizing logics was to study how different logics are related, by speaking of a space of logics, where maps preserve their essential properties. We will be interested in mapping logics, which itself means mapping both components of the logics, namely their entailment systems and their institutions.

**5.1.9 Definition.** A *map of entailment system*  $(\Phi, \alpha) : (\mathbf{Sign}, sen, \vdash) \longrightarrow (\mathbf{Sign}', sen', \vdash')$  consists of

- a natural transformation  $\alpha : sen \xrightarrow{\cdot} sen' \cdot \Phi$  and
- an  $\alpha$ -sensible<sup>2</sup> functor  $\Phi : \mathbf{Th} \rightarrow \mathbf{Th}'$ ,

such that

$$\Gamma \vdash_\Omega \varphi \text{ implies } \alpha_\Omega(\Gamma) \vdash'_{\Phi(\Omega, \emptyset)} \alpha_\Omega(\varphi) .$$

The map  $(\Phi, \alpha)$  is *conservative* if the previous implication is an equivalence.

---

<sup>2</sup> $\Phi$  is  $\alpha$ -sensible when all theorems of  $\Phi(\Omega, \Gamma)$  are completely determined by  $\Phi(\Omega, \emptyset)$  and  $\alpha_\Omega(\Gamma)$ .

**5.1.10 Definition.** A map of institutions  $(\Phi, \alpha, \beta) : (\mathbf{Sign}, sen, \mathbf{Mod}, \models) \longrightarrow (\mathbf{Sign}', sen', \mathbf{Mod}', \models')$  consists of

- a natural transformation  $\alpha : sen \xrightarrow{\cdot} sen' \cdot \Phi$ ,
- an  $\alpha$ -sensible functor  $\Phi : \mathbf{Th} \rightarrow \mathbf{Th}'$ , and
- a natural transformation  $\beta : \mathbf{Mod}' \cdot \Phi^{op} \xrightarrow{\cdot} \mathbf{Mod}$ ,

such that, for each  $\Omega \in \mathbf{Sign}$ ,  $\varphi \in sen(\Omega)$ , and  $M' \in \mathbf{Mod}'(\Phi(\Omega, \emptyset))$ ,

$$M' \models_{\Phi(\Omega, \emptyset)} \alpha_{\Omega}(\varphi) \text{ if and only if } \beta_{(\Omega, \emptyset)}(M') \models_{\Omega} \varphi .$$

**5.1.11 Definition.** A map of logics  $(\Phi, \alpha, \beta) : (\mathbf{Sign}, sen, \mathbf{Mod}, \vdash, \models) \longrightarrow (\mathbf{Sign}', sen', \mathbf{Mod}', \vdash', \models')$  consists of

- a natural transformation  $\alpha : sen \xrightarrow{\cdot} sen' \cdot \Phi$ ,
- an  $\alpha$ -sensible functor  $\Phi : \mathbf{Th} \rightarrow \mathbf{Th}'$ , and
- a natural transformation  $\beta : \mathbf{Mod}' \cdot \Phi^{op} \xrightarrow{\cdot} \mathbf{Mod}$ ,

such that

- $(\Phi, \alpha) : (\mathbf{Sign}, sen, \vdash) \longrightarrow (\mathbf{Sign}', sen', \vdash')$  is a map of entailment systems, and
- $(\Phi, \alpha, \beta) : (\mathbf{Sign}, sen, \mathbf{Mod}, \models) \longrightarrow (\mathbf{Sign}', sen', \mathbf{Mod}', \models')$  is a map of institutions.

## 5.2 The Logic of Special Relations

We are going to axiomatize the logic of special relations introduced in chapter 3 within Meseguer's framework of general logics. For this purpose, we define its entailment system and its institution.

**5.2.1 Entailment system.** Signatures of the logic of special relations, as defined in 3.2.1, form a category  $\mathbf{Sign}^{\text{LSR}}$ , by taking as signature homomorphisms pairs made out of a function  $H_1 : S \longrightarrow S'$  between special relations, and a function  $H_2 : \Sigma \longrightarrow \Sigma'$  between function symbols, preserving the partially ordered ‘monoid-with-anti-involution’ structure, and the ranked ‘alphabet-with-monotonicity-properties’ structure, respectively. This category of signatures, together with the functor  $sen^{\text{LSR}} : \mathbf{Sign}^{\text{LSR}} \longrightarrow \mathbf{Set}$  assigning to each signature  $\Omega$  the set of all well-formed  $\Omega$ -sentences, as defined in 3.2.5, and the function  $\vdash^{\text{LSR}}$  assigning to each signature  $\Omega$  the entailment relation  $\vdash_{\Omega}^{\text{LSR}} \subseteq \mathcal{P}(sen^{\text{LSR}}(\Omega)) \times sen^{\text{LSR}}(\Omega)$ , as defined in 3.2.7, all together form an *entailment system*  $(\mathbf{Sign}^{\text{LSR}}, sen^{\text{LSR}}, \vdash^{\text{LSR}})$ . We denote with  $\mathbf{Th}^{\text{LSR}}$  the category of theories in the logic of special relations.



**5.2.2 Institution.** Let  $\Omega$  be a signature in the logic of special relations. Models over  $\Omega$ , as defined in 3.3.11, form a category  $\mathbf{Mod}^{\text{LSR}}(\Omega)$ , by taking as homomorphisms relators<sup>3</sup> between quasi-allegories whose restriction to their respective subcategories of functions are Cartesian closed functors.  $\mathbf{Mod}^{\text{LSR}} : (\mathbf{Sign}^{\text{LSR}})^{op} \longrightarrow \mathbf{Cat}$  is then the contravariant functor mapping signature homomorphisms to reduct functors between their respective categories of models. This functor  $\mathbf{Mod}^{\text{LSR}}$ , together with the category of signatures  $\mathbf{Sign}^{\text{LSR}}$  and the functor  $sen^{\text{LSR}}$  discussed above in 5.2.1, and together with the function  $\models^{\text{LSR}}$  assigning to each signature  $\Omega$  the satisfaction relation  $\models_{\Omega}^{\text{LSR}} \subseteq |\mathbf{Mod}^{\text{LSR}}(\Omega)| \times sen^{\text{LSR}}(\Omega)$ , as defined in 3.3.11, all together form an *institution*  $(\mathbf{Sign}^{\text{LSR}}, sen^{\text{LSR}}, \mathbf{Mod}^{\text{LSR}}, \models^{\text{LSR}})$ .

## 5.3 Mapping Membership Equational Specifications

We start by mapping *membership equational specifications* (Bouhoula et al., 1997a) to our logic of special relation, since it is the paradigm that inspired us to look at specifications from the perspective of a logic of special relations. This is the example we will treat in detail. Since the subsequent examples follow a similar pattern, we will discuss certain aspects of their respective maps more superficially.

In order to ease the following presentation, we are going to consider only the unkinded case, i.e., when  $\mathcal{K} = \{K\}$  is a singleton set. The following discussion can easily be extended to cover the many-kinded case. We refer to (Meseguer, 1998) for further details. To gain on generality, we map membership equational logic into a logic of special relations with conditional sentences, as discussed in section 3.4.

**5.3.1 Membership equational logic.** Signature are triples  $\Omega = ((\mathcal{K}, \Sigma), \{S_K\}_{K \in \mathcal{K}})$ , where

- $(\mathcal{K}, \Sigma)$  is a many-kinded signature (here one-kinded),
- $\{S_K\}_{K \in \mathcal{K}}$  is a family of sets of sorts (for us it will be a unique set).

Atomic formulae are expressions  $t : s$  or  $t \approx t'$ , where  $t, t' \in \mathcal{T}_{\Sigma}(\mathcal{X})$  and  $s \in S_K$ . Sentences are expressions

$$A \text{ if } B_1 \wedge \cdots \wedge B_n \quad n \geq 0 ,$$

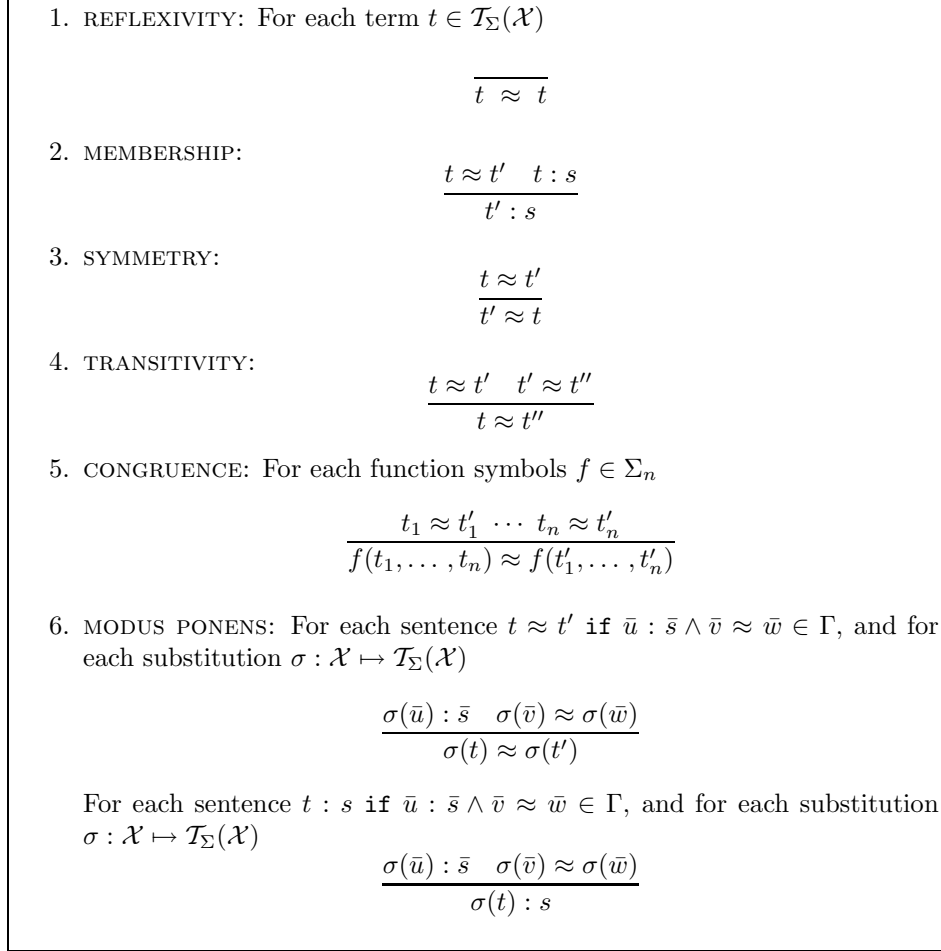
where  $A$  and  $B_i$ ,  $i = 1 \dots n$ , are atomic formulae. Sentences of the particular form

$$x : s_1 \text{ if } x : s_2 ,$$

where  $x \in \mathcal{X}$ , are called *subsort sentences*, because they induce a subsort relation  $\leq$  over the sorts of  $S_K$ . Theories are pairs  $T = (\Omega, \Gamma)$ , where  $\Omega$  is a signature and

---

<sup>3</sup>Relators are monotonic functors that preserve converse (see e.g., (Mitchell and Scedrov, 1993; Bird and de Moor, 1997) for further details).

**Figure 5.1:** Inference rules of membership equational logic

$\Gamma$  is a set of sentences. Sometimes we abbreviate sequences of terms  $t_1, \dots, t_n$  with  $\bar{t}$ , leaving  $n$  implicit. This kind of abbreviation also extends to sequences of atomic formulae, such as  $\bar{s} \approx \bar{t}$ . Given a theory  $T = (\Omega, \Gamma)$ , we define the entailment  $\Gamma \vdash_\Omega^{\text{MEL}} A$ ,  $A$  being an atomic sentence, by the inference rules of figure 5.1.

Signatures, sentences, and entailment of membership equational logic form together the entailment system  $(\mathbf{Sign}^{\text{MEL}}, \text{sen}^{\text{MEL}}, \vdash^{\text{MEL}})$ . We will denote with  $\mathbf{Th}^{\text{MEL}}$  the category of membership equational theories.

Given a signature  $\Omega = ((\mathcal{K}, \Sigma), \{S_K\}_{K \in \mathcal{K}})$ , a model in membership equational logic is an  $\Omega$ -algebra, i.e., a  $(\mathcal{K}, \Sigma)$ -algebra  $\mathcal{A}$  together with an assignment to each  $s \in S_K$ ,  $K \in \mathcal{K}$ , of a subset  $\mathcal{A}_s \subseteq \mathcal{A}_K$ . An  $\Omega$ -algebra  $\mathcal{A}$  satisfies the sentence

1.  $\varphi = t \approx t' \text{ if } \bar{u} : \bar{s} \wedge \bar{v} \approx \bar{w}$

$$2. \varphi = t : s \text{ if } \bar{u} : \bar{s} \wedge \bar{v} \approx \bar{w}$$

$n, m \geq 0$ , and we write  $\mathcal{A} \models_{\Omega}^{\text{MEL}} \varphi$  if and only if, for each  $\mathcal{K}$ -sorted assignment  $\rho : \mathcal{X} \rightarrow \mathcal{A}$ , such that  $\llbracket \bar{u} \rrbracket_{\rho} \in \mathcal{A}_{\bar{s}}$  and  $\llbracket \bar{v} \rrbracket_{\rho} = \llbracket \bar{w} \rrbracket_{\rho}$ , then

1.  $\llbracket t \rrbracket_{\rho} = \llbracket t' \rrbracket_{\rho}$
2.  $\llbracket t \rrbracket_{\rho} \in \mathcal{A}_s$

where  $\llbracket \cdot \rrbracket_{\rho} : \mathcal{T}_{\Sigma}(\mathcal{X}) \rightarrow \mathcal{A}$  is the unique  $\Sigma$ -homomorphism extending  $\rho$ . Signatures, sentences, models, and logical consequence within membership equational logic form together the institution  $(\mathbf{Sign}^{\text{MEL}}, \text{sen}^{\text{MEL}}, \mathbf{Mod}^{\text{MEL}}, \models^{\text{MEL}})$ .

**5.3.2 Map of entailment systems.** We map an atomic formula in membership equational logic, i.e., an equality  $t \approx t'$  or a membership  $t : s$ , to the formulae  $t \approx t'$  and  $t : s$  in the logic of special relations, where symbols  $\approx$  and  $:$  now stand for special relations, for which we will have to specify their properties explicitly, by means of the relation-algebra structure, and also by means of the monotonicity properties of the function symbols. We write  $\alpha(t \approx t') = t \approx t'$  and  $\alpha(t : s) = t : s$ . We map a subsort sentence  $x : s \text{ if } x : s'$  to the atomic formula  $s' \leq s$ , where  $\leq$  stands for a special relation, too. We will also need to specify how  $\approx$ ,  $:$  and  $\leq$  relate to each other. All this will become clear as we look at how theories are mapped.

We map a membership equational theory  $(\Omega, \Gamma)$ , with  $\Omega = ((\mathcal{K}, \Sigma), \{S_K\}_{K \in \mathcal{K}})$ , to a many-sorted theory  $(\Omega', \Gamma')$  in the logic of special relation, where

- $\Omega' = (M, (\mathcal{S}^*, \Sigma'))$ ,
  - $M = \{s_1, s_2\}$  is a set of two sort symbols,
  - $\mathcal{S}^* = (S^*, ;, 1', \smile, \sqsubseteq)$  is the smallest partially ordered free monoid with anti-involution generated over the set  $S = \{\approx, :, \leq\}$  satisfying:
 
$$\begin{aligned} 1' \sqsubseteq \approx & \quad \approx \sqsubseteq \approx & \quad \approx ; \approx \sqsubseteq \approx \\ 1' \sqsubseteq \leq & \quad \leq ; \leq \sqsubseteq \leq \\ \approx ; : \sqsubseteq : & \quad : ; \leq \sqsubseteq : \end{aligned}$$
  - $\Sigma' = \Sigma \cup \mathcal{K} \cup S_K$  is an alphabet of function symbols, such that all  $f \in \Sigma$  have rank  $f : s_1 \times \cdots \times s_1 \rightarrow s_1$ , and all  $c \in \mathcal{K} \cup S_K$  have rank  $c : \rightarrow s_2$ . Furthermore for all  $f$  in  $\Sigma$ , all argument positions  $i$  in  $f$  are monotonic with respect to  $(\approx, \approx)$ .
- $\Gamma' = \alpha(\Gamma) \cup \{s \leq s' \mid \forall t \in \mathcal{T}_{\Sigma}(\mathcal{X}) \quad (\Gamma \vdash t : s \Rightarrow \Gamma \vdash t : s')\}$ , such that
  - $\alpha(A) = A$ , if it is not a subsort sentence
  - $\alpha(x : s_1 \text{ if } x : s_2) = s_2 \leq s_1$

We write  $\Phi(\Omega, \Gamma) = (\Omega', \Gamma')$ . Unfortunately,  $\Phi : \mathbf{Th}^{\text{MEL}} \longrightarrow \mathbf{Th}^{\text{LSR}}$  is not an  $\alpha$ -sensible functor as required by definition 5.1.9. We need to sacrifice  $\alpha$ -sensibility of the functor, so that its natural transformation  $\alpha$  does not only depend on the signatures, but also on the theories, otherwise we would not be able to have a conservative map of institutions (see proposition 5.3.5).

**5.3.3 Proposition.** *The map  $(\Phi, \alpha) : (\mathbf{Sign}^{\text{MEL}}, \text{sen}^{\text{MEL}}, \vdash^{\text{MEL}}) \longrightarrow (\mathbf{Sign}^{\text{LSR}}, \text{sen}^{\text{LSR}}, \vdash^{\text{LSR}})$  is a conservative map of entailment systems.*

PROOF: We first proof that  $\Gamma \vdash^{\text{MEL}} \varphi$  implies  $\alpha(\Gamma) \vdash^{\text{LSR}} \alpha(\varphi)$ <sup>4</sup>. In fact, the inferences of membership equational logic given in figure 5.1 can be emulated by a series of inferences in the logic of special relations given in figure 3.2:

1. REFLEXIVITY:

$$\frac{\overline{t \ 1' \ t}}{t \approx t}$$

2. MEMBERSHIP:

$$\frac{\frac{\frac{t \approx t'}{t' \approx t} \quad t : s}{t' \approx t}}{t' \approx; : s} \quad \frac{}{t' : s}$$

3. SYMMETRY:

$$\frac{\frac{t \approx t'}{t' \approx t}}{t' \approx t}$$

4. TRANSITIVITY:

$$\frac{\frac{t \approx t' \quad t' \approx t''}{t \approx; \approx t''}}{t \approx t''}$$

The emulation of CONGRUENCE is a little bit trickier and is shown in figure 5.2. MODUS PONENS is emulated directly by REPLACEMENT (see section 3.4), except for following special case:

6. MODUS PONENS with  $x : s_2$  if  $x : s_1$

$$\frac{\frac{t : s_1 \quad s_1 \leq s_2}{t :: \leq s_2}}{t : s_2}$$

Successive MODUS PONENS inferences can be emulated in two ways:

• either

$$\frac{\frac{\frac{t : s_1 \quad s_1 \leq s_2}{t :: \leq s_2} \quad s_2 \leq s_3}{t : s_2}}{t :: \leq s_3} \quad \frac{}{t : s_3}$$

---

<sup>4</sup>The reader familiar with Meseguer's framework of general logics will have noticed that we have dropped the subscripts of  $\alpha$ ,  $\vdash^{\text{MEL}}$  and  $\vdash^{\text{LSR}}$  as the signatures are clear from the context.

• or

$$\frac{\frac{\frac{t : s_1 \quad \frac{s_1 \leq s_2 \quad s_2 \leq s_3}{s_1 \leq s_3}}{s_1 \leq s_3}}{t :: \leq s_3}}{t : s_3}$$

The pragmatics pushes me to emulate such successive MODUS PONENS inferences in the second way, i.e., through the transitivity of the subsort relation  $\leq$ .

To prove that this map is conservative, we need to show that  $\alpha(\Gamma) \vdash^{\text{LSR}} \alpha(\varphi)$  implies  $\Gamma \vdash^{\text{MEL}} \varphi$ . In fact, those inferences in our logic with special relations that have translated atomic formulae as conclusions and translated sentences as premises can be emulated by a succession of several inferences in membership equational logic. Conclusions  $\alpha(\varphi)$ , being  $\varphi$  atomic, can only be of the form  $s \approx t$  or  $s : t$ , but not  $s \leq t$ .

1. Inferences starting with translated sentences and concluding with an equality:

(a)

$$\frac{\frac{t \quad 1' \quad t}{t \approx t}}$$

(b)

$$\frac{\frac{t \approx s}{s \approx t}}{s \approx t}$$

(c)

$$\frac{\frac{s \approx u \quad u \approx t}{s \approx; \approx t}}{s \approx t}$$

(d) For every  $f \in \Sigma_n$ ,  $i = 1 \dots n$

$$\frac{s \approx t}{f(\dots, \overset{i)}{s}, \dots) \approx f(\dots, \overset{i)}{t}, \dots)}$$

(e)

$$\frac{\sigma(\bar{u}) \approx \sigma(\bar{v}) \quad \sigma(\bar{w}) : \sigma(\bar{r})}{\sigma(s) \approx \sigma(t)}$$

whenever  $s \approx t$  if  $\bar{u} \approx \bar{v} \wedge \bar{w} : \bar{r} \in \alpha(\Gamma)$

2. Inferences starting with translated sentences and concluding with a membership:

5. CONGRUENCE:

$$\begin{array}{c}
 \frac{t_1 \approx t'_1}{f(t_1, t_2, \dots, t_n) \approx f(t'_1, t_2, \dots, t_n)} \quad \frac{t_2 \approx t'_2}{f(t'_1, t_2, \dots, t_n) \approx f(t'_1, t'_2, \dots, t_n)} \\
 \hline
 f(t_1, t_2, \dots, t_n) \approx_i \approx f(t'_1, t'_2, \dots, t_n) \\
 \hline
 f(t_1, t_2, \dots, t_n) \approx f(t'_1, t'_2, \dots, t_n) \\
 \hline
 \dots \\
 \vdots \\
 f(t_1, \dots, t_{n-1}, t_n) \approx f(t'_1, \dots, t'_{n-1}, t_n) \\
 \hline
 f(t_1, \dots, t_{n-1}, t_n) \approx_i \approx f(t'_1, \dots, t'_{n-1}, t'_n) \\
 \hline
 f(t_1, \dots, t_{n-1}, t_n) \approx f(t'_1, \dots, t'_{n-1}, t'_n)
 \end{array}
 \quad \dots \quad
 \frac{t_n \approx t'_n}{f(t'_1, \dots, t'_{n-1}, t_n) \approx f(t'_1, \dots, t'_{n-1}, t'_n)}$$

Figure 5.2: CONGRUENCE emulation

(a)

$$\frac{\frac{s \approx t \quad t : u}{s \approx; : u}}{s : u}$$

(b)

$$\frac{s : t \quad \frac{\frac{t \leq v \quad v \leq u}{t \leq; \leq u}}{t \leq u}}{s :: \leq u} \\ \hline s : u$$

(c)

$$\frac{\sigma(\bar{u}) \approx \sigma(\bar{v}) \quad \sigma(\bar{w}) : \sigma(\bar{r})}{\sigma(s) : \sigma(t)}$$

whenever  $s : t$  if  $\bar{u} \approx \bar{v} \wedge \bar{w} : \bar{r} \in \alpha(\Gamma)$ 

Each of these inferences can be emulated in membership equational logic in the following way:

1. (a)

$$\overline{t \approx t}$$

(b)

$$\frac{t \approx s}{s \approx t}$$

(c)

$$\frac{s \approx u \quad u \approx t}{s \approx t}$$

(d) For every  $f \in \Sigma_n$ ,  $i = 1 \dots n$ 

$$\frac{\overline{u_1 \approx u_1} \quad \dots \quad s \overset{i)}{\approx} t \quad \dots \quad \overline{u_n \approx u_n}}{f(u_1, \dots, \overset{i)}{s}, \dots, u_n) \approx f(u_1, \dots, \overset{i)}{t}, \dots, u_n)}$$

(e)

$$\frac{\sigma(\bar{u}) \approx \sigma(\bar{v}) \quad \sigma(\bar{w}) : \sigma(\bar{r})}{\sigma(s) \approx \sigma(t)}$$

because  $s \approx t$  if  $\bar{u} \approx \bar{v} \wedge \bar{w} : \bar{r} \in \alpha(\Gamma)$ 

2. (a)

$$\frac{\frac{s \approx t}{t \approx s} \quad t : u}{s : u}$$

(b)

$$\frac{\frac{s : t}{s : v}}{s : u}$$

MODUS PONENS with  $x : v$  if  $x : t$  and with  $x : u$  if  $x : v$ 

(c)

$$\frac{\sigma(\bar{u}) \approx \sigma(\bar{v}) \quad \sigma(\bar{w}) : \sigma(\bar{r})}{\sigma(s) : \sigma(t)}$$

because  $s : t$  if  $\bar{u} \approx \bar{v} \wedge \bar{w} : \bar{r} \in \alpha(\Gamma)$  and is not of the form  $x : v$  if  $x : t$ 

□

**5.3.4 Map of institutions.** For the sake of simplicity, we will only treat the map of the unconditional fragment of membership equational logic (except for subsort sentences) into the unconditional fragment of the logic of special relations, because, in section 6.1, this is the case we are interested in. For how a map of institutions involving conditional sentences is done, see the one described in 7.2.10.

Let  $(\Phi, \alpha)$  be a map of entailment system as discussed in 5.3.2, and let  $T = (\Omega, \Gamma)$  be a membership equational theory. A model  $\mathbf{A}$  of its corresponding theory  $\Phi(T)$  in the logic of special relations is a quasi-allegory whose subcategory of functions  $\text{Fun}(\mathbf{A})$  is Cartesian closed, with

- two objects  $\llbracket s_1 \rrbracket$  and  $\llbracket s_2 \rrbracket$  corresponding to the two sorts of  $M$ ,
- a function  $\llbracket f \rrbracket : \llbracket s_1 \rrbracket^n \longrightarrow \llbracket s_1 \rrbracket$  for each  $f \in \Sigma_n$ ,
- a function  $\llbracket c \rrbracket : \mathbf{1} \longrightarrow \llbracket s_2 \rrbracket$  for each  $c \in \mathcal{K} \cup S_K$ ,
- arrows  $\llbracket \approx \rrbracket : \llbracket s_1 \rrbracket \longrightarrow \llbracket s_1 \rrbracket$ ,  $\llbracket \cdot \rrbracket : \llbracket s_1 \rrbracket \longrightarrow \llbracket s_2 \rrbracket$ , and  $\llbracket \leq \rrbracket : \llbracket s_2 \rrbracket \longrightarrow \llbracket s_2 \rrbracket$ ,

such that

$$id_{\llbracket s_1 \rrbracket} \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket^\circ \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket \approx \rrbracket \quad (5.1)$$

$$id_{\llbracket s_2 \rrbracket} \leq \llbracket \leq \rrbracket \quad \llbracket \leq \rrbracket \cdot \llbracket \leq \rrbracket \leq \llbracket \leq \rrbracket \quad (5.2)$$

$$\llbracket \cdot \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket \cdot \rrbracket \quad \llbracket \leq \rrbracket \cdot \llbracket \cdot \rrbracket \leq \llbracket \cdot \rrbracket \quad (5.3)$$

and for all  $n \in \mathbb{N}$ ,  $f \in \Sigma_n$ , and  $i \in [1 \dots n]$ ,

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \approx \rrbracket) \leq \llbracket \approx \rrbracket \cdot \llbracket f \rrbracket \quad (5.4)$$

and for each sentence in  $\Gamma$  of the kind

1.  $t \approx t'$ , we have  $\llbracket t' \rrbracket \leq \llbracket \approx \rrbracket \cdot \llbracket t \rrbracket$
2.  $t : s$ , we have  $\llbracket s \rrbracket \leq \llbracket \cdot \rrbracket \cdot \llbracket t \rrbracket$
3.  $x : s'$  if  $x : s$ , we have  $\llbracket s' \rrbracket \leq \llbracket \leq \rrbracket \cdot \llbracket s \rrbracket$



and for each induced subsort assertion  $s \leq s'$  (see 5.3.2), we have  $\llbracket s' \rrbracket \leq \llbracket \leq \rrbracket \cdot \llbracket s \rrbracket$ .

Arrow  $\llbracket \approx \rrbracket$  induces a relation  $\sim$  between arrows (functions)  $f, g$  in  $Fun(\mathbf{A})$ , by saying that

$$f \sim g \text{ if and only if } g \leq \llbracket \approx \rrbracket \cdot f \text{ in } \mathbf{A}.$$

It can be proved that  $\sim$  is a congruence relation by means of inequations (5.1). Let  $\mathcal{A}$  be the homset  $Hom(\mathbf{1}, \llbracket s_1 \rrbracket)$  in  $Fun(\mathbf{A}) / \sim$ . It can be proved that  $\mathcal{A}$  is a  $(\mathcal{K}, \Sigma)$ -algebra by means of inequations (5.4). We now can assign to each sort symbol  $s \in S_K$  the set

$$\mathcal{A}_s = \{[t] \in \mathcal{A} \mid t \leq \llbracket : \rrbracket^\circ \cdot \llbracket s \rrbracket\}^5,$$

which is, of course, a subset of  $\mathcal{A}$ .

We map the model of  $\Phi(T)$ , i.e., quasi-allegory  $\mathbf{A}$ , to the  $\Omega$ -algebra  $\mathcal{A}$ , and write  $\beta(\mathbf{A}) = \mathcal{A}$ , where  $\beta : \mathbf{Mod}^{\text{LSR}} \cdot \Phi^{op} \rightarrow \mathbf{Mod}^{\text{RL}}$  is a natural transformation.

**5.3.5 Proposition.** *The map  $(\Phi, \alpha, \beta) : (\mathbf{Sign}^{\text{MEL}}, sen^{\text{MEL}}, \mathbf{Mod}^{\text{MEL}}, \models^{\text{MEL}}) \rightarrow (\mathbf{Sign}^{\text{LSR}}, sen^{\text{LSR}}, \mathbf{Mod}^{\text{LSR}}, \models^{\text{LSR}})$  is a map of institutions.*

PROOF: Indeed  $\beta(\mathbf{A}) \models^{\text{MEL}} \varphi$  if and only if  $\mathbf{A} \models^{\text{LSR}} \alpha(\varphi)$ <sup>6</sup>:

- If  $\varphi = t \approx t'$  is an equality in membership equational logic,  $\beta(\mathbf{A}) \models^{\text{MEL}} t \approx t'$  if and only if arrows  $\llbracket t \rrbracket, \llbracket t' \rrbracket : \mathbf{1} \rightarrow \llbracket s_1 \rrbracket$  are in the same equivalence class of  $\sim$ , if and only if  $\llbracket t' \rrbracket \leq \llbracket \approx \rrbracket \cdot \llbracket t \rrbracket$ .
- If  $\varphi = t : s$  is a membership in membership equational logic,  $\beta(\mathbf{A}) \models^{\text{MEL}} t : s$  if and only if arrow  $\llbracket t \rrbracket$  is in  $\mathcal{A}_s$ , if and only if  $\llbracket t \rrbracket \leq \llbracket : \rrbracket^\circ \cdot \llbracket s \rrbracket$ .
- If  $\varphi = x : s' \text{ if } x : s$  is a subsort sentence in membership equational logic we have  $\beta(\mathbf{A}) \models^{\text{MEL}} x : s' \text{ if } x : s$  if and only if for all valuations  $\rho : \mathcal{X} \rightarrow \mathcal{A}$ , we have that

$$\rho(x) \in \mathcal{A}_s \text{ implies } \rho(x) \in \mathcal{A}_{s'},$$

and this is true if and only if  $\llbracket s \rrbracket \leq \llbracket \leq \rrbracket \cdot \llbracket s' \rrbracket$ .

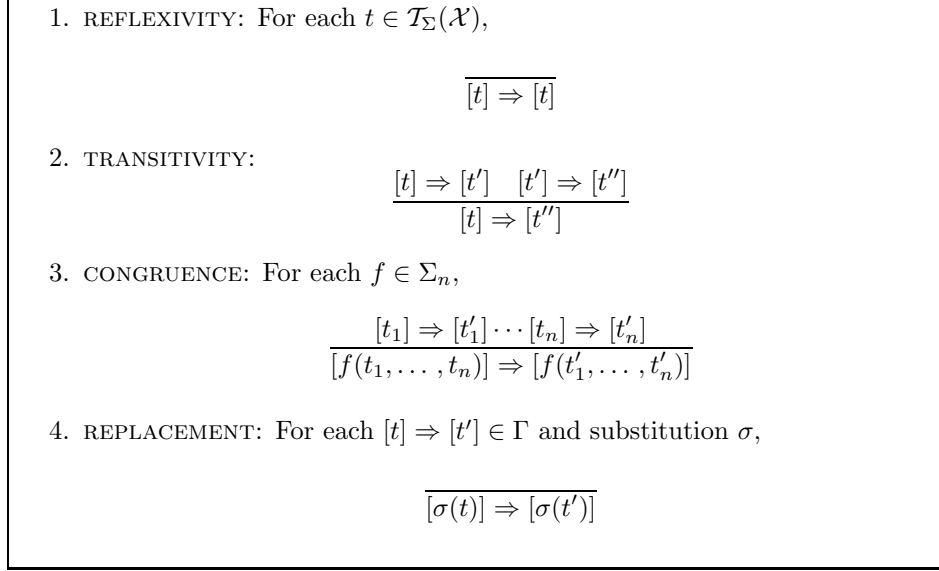
□

## 5.4 Mapping Rewriting Specifications

In this section, we show how *rewriting logic* is captured within the logic of special relations, by means of a conservative map of logics. Rewriting logic is the inherent logic underlying rewrite systems (Meseguer, 1992), and has turned out

<sup>5</sup>Recall that  $\mathcal{A}$  is a set of equivalence classes.

<sup>6</sup>Again, the reader familiar with Meseguer's framework of general logics will have noticed that we have dropped the subscripts of  $\alpha, \beta, \models^{\text{MEL}}$  and  $\models^{\text{LSR}}$ .

**Figure 5.3:** Inference rules of rewriting logic

to be suitable as a logical and semantic framework (Martí-Oliet and Meseguer, 1993), and several system implementations are based on it, like **Maude** (Clavel et al., 1996), **ELAN** (Borovansky et al., 1996), and **CafeOBJ** (Diaconescu et al., 1998). In order to ease the following discussion we will be concerned only with the unsorted, unconditional, and unlabeled fragment of rewriting logic. We refer to (Meseguer, 1992) for further details.

**5.4.1 Rewriting logic.** Signatures in rewriting logic are tuples  $\Omega = (\Sigma, E)$ , such that  $\Sigma$  is a ranked set of function symbols and  $E$  is a set of  $\Sigma$ -equations, and sentences are expressions  $[t]_E \Rightarrow [t']_E$ , also called *rewrite rules*, where  $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$  and  $[\ ]_E$  (or simply  $[\ ]$ ) denotes the  $E$ -equivalence class. A rewrite theory is a pair  $T = (\Omega, \Gamma)$ , where  $\Omega = (\Sigma, E)$  is a signature and  $\Gamma$  is a set of rewrite rules. The entailment  $\Gamma \vdash_\Omega^{\text{RL}} \varphi$  is defined by the inference rules of figure 5.3.

Signatures, sentences, and entailment of rewriting logic form together the entailment system  $(\mathbf{Sign}^{\text{RL}}, \text{sen}^{\text{RL}}, \vdash^{\text{RL}})$ . We will denote with  $\mathbf{Th}^{\text{RL}}$  the category of rewrite theories.

Recall from (Meseguer, 1992) that, given a rewrite signature  $(\Sigma, E)$ , a model in rewriting logic is a category  $\mathcal{S}$ , together with a  $(\Sigma, E)$ -algebra structure given by the family of functors  $\{\llbracket f \rrbracket_{\mathcal{S}} : \mathcal{S}^n \rightarrow \mathcal{S} \mid f \in \Sigma_n, n \in \mathbb{N}\}$  satisfying the equations in  $E$ . Recall also that a sentence  $[t] \Rightarrow [t']$  is satisfied by such a model, when there exists a natural transformation  $\eta : \llbracket t \rrbracket_{\mathcal{S}} \xrightarrow{\cdot} \llbracket t' \rrbracket_{\mathcal{S}}$  in  $\mathcal{S}$ . We write  $\mathcal{S} \models^{\text{RL}} [t] \Rightarrow [t']$ . Signatures, sentences, models, and satisfaction within rewriting logic form together the institution  $(\mathbf{Sign}^{\text{RL}}, \text{sen}^{\text{RL}}, \mathbf{Mod}^{\text{RL}}, \models^{\text{RL}})$ .

**5.4.2 Map of entailment systems.** We map a rewrite rule  $[s] \Rightarrow [t]$  to the sentence  $s \Rightarrow t$  in the logic of special relations, and write  $\alpha([s] \Rightarrow [t]) = s \Rightarrow t$ , where  $\alpha : \text{sen}^{\text{RL}} \xrightarrow{\cdot} \text{sen}^{\text{LSR}} \cdot \Phi$  is a natural transformation. The symbol  $\Rightarrow$  denotes now a special relation, and we will have to specify its properties explicitly<sup>7</sup>, by means of the relation-algebra structure, and also by means of the monotonicity properties of the function symbols. This is given in detail below, as we discuss the map of theories.

We capture the equivalence class structure of  $[s]$  and  $[t]$  by handling the equations in  $E$  defining them as sentences in the logic of special relations at the same level as rewrite rules. We will therefore deal with an additional special relation  $\approx$ , for which we have to specify its properties explicitly, too<sup>8</sup>. We will also need to specify how  $\approx$  and  $\Rightarrow$  relate to each other. This will become clear as we look at how theories are mapped.

We map a rewrite theory  $(\Omega, \Gamma)$  to a theory  $(\Omega', \Gamma')$  in the logic of special relations, where

- $\Omega' = (\mathcal{S}^*, \Sigma)$ ,
  - $\mathcal{S}^* = (S^*, ;, I', \smile, \sqsubseteq)$  being the smallest partially ordered free monoid with anti-involution generated over the set  $S = \{\Rightarrow, \approx\}$  satisfying:
 
$$\begin{array}{llll} I' \sqsubseteq \approx & \approx; \approx \sqsubseteq \approx & \smile \sqsubseteq \approx \\ I' \sqsubseteq \Rightarrow & \Rightarrow; \Rightarrow \sqsubseteq \Rightarrow & \approx; \Rightarrow \sqsubseteq \Rightarrow & \Rightarrow; \approx \sqsubseteq \Rightarrow \end{array}$$
  - $\Sigma$  being a ranked alphabet of function symbols, such that for all  $f$  in  $\Sigma$ , all argument positions  $i$  in  $f$  are monotonic with respect  $(\approx, \approx)$  and with respect to  $(\Rightarrow, \Rightarrow)$
- $\Gamma' = \alpha(\Gamma) \cup E$ .

We write  $\Phi(\Omega, \Gamma) = (\Omega', \Gamma')$ , where  $\Phi : \mathbf{Th}^{\text{RL}} \longrightarrow \mathbf{Th}^{\text{LSR}}$  is an  $\alpha$ -sensible functor.

**5.4.3 Remark.** Notice that since properties of relation  $\Rightarrow$  (and of  $\approx$ ) are explicitly stated by means of a partial order between relational expressions, and by means of polarities of argument positions of function symbols, we could theoretically vary its properties, depending on our needs. We could, for instance, specify monotonicity of particular function symbols with respect to  $\Rightarrow$  only for *some* of their argument positions, in order to deal with a slightly different kind of ‘rewrite relation’, as noticed in Remark 4.1.10.

**5.4.4 Proposition.** *The map  $(\Phi, \alpha) : (\mathbf{Sign}^{\text{RL}}, \text{sen}^{\text{RL}}, \vdash^{\text{RL}}) \longrightarrow (\mathbf{Sign}^{\text{LSR}}, \text{sen}^{\text{LSR}}, \vdash^{\text{LSR}})$  is a conservative map of entailment systems.*

<sup>7</sup>Recall that in rewriting logic  $\Rightarrow$  is a reflexive, transitive, and an under substitutions and context application stable binary relation.

<sup>8</sup>In this case  $\approx$  is a reflexive, transitive, symmetric, and an under substitutions and context application stable binary relation.

PROOF:  $\Gamma \vdash^{\text{RL}} \varphi$  implies  $\alpha(\Gamma) \cup E \vdash^{\text{LSR}} \alpha(\varphi)$ , because each inference rule of rewriting logic given in figure 5.3 is captured by a finite application of inferences rules of the logic of special relations of figure 3.2, in a similar way as seen in the proof of Proposition 5.3.3. Thus, REPLACEMENT is captured by the AXIOM inference, REFLEXIVITY by the IDENTITY plus the PARTIAL ORDER ( $I' \sqsubseteq \Rightarrow$ ) inferences, TRANSITIVITY by the COMPOSITION plus the PARTIAL ORDER ( $\Rightarrow; \Rightarrow \sqsubseteq \Rightarrow$ ) inferences, and CONGRUENCE by several MONOTONICITY, COMPOSITION, and PARTIAL ORDER ( $\Rightarrow; \Rightarrow \sqsubseteq \Rightarrow$ ) inferences. By additional COMPOSITION inferences (with equality axioms) and necessary PARTIAL ORDER inferences, we capture that in the original rewrite theory the inferences involve equivalent classes of terms.

$\alpha(\Gamma) \cup E \vdash^{\text{LSR}} \alpha(\varphi)$  implies  $\Gamma \vdash^{\text{RL}} \varphi$ , because every derivation with inference rules of figure 3.2 starting with premises being sentences with binary relation  $\Rightarrow$  or  $\approx$  only, and ending with conclusions being sentences with binary relation  $\Rightarrow$  only, are equivalent to derivations with inference rules of figure 5.3, also in a similar way as seen in the proof of Proposition 5.3.3.  $\square$

**5.4.5 Map of institutions.** Let  $(\Phi, \alpha)$  be a map of entailment system as discussed in section 5.4.2, and let  $T = (\Omega, \Gamma)$  be a rewrite theory. A model  $\mathbf{A}$  of its corresponding theory  $\Phi(T)$  in the logic of special relations is a quasi-allegory whose subcategory of functions  $\text{Fun}(\mathbf{A})$  is Cartesian closed, with

- an object  $\llbracket s \rrbracket$  for the *unique sort* of the theory (recall that we are concerned with the unsorted case here),
- a function  $\llbracket f \rrbracket : \llbracket s \rrbracket^n \longrightarrow \llbracket s \rrbracket$  for each  $f \in \Sigma_n$ , and
- arrows  $\llbracket \approx \rrbracket : \llbracket s \rrbracket \longrightarrow \llbracket s \rrbracket$  and  $\llbracket \Rightarrow \rrbracket : \llbracket s \rrbracket \longrightarrow \llbracket s \rrbracket$ ,

such that

$$id_{\llbracket s \rrbracket} \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket^\circ \leq \llbracket \approx \rrbracket \quad (5.5)$$

$$id_{\llbracket s \rrbracket} \leq \llbracket \Rightarrow \rrbracket \quad \llbracket \Rightarrow \rrbracket \cdot \llbracket \Rightarrow \rrbracket \leq \llbracket \Rightarrow \rrbracket \quad \llbracket \approx \rrbracket \cdot \llbracket \Rightarrow \rrbracket \leq \llbracket \Rightarrow \rrbracket \quad \llbracket \Rightarrow \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket \Rightarrow \rrbracket \quad (5.6)$$

and for all  $n \in \mathbb{N}$ ,  $f \in \Sigma_n$ , and  $i \in [1 \dots n]$ ,

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \approx \rrbracket) \leq \llbracket \approx \rrbracket \cdot \llbracket f \rrbracket \quad (5.7)$$

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \Rightarrow \rrbracket) \leq \llbracket \Rightarrow \rrbracket \cdot \llbracket f \rrbracket \quad (5.8)$$

and for each equation  $t \approx t' \in E$ , we have that  $\llbracket t' \rrbracket \leq \llbracket \approx \rrbracket \cdot \llbracket t \rrbracket$ , and for each rewrite rule  $[t] \Rightarrow [t'] \in \Gamma$ , we have that  $\llbracket t' \rrbracket \leq \llbracket \Rightarrow \rrbracket \cdot \llbracket t \rrbracket$ .

Arrow  $\llbracket \approx \rrbracket$  induces a relation  $\sim$  between arrows (functions)  $f, g$  in  $\text{Fun}(\mathbf{A})$ , by saying that

$$f \sim g \quad \text{if and only if} \quad g \leq \llbracket \approx \rrbracket \cdot f \quad \text{in } \mathbf{A} .$$

It can be proved that  $\sim$  is a congruence relation by means of inequations (5.5) and (5.7). Arrow  $\llbracket \Rightarrow \rrbracket$  induces morphisms  $\psi : [f] \longrightarrow [g]$  between two congruence

classes of functions  $[f]$  and  $[g]$  in the quotient category  $Fun(\mathbf{A})/\sim$ , whenever  $g \leq [\Rightarrow] \cdot f$  in  $\mathbf{A}$ . Let  $\mathcal{S}$  be the category whose object are the congruence classes of functions  $[f] : \mathbf{1} \rightarrow [s]$  in  $Fun(\mathbf{A})/\sim$ , and whose arrows are the morphisms just defined. It can be proved that  $\mathcal{S}$  is indeed a category by means of inequations (5.6). Each term  $t(\bar{x}^m) \in \mathcal{T}_\Sigma(\mathcal{X})$  induces a functor  $F_t : \mathcal{S}^m \rightarrow \mathcal{S}$ , such that

$$F_t([f_1], \dots, [f_m]) \stackrel{\text{def}}{=} [[t] \cdot \langle f_1, \dots, f_m \rangle],$$

where  $[f_1], \dots, [f_m]$  are objects in  $\mathcal{S}$  (i.e., they are congruence classes of function  $\mathbf{1} \rightarrow [s]$  in  $Fun(\mathbf{A})/\sim$ ). It can be proved that  $F_t$  is indeed a functor by means of inequation (5.8). Furthermore, due to the congruence relation  $\sim$ , these functors form a  $(\Sigma, E)$ -algebra structure.

We map the model of  $\Phi(T)$ , i.e., quasi-allegory  $\mathbf{A}$ , to category  $\mathcal{S}$ , and write  $\beta(\mathbf{A}) = \mathcal{S}$ , where  $\beta : \mathbf{Mod}^{\text{LSR}} \cdot \Phi^{op} \xrightarrow{\cdot} \mathbf{Mod}^{\text{RL}}$  is a natural transformation.

**5.4.6 Proposition.** *The map  $(\Phi, \alpha, \beta) : (\mathbf{Sign}^{\text{RL}}, \text{sen}^{\text{RL}}, \mathbf{Mod}^{\text{RL}}, \models^{\text{RL}}) \rightarrow (\mathbf{Sign}^{\text{LSR}}, \text{sen}^{\text{LSR}}, \mathbf{Mod}^{\text{LSR}}, \models^{\text{LSR}})$  is a map of institutions.*

PROOF: Let  $[t(\bar{x}^m)] \Rightarrow [t'(\bar{x}^m)]$  be a rewrite rule. Indeed,  $\beta(\mathbf{A}) \models^{\text{RL}} [t(\bar{x}^m)] \Rightarrow [t'(\bar{x}^m)]$  if and only if there exists a natural transformation  $\eta : F_t \xrightarrow{\cdot} F_{t'}$ , between the functors  $F_t, F_{t'} : \beta(\mathbf{A})^m \rightarrow \beta(\mathbf{A})$  induced by terms  $t(\bar{x}^m)$  and  $t'(\bar{x}^m)$  as discussed above. It can be proved, using inequations (5.6), that this natural transformation exists if and only if  $\llbracket t'(\bar{x}^m) \rrbracket \leq [\Rightarrow] \cdot \llbracket t(\bar{x}^m) \rrbracket$ , i.e.,  $\mathbf{A} \models^{\text{LSR}} \alpha([t(\bar{x}^m)] \Rightarrow [t'(\bar{x}^m)])$ .  $\square$

## 5.5 Mapping Specifications with Set Relations

Our last example showing how specification paradigms are captured within our framework will be the map of *specifications with set relations* (Kriaučiukas and Walicki, 1995) to the logic of special relations. In this case, we will be concerned only with the model-theoretic aspect of the map, i.e., its map of institutions. Furthermore, and in order to ease the following discussion, we will deal only with unconditional, i.e., atomic, formulae. We refer to (Kriaučiukas and Walicki, 1995) for further details.

**5.5.1 Specifications with set relations.** Signatures are ranked alphabets of function symbols  $\Omega$ , and they form a category  $\mathbf{Sign}^{\text{SSR}}$  with arity preserving functions as homomorphism. Sentences are atomic formulae of the form  $t \approx t'$ ,  $t < t'$ , or  $t \frown t'$ , where  $t$  and  $t'$  are ground terms in  $\mathcal{T}_\Omega(\emptyset)$ . Let  $\text{sen}^{\text{SSR}} : \mathbf{Sign}^{\text{SSR}} \rightarrow \mathbf{Set}$  be the functor assigning to each signature the set of its well-formed sentences. A *specification* (or theory) is a pair  $T = (\Omega, \Gamma)$ , where  $\Omega$  is a signature and  $\Gamma$  is a set of sentences. We will denote with  $\mathbf{Th}^{\text{SSR}}$  the category of theories.

Recall from (Kriaučiukas and Walicki, 1995) that, given a signature  $\Omega$ , a model is a  $\Omega$ -multialgebra  $(\mathcal{A}, \mathcal{F})$ , where  $\mathcal{A}$  is a non-empty carrier set, and  $\mathcal{F}$

is a set of set-valued functions  $\llbracket f \rrbracket : \mathcal{A}^n \rightarrow \mathcal{P}^+(\mathcal{A})$ , where  $f \in \Omega$  and  $\mathcal{P}^+(\mathcal{A})$  is the power-set of  $\mathcal{A}$  with the empty set excluded. Recall also, that terms are interpreted as follows:

$$\llbracket f(t_1, \dots, t_n) \rrbracket = \bigcup \{ \llbracket f \rrbracket(e_1, \dots, e_n) \mid e_i \in \llbracket t_i \rrbracket \}$$

Such a model satisfies a sentence

- $t \approx t'$  if  $\llbracket t \rrbracket = \llbracket t' \rrbracket$  are singleton sets,
- $t < t'$  if  $\llbracket t \rrbracket \subseteq \llbracket t' \rrbracket$ ,
- $t \frown t'$  if  $\llbracket t \rrbracket \cap \llbracket t' \rrbracket \neq \emptyset$ .

We write  $(\mathcal{A}, \mathcal{F}) \models^{\text{SSR}} t \approx t'$ ,  $(\mathcal{A}, \mathcal{F}) \models^{\text{SSR}} t < t'$ , and  $(\mathcal{A}, \mathcal{F}) \models^{\text{SSR}} t \frown t'$ , respectively.  $\Omega$ -multialgebras form a category  $\mathbf{Mod}^{\text{SSR}}(\Omega)$ , thus signatures, sentences, models, and satisfaction within specifications with set relations form together the institution  $(\mathbf{Sign}^{\text{SSR}}, \text{sen}^{\text{SSR}}, \mathbf{Mod}^{\text{SSR}}, \models^{\text{SSR}})$ .

**5.5.2 Map of institutions.** We map sentences  $t \approx t'$ ,  $t < t'$ , and  $t \frown t'$ , to sentences  $t \approx t'$ ,  $t < t'$ , and  $t \frown t'$  of the logic of special relation, such that  $\approx$ ,  $<$ , and  $\frown$  denote now special relations, and, as with the previous two maps, we will have to specify their properties explicitly, by means of the relation-algebra structure, and also by means of the monotonicity properties of the function symbols. Again, this will become clear as we look at how theories (i.e., specifications) are mapped.

We map a specification with set relations  $(\Omega, \Gamma)$  to a theory  $(\Omega', \Gamma')$  in the logic of special relations, where

- $\Omega' = (\mathcal{S}^*, \Sigma)$ ,
  - $\mathcal{S}^* = (S^*, ;, I', \smile, \sqsubseteq)$  being the smallest partially ordered free monoid with anti-involution generated over the set  $S = \{\approx, <, \frown\}$  satisfying:

$$\begin{array}{llll} \smile \sqsubseteq \approx & \approx; \approx \sqsubseteq \approx & <; \approx \sqsubseteq \approx & \\ I' \sqsubseteq < & <; < \sqsubseteq < & \approx; \frown \sqsubseteq < & \approx \sqsubseteq < \\ I' \sqsubseteq \frown & \smile \sqsubseteq \frown & \frown; < \sqsubseteq \frown & < \sqsubseteq \frown \end{array}$$

- $\Sigma$  being the ranked alphabet of function symbols in  $\Omega$ , such that, for all  $f$  in  $\Sigma$ , all argument positions  $i$  in  $f$  are monotonic with respect to  $(<, <)$  and  $(\frown, \frown)$ , and they are, both, monotonic and antimonotonic with respect to  $(\approx, <)$
- $\Gamma' = \alpha(\Gamma)$ , where
  - $\alpha(t \approx t') = t \approx t'$
  - $\alpha(t < t') = t < t'$

$$- \alpha(t \frown t') = t \frown t'$$

We write  $\Phi(\Omega, \Gamma) = (\Omega', \Gamma')$ , where  $\Phi : \mathbf{Th}^{\text{SSR}} \longrightarrow \mathbf{Th}^{\text{LSR}}$  is an  $\alpha$ -sensible functor.

Let  $T = (\Omega, \Gamma)$  be a specification with set relations. A model  $\mathbf{A}$  of its corresponding theory  $\Phi(T)$  in the logic of special relations is a quasi-allegory whose subcategory of functions  $\text{Fun}(\mathbf{A})$  is Cartesian closed, with

- an object  $\llbracket s \rrbracket$  for the *unique sort*,
- a function  $\llbracket f \rrbracket : \llbracket s \rrbracket^n \longrightarrow \llbracket s \rrbracket$  for each  $f \in \Sigma_n$ , and
- arrows  $\llbracket \approx \rrbracket : \llbracket s \rrbracket \longrightarrow \llbracket s \rrbracket$ ,  $\llbracket < \rrbracket : \llbracket s \rrbracket \longrightarrow \llbracket s \rrbracket$ , and  $\llbracket \frown \rrbracket : \llbracket s \rrbracket \longrightarrow \llbracket s \rrbracket$ .

such that

$$\llbracket \approx \rrbracket^\circ \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket \approx \rrbracket \quad \llbracket \approx \rrbracket \cdot \llbracket < \rrbracket \leq \llbracket \approx \rrbracket \quad (5.9)$$

$$id_{\llbracket s \rrbracket} \leq \llbracket < \rrbracket \quad \llbracket < \rrbracket \cdot \llbracket < \rrbracket \leq \llbracket < \rrbracket \quad \llbracket \frown \rrbracket \cdot \llbracket \approx \rrbracket \leq \llbracket < \rrbracket \quad \llbracket \approx \rrbracket \leq \llbracket < \rrbracket \quad (5.10)$$

$$id_{\llbracket s \rrbracket} \leq \llbracket \frown \rrbracket \quad \llbracket \frown \rrbracket^\circ \leq \llbracket \frown \rrbracket \quad \llbracket < \rrbracket \cdot \llbracket \frown \rrbracket \leq \llbracket \frown \rrbracket \quad \llbracket < \rrbracket \leq \llbracket \frown \rrbracket \quad (5.11)$$

and for all  $n \in \mathbb{N}$ ,  $f \in \Sigma_n$ , and  $i \in [1 \dots n]$ ,

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket < \rrbracket) \leq \llbracket < \rrbracket \cdot \llbracket f \rrbracket \quad (5.12)$$

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \frown \rrbracket) \leq \llbracket \frown \rrbracket \cdot \llbracket f \rrbracket$$

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \approx \rrbracket) \leq \llbracket < \rrbracket \cdot \llbracket f \rrbracket$$

$$apply \cdot (curry_i(\llbracket f \rrbracket) \times \llbracket \approx \rrbracket) \leq \llbracket < \rrbracket^\circ \cdot \llbracket f \rrbracket$$

We explain how we get a multialgebra  $(\mathcal{A}, \mathcal{F})$  out of the quasi-allegory  $\mathbf{A}$ . We take as carrier set the quotient of the set of all arrows in  $\text{Fun}(\mathbf{A})$  that actually are specified to be singleton sets, with respect to the relation  $\sim$  between arrows induced by  $\llbracket \approx \rrbracket$ .

$$\mathcal{A} = \{a : \mathbf{1} \longrightarrow \llbracket s \rrbracket \text{ in } \text{Fun}(\mathbf{A}) \mid a \leq \llbracket \approx \rrbracket \cdot a\} / \sim$$

It follows from inequations (5.9) that  $\sim$  is indeed a congruence. For each  $f \in \Sigma_n$  we define a set-valued function  $F_f : \mathcal{A}^n \rightarrow \mathcal{P}^+(\mathcal{A})$  in  $\mathcal{F}$ , in such a way that, given an element  $x$  of the carrier set  $\mathcal{A}$ , it assigns to it the set of those elements of  $\mathcal{A}$  that correspond to arrows interpreted to be ‘below’  $f$  composed with  $x$  in the quasi-allegory (with respect to special relation  $<$ ):

$$F_f(x) \stackrel{\text{def}}{=} \{[y] \in \mathcal{A} \mid \llbracket f \rrbracket \cdot x \leq \llbracket < \rrbracket \cdot y\}^9 \quad (5.13)$$

Unfortunately, models based on quasi-allegories and models based on multi-algebras are not directly interchangeable. Therefore, in order to define a map of institutions, we need that specifications with set relations satisfy an additional condition. A similar situation was encountered by Levy and Agustí when

<sup>9</sup>Recall that  $\mathcal{A}$  is a set of equivalence classes.

developing a bi-rewriting based operational semantics for non-deterministic specifications (Levy and Agustí, 1992).

Thus, we map the model of  $\Phi(T)$ , i.e., quasi-allegory  $\mathbf{A}$ , to the multialgebra  $(\mathcal{A}, \mathcal{F})$  as constructed above, only if the translated theory  $\Phi(T)$  satisfies the following condition: For every two term  $t, t' \in \mathcal{T}_\Omega(\emptyset)$

$$\Gamma' \vdash^{\text{LSR}} t \frown t' \quad \text{implies} \quad \begin{cases} \Gamma' \vdash^{\text{LSR}} u \approx u \\ \Gamma' \vdash^{\text{LSR}} u < t \\ \Gamma' \vdash^{\text{LSR}} u < t' \end{cases} \quad (5.14)$$

for some term  $u \in \mathcal{T}_\Omega(\emptyset)$ . Then we write  $\beta(\mathbf{A}) = (\mathcal{A}, \mathcal{F})$ , where  $\beta : \mathbf{Mod}^{\text{LSR}} \cdot \Phi^{op} \dashrightarrow \mathbf{Mod}^{\text{SSR}}$  is a natural transformation.

We will use the following lemma in order to proof that, by this way of obtaining a multialgebra out of a quasi-allegory, we indeed have a map of institutions.

**5.5.3 Lemma.** *The interpretation  $T_t \subseteq \mathcal{A}$  of a ground term  $t \in \mathcal{T}_\Omega(\emptyset)$  in multialgebra  $(\mathcal{A}, \mathcal{F})$  is*

$$T_t = \{[y] \in \mathcal{A} \mid \llbracket t \rrbracket \leq \llbracket < \rrbracket \cdot y\} . \quad (5.15)$$

**PROOF:** By structural induction over the term  $t$  (we sketch it for unary function symbols):

1. If  $t$  is a constant: trivial by (5.13).
2. If  $t = f(t')$ , for  $f \in \Sigma$  and  $t' \in \mathcal{T}_\Omega(\emptyset)$ :

$$\begin{aligned} & F_{f(t')} \\ = & \quad \{\text{definition of multialgebra, see 5.5.1}\} \\ & \bigcup_{x \in T_{t'}} F_f(x) \\ = & \quad \{\text{induction hypothesis}\} \\ & \{[y] \in \mathcal{A} \mid \llbracket f \rrbracket \cdot x \leq \llbracket < \rrbracket \cdot y, \text{ where } x \in \mathcal{A} \wedge \llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot x\} \\ = & \quad \{\text{inequations (5.10) and (5.12)}\} \\ & \{[y] \in \mathcal{A} \mid \llbracket f \rrbracket \cdot \llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot y\} \end{aligned}$$

□

**5.5.4 Proposition.** *The map  $(\Phi, \alpha, \beta) : (\mathbf{Sign}^{\text{SSR}}, \text{sen}^{\text{SSR}}, \mathbf{Mod}^{\text{SSR}}, \models^{\text{SSR}}) \longrightarrow (\mathbf{Sign}^{\text{LSR}}, \text{sen}^{\text{LSR}}, \mathbf{Mod}^{\text{LSR}}, \models^{\text{LSR}})$  is a map of institutions.*



PROOF: We distinguish three cases:

1.  $\mathbf{A} \models^{\text{LSR}} t \approx t'$  if and only if  $\llbracket t' \rrbracket \leq \llbracket \approx \rrbracket \cdot \llbracket t \rrbracket$ . And this happens if and only if  $T_t$  and  $T_{t'}$  are equal singleton sets, i.e.,  $\beta(\mathbf{A}) \models^{\text{SSR}} t \approx t'$ , because, by definition of the map  $\beta$  and by inequations (5.9),

$$\begin{aligned} T_t &= \{[y] \in \mathcal{A} \mid \llbracket t \rrbracket \leq \llbracket < \rrbracket \cdot y\} = \{\llbracket t \rrbracket\} \\ T_{t'} &= \{[y] \in \mathcal{A} \mid \llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot y\} = \{\llbracket t' \rrbracket\} \end{aligned}$$

and  $\llbracket t \rrbracket = \llbracket t' \rrbracket$ .

2.  $\mathbf{A} \models^{\text{LSR}} t < t'$  if and only if

$$\llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot \llbracket t \rrbracket \quad (5.16)$$

and this happens if and only if  $[x] \in T_t$  implies  $[x] \in T_{t'}$ , because

$$\begin{aligned} &[x] \in T_t \\ \equiv &\quad \{\text{equation (5.15)}\} \\ &\llbracket t \rrbracket \leq \llbracket < \rrbracket \cdot x \\ \Rightarrow &\quad \{\text{inequation (5.16)}\} \\ &\llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot \llbracket < \rrbracket \cdot x \\ \Rightarrow &\quad \{\text{inequations (5.10)}\} \\ &\llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot x \\ \equiv &\quad \{\text{equation (5.15)}\} \\ &[x] \in T_{t'} \end{aligned}$$

i.e.,  $\mathbf{A} \models^{\text{LSR}} t < t'$  if and only if  $T_t \subseteq T_{t'}$  if and only if  $\beta(\mathbf{A}) \models^{\text{SSR}} t < t'$ .

3.  $\mathbf{A} \models^{\text{LSR}} t \frown t'$  if and only if

$$\llbracket t' \rrbracket \leq \llbracket \frown \rrbracket \cdot \llbracket t \rrbracket$$

and this happens if and only if there exists (by (5.11) and (5.14)) a  $[x] \in \mathcal{A}$ , such that

$$\begin{aligned} &\llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot x \\ &\llbracket t' \rrbracket \leq \llbracket < \rrbracket \cdot x \quad (\Leftrightarrow \quad x \leq \llbracket < \rrbracket^\circ \cdot \llbracket t' \rrbracket) \end{aligned}$$

i.e.,  $[x] \in T_t$  and  $[x] \in T_{t'}$  and we have  $\mathbf{A} \models^{\text{LSR}} t \frown t'$  if and only if  $T_t \cap T_{t'} \neq \emptyset$  if and only if  $\beta(\mathbf{A}) \models^{\text{SSR}} t \frown t'$ .

□

## 5.6 Conclusion

The pragmatics of special relations in membership equational logic, rewriting logic, and specifications with set relations has been highlighted by describing these frameworks as particular instances of our logic of special relations, and thus capturing the properties of special relations with the partial order of a relation-algebra structure. Furthermore, the abstract semantics based on quasi-allegories served us also to capture membership algebras,  $\mathcal{R}$ -systems, as well as specific multialgebras. By having established maps of entailment systems and maps of institutions between suitable axiomatizations of these logics, we are able now to put the subsequent analysis of chapter 6 on a firm formal ground, and consequently, we can safely apply particular aspects of our general perspective of term rewriting, introduced in chapter 4, to the study of some computational issues of these three specification paradigms.

## Chapter 6

# On Specific Computations

Our logic of special relations acts as a framework of several distinct specification paradigms. In this chapter, we show that this yields some advantages for analyzing certain computational issues of membership equational specifications, rewriting specifications, and specifications of set relations, from the perspective of term rewriting along binary relations, as defined in chapter 4.

In particular, we show that our framework unifies several notions that within the area of order-sorted term rewriting, have been considered separately, namely sort-decreasingness, confluence, and regularity. Furthermore, this allows us to weaken the sort-decreasingness requirement of membership equational theories, in a similar way as Eker and Meseguer did.

We further show a particular example of the possibility of effectively using the technique of term rewriting along binary relations in theories with monotonic function symbols—which, in general, is quite troublesome to automate—, by making use of the notion of polarity, and by attempting to present an alternative operational semantics for rewrite theories coming from Horn logic.

Finally, we revise the term-rewriting based proof calculus for specifications with set relations proposed by Kriaučiukas and Walicki (1995), and show how an erroneous notion of relative closeness falsifies their completeness theorem.

### 6.1 Order-sorted Term Rewriting

**6.1.1 Sort-decreasingness.** When considering completion procedures for order-sorted rewrite system, one has to face the problem that order-sorted replacement of equals by equals is not complete in general, and consequently, one has to pose the restriction of *sort-decreasingness* on the rewrite rules of the rewrite system (Gnaedig et al., 1988). A rewrite rule is sort-decreasing if the sort to which the right-hand side of the rule belongs, is subsort of the one to which the left-hand side belongs. This restriction extends to the completion process, being an new source of failure, in addition to unorientable equations.

Several unsatisfying and complicated ways to solve this problem have been

suggested (Gnaedig et al., 1988; Kirchner et al., 1988; Ganzinger, 1989; Comon, 1998), but it turns out that a semantic treatment of sorts provides an elegant solution to the problems posed by the sort-decreasingness requirement. For instance, within the paradigm of membership equational specifications, Bouhoula, Jouannaud, and Meseguer study a *Knuth-Bendix*-like completion procedure that avoids non-sort-decreasing rewrites, by adding semantic preserving membership assertions to the original theory presentation, in a way similar to adding semantic preserving equations when divergent critical pairs among rewrite rules arise (Bouhoula et al., 1997b).

Let us analyze this latter approach from the perspective of our logic of special relations.

**6.1.2 Membership equational specifications.** Recall from 5.3.1 that in general a signature  $\Omega = ((\mathcal{K}, \Sigma), \{S_K\}_{K \in \mathcal{K}})$  in membership equational logic consists of a many-kinded signature  $(\mathcal{K}, \Sigma)$  and a family  $\{S_K\}_{K \in \mathcal{K}}$  of sets of sorts; for the sake of simplicity we will be only interested in the ‘one-kinded’ fragment, i.e., when  $\mathcal{K}$  is a singleton set, and the family  $\{S_K\}_{K \in \mathcal{K}}$  consists of only one set of sorts. Recall, also, that atomic formulae are membership assertions  $t : s$  or equations  $t \approx t'$ , where  $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$  and  $s \in S_K$ , that sentences are expressions  $A \text{ if } B_1 \wedge \dots \wedge B_n$ ,  $n \geq 0$ , where  $A$  and  $B_i$ ,  $i = 1 \dots n$ , are atomic formulae, and that sentences of the particular form  $x : s_1 \text{ if } x : s_2$ , where  $x \in \mathcal{X}$ , are called *subsort sentences*, because they induce a subsort relation  $\leq$  over the sorts of  $S_K$ . From how we map membership equational theories into theories of the logic of special relations, we consider subsort sentences as atomic formulae of the form  $s_1 \leq s_2$ . For the subsequent discussion, we will treat the unconditional fragment of membership equational logic, with the exception of subsort sentences.

We have seen in 5.3.2 that a theory in membership equational logic is a particular theory in the logic with special relations, involving three different special relations  $S = \{\approx, :, \leq\}$  standing for ‘equality’, ‘membership’, and ‘subsort’ respectively, and where the relation-algebra structure is partially ordered by the minimal partial order such that

$$\begin{array}{lll} 1' \sqsubseteq \approx & \approx \sqsubseteq \approx & \approx ; \approx \sqsubseteq \approx \\ 1' \sqsubseteq \leq & \leq ; \leq \sqsubseteq \leq & \\ \approx ; : \sqsubseteq : & : ; \leq \sqsubseteq : & \end{array}$$

**6.1.3 Rewriting equality, membership, and subsort relations.** From how the map from membership equational logic to rewriting logic is defined (see section 5.3), there is a particular restriction put on sentences coming from membership equational logic, namely that only special relations in  $S$  and not general binary relations in  $S^*$  are used. Thus, in order to check for local confluence of a term rewriting system associated to a theory presentation, we consider overlaps on the left-hand sides of the following five possible distinct rewrite relations:

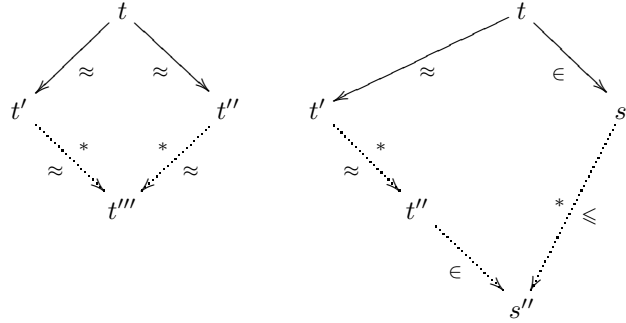
$$s \xrightarrow[\approx]{} t \quad s \xrightarrow[\in]{} t \quad s \xrightarrow[\exists]{} t \quad s \xrightarrow[\leq]{} t \quad s \xrightarrow[\geq]{} t$$

**6.1.4 Notation.** Notice that we use the symbol  $\in$  instead of  $:$  as type assignment, because it is a non-symmetric symbol like the special relation it denotes. For the subsequent discussion, and when we are concerned with rewriting, we will follow this convention, though in the logical sentences we will continue using  $:$  instead of  $\in$ .

**6.1.5 Term rewriting in membership equational specifications.** In the very special case of membership equational logic, suitable restrictions on the term ordering allow us to rule out some of the multiple cases of critical pairs we would otherwise take into account.

1. By requiring all operator symbols for term construction to precede in the ordering all sort constants we avoid rules of the form  $\xrightarrow{\supset}$ .
2. By requiring the ordering on sort constants to resemble the sort hierarchy, i.e.,  $s_2 \succ s_1$  whenever  $s_2 \geq s_1$ , we avoid rules of the form  $\xrightarrow{\leq}$ .

We only need to consider three rewrite relations,  $\xrightarrow{\approx}$ ,  $\xrightarrow{\in}$  and  $\xrightarrow{\geq}$ , and since function symbols for term construction are kept separate from sort symbols only the following two cases of local confluence need to be checked:



Furthermore, since sort symbols are only allowed to be constants in membership assertions, the unique polarized proper subterm positions are those with respect to  $(\approx, \approx)$ , and because of the symmetry of  $\approx$ , no variable instance pairs need to be considered (see the discussion in 4.6.3). Each of the previous two cases of local confluence correspond to the conventional notion on ‘local confluence’ —as traditionally known from rewriting along equality— and a weaker notion of *sort-decreasingness* as the one defined in (Bouhoula et al., 1997b). We will show the latter observation in more detail.

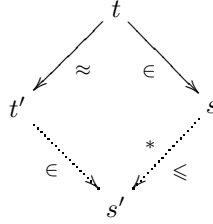
**6.1.6 Critical reduced membership.** In the framework of membership equational logic, sort-decreasingness is defined together with the notion of *critical reduced membership*, which we state here adapted to our framework, and also to the unconditional case.

**6.1.7 Definition.** Given rewrite rules  $t \xrightarrow[\in]{\quad} s$  and  $l \xrightarrow[\approx]{\quad} r$  and  $\sigma(t|_p) = \sigma(l)$  for some non-variable position  $p$  and most general unifier  $\sigma$ , then  $\sigma(t[r]_p) : s$  is a *critical reduced membership*.

A critical reduced membership is an actual critical atom as defined in definition 4.4.16 or 4.5.4, but with the additional restriction that only special relations in  $S$  may appear in sentences. This restriction comes from how membership equational logic is mapped into the logic of special relations (see section 5.3).

**6.1.8 Definition.** A critical reduced membership  $t' : s$  is *sort-decreasing* if there exists  $s'$  such that  $s \xrightarrow[\geq]{*} s'$  and  $t' \xrightarrow[\in]{\quad} s'$ .

Analyzing this definition within our framework, it is obvious that sort-decreasingness is actually too strong a condition for decidability of equality and membership statements, because of the unnecessarily required *one-step* rewrite  $t' \xrightarrow[\in]{\quad} s'$ :



We have seen, in section 4.3, that —given termination— local confluence suffices for decidability of atomic formulae in theories with special relations. In membership equational theories, the weaker sort-decreasingness condition that actually suffices for decidability of equality, membership and subsort assertions is the one stated in 6.1.5, and it is similar to Eker and Meseguer’s notion of *descendingness* (Eker and Meseguer, 1997), who, following a different approach than the one presented here, also suggested to weaken the sort-decreasingness requirement in membership equational theories.

**6.1.9 Regularity.** Eker and Meseguer’s *descendingness* differs from our general notion of local confluence in that it involves the notion of *least sort*, and hence assumes *strong regularity* of the signature. Regularity is another well-know restriction usually put on order-sorted signatures and assures the existence of a least sort for a given term in the hierarchy of sorts. Although not necessary for the decidability of equality, membership, and subsort assertions, membership equational logic takes regular and strongly regular signatures into account, for efficiency purposes.

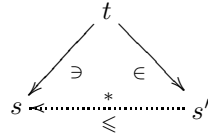
Actually, in the presence of strong regularity, Eker and Meseguer’s descendingness notion and our general local confluence are equivalent. This is easily proved within our framework, and constitutes an additional argument in favor

of the elegance of term rewriting along binary relations for the study of such issues, as we can see below.

**6.1.10 Definition.** A membership equational theory (and by extension the term rewriting system) is *strongly regular* if for each term  $t$  there exists a sort  $s$ , such that

1.  $t \xrightarrow{\in} s$ , and
2. if there exists a sort  $s'$  such that  $t \xrightarrow{\in} s'$ , then  $s' \xrightarrow[\geq]{*} s$ .

We say that  $s$  is the *least sort* of  $t$ :



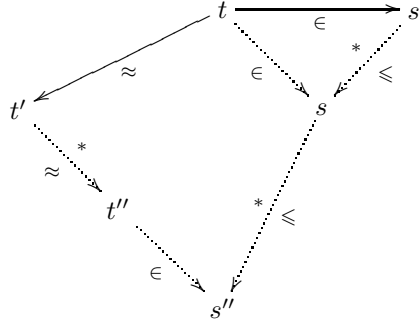
The following is an alternative definition of descendingness to the one given by Eker and Meseguer. It is given with respect to critical reduced memberships as defined in definition 6.1.7, for comparison with the definition 6.1.8 of sort-decreasingness.

**6.1.11 Definition.** A critical reduced membership  $t' : s$  from rewrite rules  $t \xrightarrow{\in} s$  and  $l \xrightarrow[\approx]{*} r$ , for which  $s$  is least sort of  $t$ , is *descending*, if there exists a term  $t''$  with least sort  $s'$  such that  $t' \xrightarrow[\approx]{*} t''$  and  $s \xrightarrow[\geq]{*} s'$ .

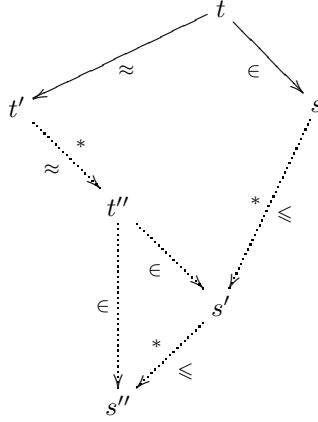
Now we are ready to prove the equivalence between descendingness and general local confluence, in the presence of strong regularity.

**6.1.12 Proposition.** *Given a strongly regular term rewriting system, all critical pairs between rules  $\xrightarrow[\approx]{*}$  and  $\xrightarrow{\in}$  are convergent, and thus the system is locally confluent, if and only if all critical reduced memberships are descending.*

PROOF: For the ‘if’ direction, let  $t' : s'$  be a critical atom formed from rewrite rules  $t \xrightarrow[\approx]{*} t'$  and  $t \xrightarrow{\in} s'$ , and let  $s$  be the least sort of  $t$ . Then  $t' : s$  is a critical reduced membership formed from rewrite rules  $t \xrightarrow[\approx]{*} t'$  and  $t \xrightarrow{\in} s$ , and, by descendingness, there exists a term  $t''$  with least sort  $s''$ , such that  $t' \xrightarrow[\approx]{*} t''$  and  $s \xrightarrow[\geq]{*} s''$ , and by strong regularity,  $s' \xrightarrow[\geq]{*} s$ . Therefore,  $t' : s'$  is locally confluent:



For the ‘only if’ direction, let  $t' : s$  be a critical reduced membership formed from rewrite rules  $t \xrightarrow{\approx} t'$  and  $t \xrightarrow{\in} s$ , where  $s$  is least sort of  $t$ . Then, by local confluence, there exists a term  $t''$  and a sort  $s'$ , such that  $t' \xrightarrow{*} t''$ ,  $t'' \xrightarrow{\in} s'$ , and  $s \xrightarrow{*} s'$ . By strong regularity,  $t''$  has a least sort  $s''$ , and therefore  $s' \xrightarrow{*} s''$ , consequently,  $t' : s$  is descending:



□

## 6.2 Bi-rewriting Equational and Horn Logic

In this section we apply the technique of term rewriting along binary relations as a ‘multi-paradigm’ proof calculus. We will sketch this idea on a couple of very intuitive examples, namely on equational logic and Horn logic, following maps of these logics into rewriting logic, based on the work of Martí-Oliet and Meseguer (1993).

We are going to present the proof calculi from the particular perspective of bi-rewriting, which is an instance of term rewriting along binary relations when dealing with a unique, reflexive, transitive, but non-symmetric, special relation.



We refer to 2.3.2 for notation and further details on bi-rewriting. Furthermore, we are going to use the extension bi-rewriting modulo an equational theory thoroughly studied in (Levy and Agustí, 1996).

Our attempt may appear strange at first sight, but our purpose is to show that proof calculi for equational and Horn logic are in fact specific cases of bi-rewrite systems, and that their peculiarities restrict in a significant way the general proof calculus based on rewriting binary relations. Furthermore, the presentation of a bi-rewriting based proof calculus for Horn theories serves also as an example to show how function symbols with different polarizations arise naturally in their corresponding theories in the logic of special relations. Because of these polarizations, an ordered *subterm* chaining calculus for Horn logic can be effectively applied (recall the discussion in section 4.5).

**6.2.1 Bi-rewriting equational logic.** Here we will be concerned with equational theories distinguishing regular axioms from structural axioms. Thus, an equational theory is described as a tuple  $((\Sigma, E), \Gamma)$ , where  $(\Sigma, E)$  is a signature consisting of a set  $\Sigma$  of function symbols and a set  $E$  of structural axioms ( $\Sigma$ -equations), and  $\Gamma$  is a set of equations of the form  $[s]_E = [t]_E$  between equivalence classes of terms. Notice that if  $E$  is the empty set, the equations in  $\Gamma$  are between terms.

An equational theory  $((\Sigma, E), \Gamma)$  is mapped to a rewrite theory  $((\Sigma, E), \Gamma')$ , such that for every equation  $[s] = [t]$  in  $\Gamma$ , two rules  $[s] \Rightarrow [t]$  and  $[t] \Rightarrow [s]$  are in  $\Gamma'$ , in order to make explicit the property of symmetry. The bi-rewrite system  $\langle \Gamma'_{\Rightarrow}, \Gamma'_{\Leftarrow} \rangle$  resulting from orienting the rules of  $\Gamma'$  has for every rule  $[s] \xrightarrow{\Rightarrow} [t]$  in  $\Gamma'_{\Rightarrow}$  also a rule  $[s] \xrightarrow{\Leftarrow} [t]$  in  $\Gamma'_{\Leftarrow}$ , i.e., each former equation appears as a rewrite rule in both rewrite systems.

**6.2.2 Example.** Let us consider the map of equational theory  $((\{+, s, 0\}, \emptyset), \Gamma)$  specifying the (non-associative/commutative) sum operator

$$\Gamma = \left\{ \begin{array}{ll} x + 0 & \approx x \\ x + s(y) & \approx s(x + y) \end{array} \right.$$

into rewrite theory  $((\{+, s, 0\}, \emptyset), \Gamma')$  given below:

$$\Gamma' = \left\{ \begin{array}{ll} x + 0 & \Rightarrow x \\ x & \Rightarrow x + 0 \\ x + s(y) & \Rightarrow s(x + y) \\ s(x + y) & \Rightarrow x + s(y) \end{array} \right.$$

Orienting the rules in  $\Gamma'$ , following e.g., a lexicographic path ordering based on the signature precedence  $+ \succ s \succ 0$ , we get the following bi-rewrite system:

$$\begin{aligned}\Gamma'_{\Rightarrow} &= \left\{ \begin{array}{ll} x + 0 & \longrightarrow x \\ x + s(y) & \xRightarrow{\Rightarrow} s(x + y) \end{array} \right. \\ \Gamma'_{\Leftarrow} &= \left\{ \begin{array}{ll} x + 0 & \xrightarrow{\Leftarrow} x \\ x + s(y) & \xRightarrow{\Leftarrow} s(x + y) \end{array} \right.\end{aligned}$$

**6.2.3 Standard equational term rewriting.** Because of symmetry, we actually are duplicating each rewrite rule. Since the generation of critical pairs is done by looking for overlaps between left-hand sides of two rules, one of each rewrite system, in this case this is equivalent to look for overlaps among the rules of one unique rewrite system, i.e., rules that actually rewrite on equations, as already stated in remark 2.3.16. When dealing with equational theories, bi-rewrite systems can be ‘simplified’ to standard rewrite systems, as for instance the following equational term rewrite system for the equational theory of example 6.2.2.

$$\Gamma = \left\{ \begin{array}{ll} x + 0 & \xrightarrow{\approx} x \\ x + s(y) & \xrightarrow{\approx} s(x + y) \end{array} \right.$$

Recall, also, from remark 2.3.18, that overlaps on variable positions are not needed, since all resulting peaks are convergent. If the equational rewrite system  $\Gamma$  is Church-Rosser, the bi-rewrite system  $\langle \Gamma'_{\Rightarrow}, \Gamma'_{\Leftarrow} \rangle$ , obtained from the set  $\Gamma'$  to which  $\Gamma$  is mapped to, is also Church-Rosser.

In the case  $E$  is non-empty, rewriting must be done modulo the set of axioms in  $E$ . As mentioned in 2.3.13, this has been thoroughly studied by the rewriting community, and their results were applied also to bi-rewrite systems (Levy and Agustí, 1996).

We already emphasized, in 2.3.19, that symmetry plays an important role, because when reasoning with equivalence relations, we can deal with the notion of *equivalence class*. Since we do not have two different rewrite systems any more, critical pairs are computed by overlapping left-hand sides of rules of one unique rewrite system. If such rewrite system is convergent, this has important practical consequences: Each term not only has an irreducible term, the so called *normal form*, but this normal form is also unique for each term. Rewriting is done within an equivalence class, and all the members of this class share the same normal form. A decision procedure for the word problem in equational theories, based on convergent rewriting systems, is much simpler than in arbitrary rewrite theories. Just the normal forms of the two terms of the equation we want to validate are computed and checked for identity. Furthermore, the property of *don't care* non-determinism of theorem proving in convergent equational theories is kept.

**6.2.4 Bi-rewriting Horn logic.** We now present bi-rewriting of Horn theories. It will serve to illustrate how different polarizations for function symbols with respect to special relations arise naturally when Horn theories are mapped into corresponding theories in rewriting logic, seen as a particular logic of special relations (see section 5.4). These polarizations allow us to effectively apply ordered chaining inferences also on subterm position. As with equational theories, we will be concerned with Horn theories that distinguish regular axioms from structural axioms.

Thus, a Horn theory is described as a tuple  $((\Sigma, \Pi, E), \Gamma)$ . The triple  $(\Sigma, \Pi, E)$  is the signature, and consists of a set  $\Sigma$  of function symbols, a set  $\Pi$  of predicate symbols, and a set  $E$  of structural axioms (i.e.,  $\Sigma$ -equations).  $\Gamma$  is a set of Horn clauses of the form  $[s]_E \leftarrow [t_1]_E, \dots, [t_n]_E$ . A Horn theory  $((\Sigma, \Pi, E), \Gamma)$  is mapped to a two-sorted rewrite theory  $(\Sigma \cup \Pi', E \cup E', \Gamma')$  with sorts **term** and **prop**. All functions symbols in  $\Gamma$  take arguments of sort **term** and are themselves of sort **term**, and set  $\Pi'$  contains a constant *true* of sort **prop**, a binary infix operator  $\wedge$  of sort **prop** taking as argument two elements of sort **prop**, and for each  $n$ -ary predicate  $p$  in  $\Pi$ , an  $n$ -ary function symbol  $p$  of sort **prop** taking as arguments  $n$  elements of sort **term**.  $E'$  is the set containing the laws of associativity, commutativity, and identity (with respect to constant *true*) of operator  $\wedge$ , and  $\Gamma'$  is the set of rules obtained by mapping each clause  $[s]_E \leftarrow [t_1]_E, \dots, [t_n]_E$  to the rule  $[s]_{E \cup E'} \Rightarrow [t_1 \wedge \dots \wedge t_n]_{E \cup E'}$ , and each unit clause  $[s]_E$  to the rule  $[s]_{E \cup E'} \Rightarrow [true]_{E \cup E'}$ .

**6.2.5 Example.** Horn theory  $((\{cesc, anna, jordi\}, \{parent, ancestor\}, \emptyset), \Gamma)$ —specifying the parent and ancestor relations— with the following clauses

$$\Gamma = \begin{cases} parent(cesc, anna) \\ parent(anna, jordi) \\ ancestor(x, y) \leftarrow parent(x, y) \\ ancestor(x, y) \leftarrow parent(x, z), ancestor(z, y) \end{cases}$$

is mapped to rewrite theory  $((\{cesc, anna, jordi, parent(-, -), ancestor(-, -), true, - \wedge -\}, E'), \Gamma')$  as follows,  $E'$  being the set defined above:

$$\Gamma' = \begin{cases} [parent(cesc, anna)]_{E'} \Rightarrow [true]_{E'} \\ [parent(anna, jordi)]_{E'} \Rightarrow [true]_{E'} \\ [ancestor(x, y)]_{E'} \Rightarrow [parent(x, y)]_{E'} \\ [ancestor(x, y)]_{E'} \Rightarrow [parent(x, z) \wedge ancestor(z, y)]_{E'} \end{cases}$$

**6.2.6 SLD-resolution is not bi-rewriting.** It is well-known that a proof calculus based on the resolution inference is adequate as operational semantics for Horn logic programming: Queries to a program are existentially quantified formulas  $\exists \bar{x} \, u_1, \dots, u_m$ <sup>1</sup>, and are solved by refuting its negation. A resolution

<sup>1</sup> $\bar{x}$  denotes the free variables of terms  $u_1, \dots, u_m$ .

step is then as follows<sup>2</sup>:

$$\frac{\leftarrow u_1, u_2, \dots, u_m \quad s \leftarrow t_1, \dots, t_n}{\leftarrow \sigma(t_1), \dots, \sigma(t_n), \sigma(u_2), \dots, \sigma(u_m)}$$

where  $\sigma$  is a most general unifier of  $u_1$  and  $s$ .

A query in its correspondent rewrite theory reads then  $\exists \bar{x} [u_1 \wedge u_2 \wedge \dots \wedge u_m]_{E'} \Rightarrow [true]_{E'}$ , which is solved also by refuting its negation. The inference step which corresponds to the above resolution step reads:

$$\frac{[u_1 \wedge u_2 \wedge \dots \wedge u_m] \not\Rightarrow [true] \quad [s] \Rightarrow [t_1 \wedge \dots \wedge t_n]}{[\sigma(t_1) \wedge \dots \wedge \sigma(t_n) \wedge \sigma(u_2) \wedge \dots \wedge \sigma(u_m)] \not\Rightarrow [true]} \quad (6.1)$$

where  $\sigma$  is, as before, a most general unifier of  $u_1$  and  $s$ . This inference step is actually a *negative chaining* step (see figure 2.1).

Since chaining is only done through the term on the left-hand side of the rule representing the negated query, until a term in the  $E'$ -equivalence class of  $true$  is reached, we can see this inference also as applying rule  $[s] \Rightarrow [t_1 \wedge \dots \wedge t_n]$  in order to *narrow*<sup>3</sup> the ‘query term’  $[u_1 \wedge u_2 \wedge \dots \wedge u_m]$ :

$$[u_1 \wedge u_2 \wedge \dots \wedge u_m] \rightsquigarrow [\sigma(t_1) \wedge \dots \wedge \sigma(t_n) \wedge \sigma(u_2) \wedge \dots \wedge \sigma(u_m)]$$

Here  $\sigma$  is, again, a most general unifier of  $u_1$  and  $s$ . This is the approach followed by C. Kirchner, H. Kirchner and Vittek (1995), who also studied the map of proofs in Horn theories to proofs in rewrite theories. They map Horn clauses to narrowing rules, and the proof-theoretic structure of Horn logic, based on SLD-resolution, is therefore captured by the straightforward application of the deduction rules of rewriting logic. They further add to the rewrite theory a notion of strategy, to efficiently compute with the given rewrite rules, and call such a rewrite theory plus strategy a *computational system*.

Negative chaining —inference (6.1) above— will be ordered if rules  $[s] \Rightarrow [t_1 \wedge \dots \wedge t_n]$  of the rewrite theory  $\Gamma'$  are oriented from left to right, i.e.,  $[s] \succ [t_1 \wedge \dots \wedge t_n]$ . Indeed, the operational behavior of query solving in Horn theories, following resolution strategies known from logic programming, such as Prolog’s SLD-resolution, is captured by the trivial bi-rewrite system  $\langle \Gamma'_{\Rightarrow}, \emptyset \rangle$ , where  $\Rightarrow \subseteq \longrightarrow$ . This bi-rewrite system is actually a standard rewrite system, since we are not rewriting in two directions, and its operational behavior corresponds to standard deduction in rewriting logic. But the ordering induced by these rules will not be, in general, a reduction ordering, and therefore this bi-rewrite system will be non-terminating.

When taking a reduction ordering on terms into account, the process of theorem proving in Horn logic maps to an ordered chaining inference tree. We will show this through an example.

<sup>2</sup>For the sake of simplicity this inference is shown for Horn theories with no structural axioms, i.e.,  $E = \emptyset$ .

<sup>3</sup>Narrowing was originally devised as an efficient E-unification procedure using convergent sets of rewrite rules (Hullot, 1980).

**6.2.7 Example.** If we orient the rules of the rewrite theory obtained in example 6.2.5 following e.g., a lexicographic path ordering based on the signature precedence

$$\wedge \succ \text{ancestor} \succ \text{parent} \succ \text{jordi} \succ \text{anna} \succ \text{cesc} \succ \text{true} ,$$

we get the following bi-rewrite system:

$$\begin{aligned} \Gamma'_{\Rightarrow} &= \left\{ \begin{array}{l} [\text{parent}(\text{cesc}, \text{anna})] \xrightarrow{\Rightarrow} [\text{true}] \\ [\text{parent}(\text{anna}, \text{jordi})] \xrightarrow{\Rightarrow} [\text{true}] \\ [\text{ancestor}(x, y)] \xrightarrow{\Rightarrow} [\text{parent}(x, y)] \end{array} \right. \\ \Gamma'_{\Leftarrow} &= \left\{ [\text{parent}(x, z) \wedge \text{ancestor}(z, y)] \xrightarrow{\Leftarrow} [\text{ancestor}(x, y)] \right. \end{aligned}$$

**6.2.8 Ordered chaining for Horn theories.** As said in 2.3.2, by orienting the rules of a rewrite theory by means of a reduction ordering on terms, critical pairs (or even variable instance pairs) among the rules of both rewrite systems may arise: We need to start a process of completion for proving theorems, by generating new rules, i.e., our proof calculus will be based on ordered chaining (see 2.4.6). It is interesting that the unique operator of the signature that is monotonic with respect to relation  $\Rightarrow$  is the conjunction operator  $\wedge$ . In other words,  $\wedge$  is the only function symbol whose argument positions are positive with respect to  $(\Rightarrow, \Rightarrow)$ . Consequently, the overlap required for generating new rules is only needed on whole propositions and not on terms within them. Furthermore, since the map of Horn to rewrite theories does not introduce variables as arguments of  $\wedge$ , unification on variable positions is not needed, and the intractable variable instance atom generation can be completely avoided.

Figure 6.1 shows the sequence of ordered chaining inferences (see figure 2.1 for details of each inference rule) for proving theorem  $\text{ancestor}(\text{cesc}, \text{jordi}) \Rightarrow \text{true}$  in the rewrite theory of example 6.2.5. The framed sentence is the negation of the theorem. All other premises are sentences of the rewrite theory or conclusions of previous inferences. Bold faced terms are the ones unified (i.e., chained through). For instance the top most inference step of figure 6.1 corresponds to the generation of a critical pair among rewrite rules  $\text{parent}(x, z) \wedge \text{ancestor}(z, y) \xrightarrow{\Leftarrow} \text{ancestor}(x, y)$  and  $\text{ancestor}(x', y') \xrightarrow{\Rightarrow} \text{parent}(x', y')$ .

Unfortunately, as we can observe from figure 6.1, the linear strategy of resolution in Horn theories must be —for completeness— abandoned, since the generation of rules from critical pairs (i.e., ordered chaining inference steps) correspond to resolution among clauses of the given theory. But the advantages of the use of term ordering arise, when it is possible to saturate (i.e., to complete) a bi-rewrite system obtained from the previously explained map: The search for proofs by SLD-resolution (or straightforward deduction in rewriting logic, see section 6.2.6), which could have been non-terminating, is now ‘replaced’ by terminating bi-rewriting (because of the reduction ordering on terms).

Ordered positive chaining:

$$\frac{\text{ancestor}(x, y) \Rightarrow \text{parent}(x, y) \wedge \text{ancestor}(z, y) \quad \text{ancestor}(x', y') \Rightarrow \text{parent}(x', y')}{\text{ancestor}(x, y) \Rightarrow \text{parent}(x, z) \wedge \text{parent}(z, y)}$$

Ordered positive chaining:

$$\frac{\text{ancestor}(x, y) \Rightarrow \text{parent}(x, z) \wedge \text{parent}(z, y) \quad \text{parent}(\text{cesc}, \text{anna}) \Rightarrow \text{true}}{\text{ancestor}(\text{cesc}, y) \Rightarrow \text{true} \wedge \text{parent}(\text{anna}, y)}$$

$E'$ -equivalence:

$$\text{ancestor}(\text{cesc}, y) \Rightarrow \text{true} \wedge \text{parent}(\text{anna}, y) \equiv_{E'} \text{ancestor}(\text{cesc}, y) \Rightarrow \text{parent}(\text{anna}, y)$$

Ordered negative chaining:

$$\frac{\boxed{\text{ancestor}(\text{cesc}, \text{jordi}) \not\Rightarrow \text{true}} \quad \text{ancestor}(\text{cesc}, y) \Rightarrow \text{parent}(\text{anna}, y)}{\text{parent}(\text{anna}, \text{jordi}) \not\Rightarrow \text{true}}$$

Ordered resolution:

$$\frac{\text{parent}(\text{anna}, \text{jordi}) \not\Rightarrow \text{true} \quad \text{parent}(\text{anna}, \text{jordi}) \Rightarrow \text{true}}{\square}$$

**Figure 6.1:** Sequence of ordered chaining inferences

**6.2.9 Example.** Consider the following set  $\Gamma$  of Horn clauses:

$$\begin{aligned} q(x) &\leftarrow p(x) \\ p(x) &\leftarrow q(x) \end{aligned}$$

and the (negated) query:

$$\leftarrow q(a)$$

Though it is evident that we cannot refute it, the process of applying SLD-resolution will never terminate. Instead, given a signature precedence  $q \succ p$ , the rewrite theory to which this Horn theory is mapped, forms a convergent bi-rewrite system:

$$\begin{aligned} \Gamma'_{\Rightarrow} &= \{q(x) \longrightarrow p(x)\} \\ \Gamma'_{\Leftarrow} &= \{p(x) \longrightarrow q(x)\} \end{aligned}$$

Now we can effectively proof that  $q(a) \not\Rightarrow true$ , because  $q(a) \xrightarrow{\Rightarrow} p(a)$  is the only rewrite step that can be performed.

## 6.3 Rewriting with Set Relations

Kriaučiukas and Walicki propose a proof calculus for specifications with set relations based on Bachmair and Ganzinger's ordered chaining calculus, in order to use it as a new reasoning system for their specification framework, since term rewriting is much more prone for automation. They first presented an inference system for ground first-order clauses (Kriaučiukas and Walicki, 1995), which they later on extended to the non-ground case in (Kriaučiukas and Walicki, 1996).

Recall from section 5.5 that specifications with set relations, are essentially theories of first-order clauses, whose literals are atomic formulae of the form  $s \approx t$ ,  $s < t$ , or  $s \frown t$ , and their respective negations. In the same section, we have shown how the fragment consisting of positive unit clauses is actually a particular instance of our logic of special relations. For this reason, we are going to apply our knowledge on term rewriting along binary relation to revise some of the claims made by Kriaučiukas and Walicki concerning the notion of *relative closeness* and its relationship to the completeness of their proof calculus.

**6.3.1 The proof calculus.** First of all, we are going to express the inference rules of their calculus as a general chaining calculus for theories in our logic of special relations coming from specifications with set relations. Essentially, the calculus is the same as the one in (Kriaučiukas and Walicki, 1995), but here we express Kriaučiukas and Walicki's composition and replacement rules of set relations by means of polarities of function symbols and partial orders over relation-algebra expressions. The resulting proof calculus is shown in figure 6.2.

**6.3.2 Ordering restrictions.** In analogy to the ordered chaining calculus (see 2.4.6), Kriaučiukas and Walicki add ordering restrictions to the inference rules in order to prune the search space one has to explore for finding a refutation. Thus, only maximal literals and maximal terms may take part in an actual inference step, with the exception of a compositionality resolution step, where the 'active' literal of the second premise may not be maximal. But term  $s$  of this literal still has to be maximal in the clause.

**6.3.3 The calculus is (not) refutationally complete.** Kriaučiukas and Walicki claim that the calculus is refutationally complete, but it turns out, from how their ground-completeness theorem (Theorem 5.1 in (Kriaučiukas and Walicki, 1995)) is stated, that it is not. Their theorem is namely based on the notion of a *relatively closed* set of clauses. It states, that a model exists for every consistent and relatively closed set of clauses. Unfortunately their notion of *relative closeness* of a set, which is analogous to Bachmair and Ganzinger's

Reflexivity resolution:

$$\frac{C, \neg(s \alpha s)}{C}$$

where  $\alpha \in \{<, \prec, \neg\}$ .

Superposition:

$$\frac{C, s \alpha t \quad D, u[s]_p \beta v}{C, D, u[t]_p \gamma v}$$

where  $\gamma$  is the smallest special relation for which exists  $\alpha'$  such that  $p$  in  $u$  is positive with respect to  $(\alpha, \alpha')$  and  $\alpha'; \beta \sqsubseteq \gamma$ .

$$\frac{C, s \alpha t \quad D, \neg(u[s]_p \beta v)}{C, D, u[t]_p \gamma v}$$

where  $\gamma$  is the largest special relation for which exists  $\alpha'$  such that  $p$  in  $u$  is positive with respect to  $(\alpha, \alpha')$  and  $\alpha'; \gamma \sqsubseteq \beta$ .

Compositionality resolution:

$$\frac{C, s \alpha t \quad D, s \beta u}{C, \neg(t \gamma u), s \beta u}$$

where  $\gamma$  is the largest special relation such that  $\alpha; \gamma \sqsubseteq \beta$ .

$$\frac{C, s \alpha t \quad D, \neg(s \beta u)}{C, t \gamma u, \neg(s \beta u)}$$

where  $\gamma$  is the smallest special relation such that  $\alpha; \beta \sqsubseteq \gamma$ .

**Figure 6.2:** Kriauciukas and Walicki's proof calculus

notion of *saturation up to redundancy* discussed in 2.4.7, is too weak for such a more general framework, as the following counter-example shows:

**6.3.4 Example.** Let  $(\mathcal{S}^*, \Sigma)$  be a signature of the logic of special relations, with  $\mathcal{S}^*$  as defined in section 5.5, and with  $\Sigma = \{a, b, c, d, f(-)\}$  (four constants and one unary function symbol). Recall from section 5.5 that the argument position of  $f$  is positive with respect  $(<, <), (\neg, \neg), (\approx, <)$ , and  $(\approx, >)$ . Let  $\succ$  be an ordering on terms defined as a lexicographic path ordering based on the following precedence of function symbols  $f \succ a \succ b \succ c \succ d$ , and let  $\succ$  also denote its multiset extension to an ordering on literals and clauses. Let us now consider the set consisting of the following ground (unit) clauses, listed in increasing order:

$$\begin{array}{ccc} d & \approx & d \\ c & \neg & d \end{array}$$



$$\begin{array}{lcl}
b & \approx & d \\
b & \approx & b \\
a & < & c \\
a & \approx & b \\
f(d) & \not\approx & f(d) \\
f(c) & \not\approx & f(d) \\
f(b) & \not\approx & f(c)
\end{array}$$

All valid inferences with premises from this set are redundant, since all of them conclude with a clause already in the initial set, except for the following superposition inference:

$$\frac{a \approx b \quad a < c}{b < c}$$

But, in this case,  $b < c$  is redundant, since  $b \approx d$  and  $c \frown d$  force  $b < c$ , and both clauses are smaller in the ordering than  $b < c$ . A ground literal  $A$  is forced by a set of ground literals  $S$ , if it is in the *rewriting closure* of  $S$ , i.e., there exists a sequence of rewrite steps, with rules from  $S$ , that prove  $A$ . But the initial set has no model, because no multialgebra can satisfy  $a \approx b$ ,  $a < c$  and  $f(b) \not\approx f(c)$  at the same time.

**6.3.5 False relative closeness.** As already mentioned, the weak point lies in the notion of a *relatively closed* set of clauses. This notion itself is based on the construction of *confluent* term rewriting systems out of a consistent initial set of clauses, which should provide a rewrite proof for every atom that is valid in the model of the set. If the initial set of clauses was not consistent, than it could be only *relatively closed* if it had the empty clause. The proof of the completeness of their calculus fails because of the false claim that any atom provable by a sequence of rewrite steps, in what they define as a *confluent system*, has a rewrite proof. A counter-example follows:

**6.3.6 Example.** Let  $(S^*, \Sigma)$  be a signature of example 6.3.4. Let  $\Gamma$  consist of the following rewrite rules:

$$\begin{array}{lcl}
a & \longrightarrow & b \\
& \approx & \\
a & \longrightarrow & c \\
& < & \\
b & \longrightarrow & d \\
& \approx & \\
c & \longrightarrow & d \\
& \frown &
\end{array}$$

We have that  $\Gamma$  is a confluent system —according to Kriaučiukas and Walicki's definition of 'confluent system'— since  $b < c$  is the unique critical atom in  $\Gamma$  —it arises from the peak  $b \xleftarrow{\approx} a \xrightarrow{<} c$  and  $\approx; < \sqsubseteq <$ . But this atom has a rewrite proof in  $\Gamma$

$$b \xrightarrow{\approx} d \xleftarrow{\frown} c$$

since  $\approx; \neg \sqsubseteq <$ . Due to the monotonicity properties of function symbol  $f$  we have  $f(b) < f(c)$  is provable in  $\Gamma$  by the sequence of rewrite steps

$$f(b) \xleftarrow{<} f(a) \xrightarrow{<} f(c) ,$$

but it has no rewrite proof! Though there are two ways to reach a common term  $f(d)$  starting from  $f(a)$  and  $f(c)$  respectively,

$$\begin{array}{c} f(b) \xrightarrow{<} f(d) \xleftarrow{\neg} f(c) \\ f(b) \xrightarrow{>} f(d) \xleftarrow{\neg} f(c) \end{array}$$

in both cases  $<; \neg \sqsubseteq <$  and  $>; \neg \sqsubseteq <$ .

**6.3.7 Not well-polarized signature.** The reason of the existence of such counter-example is that the signatures coming from mapping specifications with set relations to theories in the logic of special relations are not well-polarized (see definition 4.4.3), in general. In example 6.3.6, the argument position in  $f$  is positive with respect to  $(\approx; \neg, <)$ , because  $\approx; \neg \sqsubseteq <$  and the argument position in  $f$  is positive with respect to  $(<, <)$ , but unfortunately, there are no relations  $\alpha, \beta \in S \cup \check{S}$ , such that the argument position in  $f$  is positive with respect to  $(\approx, \alpha)$  and with respect to  $(\neg, \beta)$  and  $\alpha; \beta \sqsubseteq <$ .

**6.3.8 Subtleties of the generalization.** Of course, affirming that Kriaučiukas and Walicki's proof calculus is *incomplete* is not appropriate; Kriaučiukas himself agreed that “[our] arguments hit [their] method of proof, but not the proof calculus, because redundancy is not used in the definition of the strategy” (Kriaučiukas, 1999). With this revision of Kriaučiukas and Walicki's work on an extension of the bi-rewriting and ordered chaining techniques to other general binary relation we wanted to highlight the negative consequences of a too shallow and straightforward generalization from equality to inclusions to arbitrary binary relations, relying on the false hypothesis that such apparent direct generalization would translate on a direct application of the notion of completeness up to redundancy from the equational and the bi-rewriting technique to a more general framework. On the contrary, such generalization carries many subtleties to take into account, as we have shown in chapter 4, and the signature's special relations and function symbols need to verify many properties, such as well-polarization, for the computational power of term rewriting to be applied as a proof calculus in such more general specification paradigms.

## 6.4 Conclusion

By applying the general notion of term rewriting, we were able to analyze several quite different computational aspects of three distinct specification frameworks. Within the area of order-sorted term rewriting, we could neatly relate the notions of confluence, sort-decreasingness, descendingness, and regularity, and describe them with one unique general notion of local confluence. Furthermore,

we showed how theories of special relations with different polarizations of function symbols arise naturally when looking at Horn theories as rewrite theories, and further as theories in our logic of special relations. Consequently, we deal with a rewrite relation that does not rewrite within the structure of propositions, and we propose convergent bi-rewrite systems as possible alternative operational semantics for Horn logic programming. Finally, we also revised the generalization of term rewriting carried out for non-deterministic specifications with set relations, pointing out to the false notion of redundancy that invalidates the completeness theorem for a proof calculus based on these techniques.

In the next part, we are going to make a foray into the terrain of diagrammatic reasoning, and explore the role of special relations in reasoning with specifications based on visual clues. We show that our framework is general enough to cope also with formalisms based on intuitive graphical representations, which attempts to make formal specification approachable by non-logicians.



## Part IV

# Reasoning in a Diagrammatic Logic



## Chapter 7

# Special Relations in a Diagrammatic Logic

The systematic development of correct programs from complete formal specifications by means of verified refinement steps has attracted considerable effort. Much less attention has been devoted to formal *requirements engineering*, the difficult process of creation of adequate formal and complete specifications from informal requirements (Wieringa, 1996; Robertson and Agustí, 1999). Specifications cannot be validated conclusively with respect to the real world, they can only be judged subjectively as adequate descriptions of a problem, maybe with the help of some theorem prover which computes consequences of the formulation. In this direction Levy and Agustí defined the *calculus of refinements* (Levy, 1995), a functional specification language designed to shorten the distance between the informal description of a problem and its first formalization, by concentrating on an incremental approach to executable formal specification (Robertson et al., 1994).

Formal specification languages, to be widely used in requirements engineering, should not present undue difficulties of use and interpretation to the persons who create and read the specifications, who usually are experts on the application domain and not programmers. Agustí, Robertson, and Puigsegur were interested in intuitive forms of problem description and resolution within a complete formal language, and thus started to express preliminary specifications in a notation accessible to non-specialists, while providing clear points of refinement towards complete specifications in languages targeted to more specialized developers. Consequently, they first defined the graphical specification language GRASP (Agustí et al., 1995), that evolved towards a visual logic programming language (Puigsegur et al., 1996; Agustí et al., 1998a). This research is the subject of a forthcoming doctoral dissertation (Puigsegur, 2000).

This chapter is the result of the joint work with Puigsegur and Agustí, motivated by our belief that the understanding and pragmatics of formal logic and automated deduction are sensitive to the syntax used to express statements.

For instance, it has been shown that the taxonomic syntax of many knowledge representation languages facilitates the inference (McAllester et al., 1989). We therefore explore the somehow informal claim that a diagrammatic syntax not only can be an alternative formal notation, emphasizing some structural features of logical statements, but it could be useful to conduct visual inferences and communicate them.

The key visual concepts exploited by Puigsegur and Agustí in their diagrammatic logic are that of a box representing a set, and that of a box contained within another box, representing set inclusion. This *spatial* relationship is the basic building block for defining predicates and writing visual programs. In a first approach, Agustí, Puigsegur, and Robertson (1995) translated their diagrammatic sentences into standard Horn clauses in order to execute their visual programs as pure Prolog programs. But in order to have a real diagrammatic programming language it was necessary to establish a completely visual operational semantics, and to explore its advantages and disadvantages; this is our contribution to their work.

In previous chapters of this thesis, we have drawn our attention to the importance of special relations in specification frameworks, and we studied their role in rewriting-based proof calculi. Since the *spatial* relationship of box containment acts as *special* relation in the diagrammatic logic, we believed it necessary to formally establish the relationship between Puigsegur and Agustí's diagrammatic logic and our logic of special relations, introduced in chapter 3, by showing that diagrammatic programs are actually specific theories in the logic of special relations. In order to establish this correspondence we proceed, as done in chapter 5 with other frameworks, by axiomatizing the diagrammatic logic as a general logic, and defining a map from this logic into the logic of special relations. As in section 5.5, we will be only concerned with the map of institutions, and we will only sketch the proof. The existence of such map guarantees the correctness of a chaining based proof inference mechanism as suitable operational semantics for the language, and we achieve to define such calculus in a completely diagrammatic way.

## 7.1 A Diagrammatic Horn Logic

Puigsegur and Agustí were after a formal syntax for logic programming which clearly 'resembles' the corresponding semantics. The standard form of symbolic logic was set by mathematicians in linear form, patterned after Boolean algebra. This syntax favors the interpretation of predicates as truth functions and connectives as truth operators. Nothing in it suggests the more usual interpretation by logic practitioners of predicates as relations, that is, sets of tuples of individuals from the universe of discourse. In this section, we present Puigsegur and Agustí's main intuitions by means of an example, transforming a predicate definition in standard Horn logic syntax into a diagram. We want to emphasize their belief, which is also ours, that such a diagram better conveys its meaning. Our particular contribution relies in showing how this diagrammatic syntax is



also fit to represent automated deduction in it.

**7.1.1 Clause transformation.** For example, to define who the ancestors of a person are, we could write the following Horn clauses,

$$\begin{aligned} \text{ancestor}(x, y) &\longleftarrow \text{person}(x) \wedge \text{parent}(x, y) \\ \text{ancestor}(x, y) &\longleftarrow \text{person}(x) \wedge \text{parent}(x, z) \wedge \text{ancestor}(z, y) \end{aligned}$$

in which, as we said previously, the interpretation closest to the syntax is by considering *person*, *ancestor*, and *parent* as truth valued functions. However, if we want to suggest their interpretations as a set of tuples of individuals, then a better notation could be set notation using membership ( $\in$ ) as unique predicate, and write the previous expressions as follows:

$$\begin{aligned} \langle x, y \rangle \in \text{ancestor} &\longleftarrow x \in \text{person} \wedge \langle x, y \rangle \in \text{parent} \\ \langle x, y \rangle \in \text{ancestor} &\longleftarrow x \in \text{person} \wedge \langle x, z \rangle \in \text{parent} \wedge \langle z, y \rangle \in \text{ancestor} \end{aligned}$$

The next step in order to make the syntax better convey the intended operational meaning of logical expressions could be to somehow make explicit the wanted directionality of some arguments as input or output. The importance of directionality was stressed for instance in (Dewille, 1990). In our example, both, *ancestor* and *parent*, are better understood as non-deterministic functions, which give for each individual the set of its ancestors and parents respectively. Then we can transform the previous expressions into the following ones,

$$\begin{aligned} y \in \text{ancestor}(x) &\longleftarrow x \in \text{person} \wedge y \in \text{parent}(x) \\ y \in \text{ancestor}(x) &\longleftarrow x \in \text{person} \wedge z \in \text{parent}(x) \wedge y \in \text{ancestor}(z) \end{aligned}$$

where we use the conventional linear syntax for functional application. This new functional notation allows us to show syntactically two other structural features of the definition of predicates: First, predicate composition by successive applications, and second, logical implication by set inclusion. In the standard syntax, predicate composition is expressed by means of the ‘and’ connective ( $\wedge$ ), see for instance the composition of *parent* and *ancestor* in the second of the previous clauses. However, predicate composition could be noted more directly by successive function applications. Then we can write the previous expressions as follows<sup>1</sup>

$$\begin{aligned} y \in \text{ancestor}(x) &\longleftarrow x \in \text{person} \wedge y \in \text{parent}(x) \\ y \in \text{ancestor}(x) &\longleftarrow x \in \text{person} \wedge y \in \text{ancestor}(\text{parent}(x)) \end{aligned}$$

Now we are ready to represent logical implication by means of set inclusion. In most Horn clauses we can distinguish a main implication, which connects the

---

<sup>1</sup>Notice that we use *ancestor* as a function over subsets. The purpose of this example is to emphasize the advantages of functional composition, thus we shall not worry about typing issues here.

head of the clause with conditions in the body sharing the same output variable. For instance, the previous clauses can be written as follows,

$$\begin{aligned} (y \in \text{ancestor}(x) \leftarrow y \in \text{parent}(x)) \leftarrow x \in \text{person} \\ (y \in \text{ancestor}(x) \leftarrow y \in \text{ancestor}(\text{parent}(x))) \leftarrow x \in \text{person} \end{aligned}$$

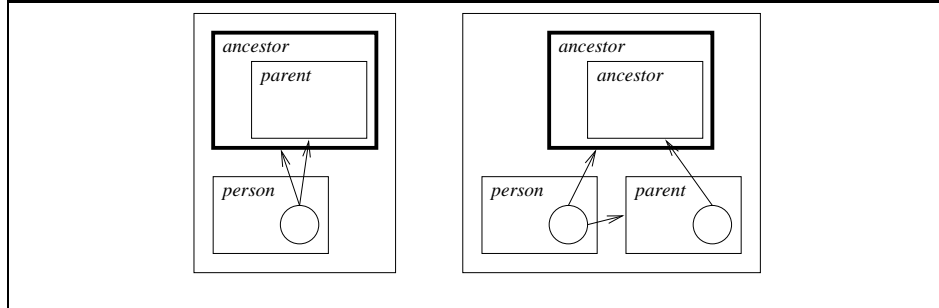
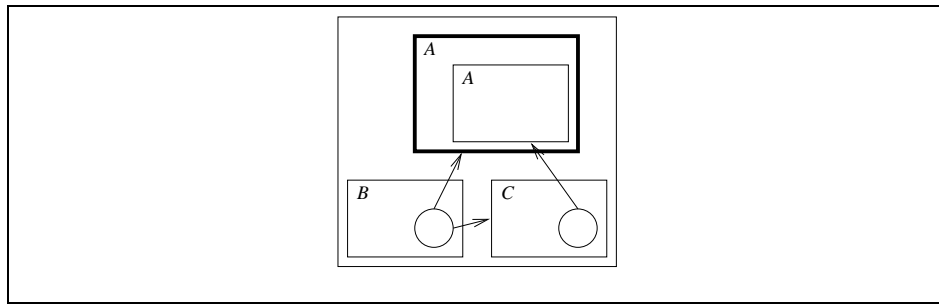
making explicit the main implication. Then it can be more directly represented by set inclusion ( $\subseteq$ ), leaving the other implications as conditions. For instance, in our example, instead of the implications, we represent the inclusion of *parent* into *ancestor* and of *ancestor\_of\_parent* into *ancestor* as follows:

$$\begin{aligned} \text{parent}(x) \subseteq \text{ancestor}(x) \leftarrow x \in \text{person} \\ \text{ancestor}(\text{parent}(x)) \subseteq \text{ancestor}(x) \leftarrow x \in \text{person} \end{aligned}$$

**7.1.2 Inclusional Horn clauses.** Up to now, we have transformed the initial Horn clauses into a new type of Horn clauses with two predicates only, inclusion ( $\subseteq$ ) and membership ( $\in$ ) between terms composed with predicates of the initial syntax. We call them *inclusional Horn clauses*. Notice that the transformations, although explained by means of an example, are completely general, and could be applied to any standard Horn clause definition of a predicate. However, the real interest in using this set notation for Horn clauses comes from its natural representation by means of diagrams. To do so, Puigsegur and Agustí exploit two well known visual formalisms, namely Venn/Euler diagrams to emphasize set inclusion and membership, and directed acyclic graphs to describe the structure of terms.

**7.1.3 Diagrams.** The correspondence between an inclusional Horn clause and the corresponding diagram is as follows. Variables are represented as circles, and each  $n$ -ary predicate (seen as an  $(n-1)$ -ary non-deterministic function whose result is the  $n$ -th argument of the original predicate) is represented by a square box labeled with the name of the predicate, and is pointed by  $n-1$  arrows coming from its arguments. The argument order of the predicate may be disambiguated by labeling the arrows, if necessary. The square box denotes the set of results corresponding to the  $n$ -th argument of the  $n$ -ary predicate not represented with an incoming arrow. The predicate being defined is distinguished by drawing its box highlighted with thick lines, and it contains the box corresponding to the left-hand side of the inclusion in the inclusional Horn clause. The memberships in the body of the clause are represented by circles contained in boxes denoting the corresponding predicates. Then each clause is represented by a set of graphical containments circumscribed by an unlabeled box to note the scope of the diagram. For instance, each one of the previous inclusional Horn clauses can be represented by a different diagram as shown in figure 7.1.

Puigsegur and Agustí claim that the ‘degree of homomorphism’ or ‘resemblance’, as understood by Barwise and Hammer (1996), between a diagram and the predicate it describes, is higher than with linear formulas. Another structural

**Figure 7.1:** Diagrammatic Horn clauses**Figure 7.2:** Pattern diagram of recursion

feature of predicate definition highlighted by diagrams is recursion. The structure of recursion is represented by the visual metaphor of a picture reproduced inside itself, as can be seen in the right diagram of figure 7.1. Furthermore, ‘pattern diagrams’ labeled with variable symbols could be useful to record different patterns of recursion like the one in figure 7.2.

It is a simple recursive pattern common to many definitions, as, for instance, the one of *ancestor* given in figure 7.1. A library of diagrammatic patterns recording different styles of description could become a pragmatic tool to support the use of logic. However, the utility of this diagrammatic language to a particular community of users can only be claimed after serious empirical testing, which still has to be done.

In the previous diagrams we only represented predicate and variable symbols. In general we also need to represent terms built from function and constant symbols. These terms denoting individuals are represented by directed acyclic graphs (DAGs). Their nodes are either round boxes, labeled by the name of a function (or of a constant), or circles. As usual, arrows in a DAG represent function application. The advantage of representing terms as DAGs, compared to linear textual terms, is well known.

Figure 7.3 shows a more complex Horn clause in standard syntax, inclusional syntax, and the corresponding diagram. Notice that there is no need to name variables—we denote them simply by circles. The graph structure allows us to represent variable sharing or coreference.

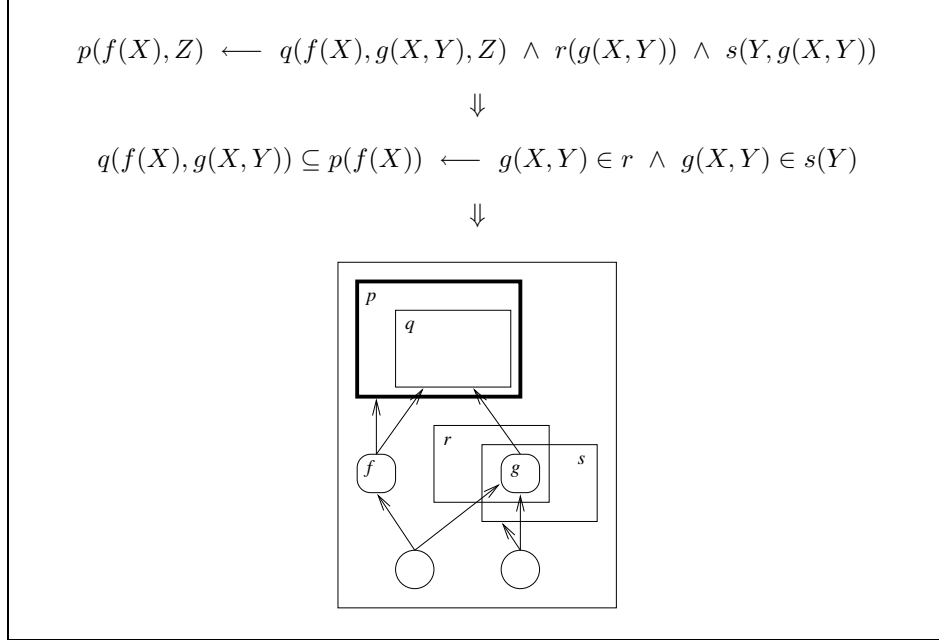


Figure 7.3: From Horn clause to diagram

## 7.2 Spatial and Special Relations

Since the diagrammatic language proposed in (Agustí et al., 1998a; Puigsegur and Agustí, 1998) is actually Horn logic with set inclusion and set membership, it can be seen as an instance of the logic of special relations introduced in chapter 3. Consequently, a chaining based proof calculus is suitable as operational semantics for the language, and we outline such calculus in a completely diagrammatic way. The diagrammatic nature provides us with some interesting features, namely the ability of keeping track, within a diagram, of the proof that is built while trying to refute a clause, and the possibility to represent alternative proofs within a unique diagram.

Before identifying the institution of the diagrammatic logic and its map to the institution of the logic of special relations, let us first review some notation concerning relational expressions that we will use in subsequent paragraphs.

**7.2.1 Relational expressions.** A relation  $R$  between elements of a set  $A$  and elements of a set  $B$  is a set of pairs  $R \subseteq A \times B$ . The *domain* and *codomain* of  $R$  is defined as follows:

$$\begin{aligned}
 \text{dom}(R) &= \{a \in A \mid \exists b \in B \ (a, b) \in R\} \\
 \text{cod}(R) &= \{b \in B \mid \exists a \in A \ (a, b) \in R\}
 \end{aligned}$$





Let  $C$  be a set, and let  $S \subseteq B \times C$  be a relation. Relation composition, denoted with the binary operator  $;$ , and relation reversion, denoted with the unary

operator  $\smile$ , is defined as follows:

- $(a, c) \in R; S$  if and only if there exists  $b \in B$  such that  $(a, b) \in R$  and  $(b, c) \in S$
- $(a, b) \in \check{R}$  if and only if  $(b, a) \in R$

**7.2.2 Syntax.** Signatures in the diagrammatic Horn logic are standard first-order signatures  $\Omega = (\Sigma, \Pi)$ , where  $\Sigma$  is a ranked alphabet of function symbols and  $\Pi$  is a ranked alphabet of predicate symbols. The distinguishing syntactic aspect of the diagrammatic Horn logic with respect to standard Horn logic is how sentences are formed, since they are diagrams as described in 7.1.3. Diagrams are essentially graphs with three sorts of nodes together with a relation of containment between nodes of the graph. Each diagram has a distinguished square box, highlighted with thick lines, and denotes the predicate we are defining (in figure 7.3 it is predicate  $p$ ), and which contains one or more graphical items, either a round box, a circle, or another square box, free of other containments. All other square boxes (drawn with thin lines) contain exactly one graphical item, and these can only be round boxes or circles, never other square boxes. Nevertheless, overlapping of different square boxes is allowed (e.g., in figure 7.3, boxes  $r$  and  $s$ ).

**7.2.3 Definition.** A *sentence* or *diagram* is a pair  $D = (G, L)$ , consisting of a customized higraph<sup>2</sup>, together with labeling functions, given as follows:

- $G = (N, E, C_1, C_2)$  is a directed acyclic graph with
  - nodes consisting of a finite set  $S$  of square boxes , a finite set  $R$  of round boxes , a finite set  $X$  of circles , and one highlighted square box , i.e.,  $N = X \cup R \cup S \cup \{\text{thick square box}\}$ ,
  - edges going from circles and round boxes to round boxes and square boxes (including the highlighted box), i.e.,  $E \subseteq (R \cup X) \times (R \cup S \cup \{\text{thick square box}\})$ ;  $E$  is a multiset, because we may have more than one edge between two particular nodes of the graph<sup>3</sup>, and
  - a couple of irreflexive relations of containment between nodes, such that  $C_1 \subseteq \{\text{thick square box}\} \times N$  and  $C_2 \subseteq S \times (R \cup X)$ ;
- $L = (l_S, l_R)$  is a pair of total functions labeling every square box (including the highlighted box) with a predicate symbol and every rounded box with a function symbol, i.e.,  $l_S : (S \cup \{\text{thick square box}\}) \rightarrow \Pi$  and  $l_R : R \rightarrow \Sigma$ .

Given the multiset of edges  $E$ , let  $\text{conn}_E : N \rightarrow \mathcal{P}(N)$  be the function that, given a node  $w \in N$ , yields the multiset of nodes with an edge pointing to it, i.e.,  $\text{conn}_E(w) = \text{cod}(\{(w, w)\}; \check{E})$ . A sentence (diagram) is *well-formed* if and only if

<sup>2</sup>Higraphs were introduced by Harel (1988).

<sup>3</sup>Recall that a multiset is a set where the number of occurrences of an element is significant.

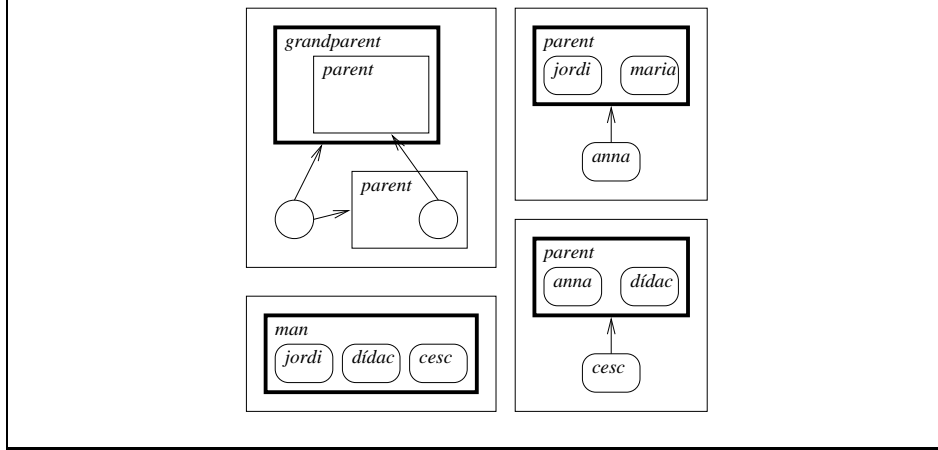


Figure 7.4: A diagrammatic logic program

- The highlighted box contains only boxes free from any containment, i.e.,  $C_1; C_2 = \emptyset$ ,
- square boxes contain exactly one node, either round box or circle, i.e.,  $C_2$  is actually a total *function* on the domain  $S$ ,
- each node is pointed by the correct number of arrows according to the arity of the function or predicate symbol that labels it, i.e., for every  $w \in R$ ,  $|conn_E(w)| = \text{arity}(l_R(w))$ , and for every  $w \in S$ ,  $|conn_E(w)| = \text{arity}(l_S(w)) - 1$ , where function  $|\cdot|$  yields the cardinality of a set, and the function *arity* the arity of a function symbol.
- all circles and round boxes that are also leaves of the graph are always contained in some other square box, i.e.,  $(R \cup X) \setminus \text{dom}(E) \subseteq \text{cod}(C_1) \cup \text{cod}(C_2)$ , where the function  $\setminus$  yields the set difference between two sets.

A collection of the kind of diagrams of figures 7.1 and 7.3 form a diagrammatic logic program, such as the one given in figure 7.4. Thus, a *diagrammatic logic program* is a theory  $T = (\Omega, \Gamma)$ , where  $\Omega$  is a signature and  $\Gamma$  is a set of diagrams.

**7.2.4 Semantics.** Let  $\Omega$  be a signature in the diagrammatic Horn logic.  $\Omega$ -models are just standard first-order models, i.e., they consist of a universe  $\mathcal{A}$  together with an assignment to each symbol  $f \in \Sigma_n$  of a function  $f_{\mathcal{A}} : \mathcal{A}^n \rightarrow \mathcal{A}$  and to each predicate symbol  $p \in \Pi_n$  of a set  $p_{\mathcal{A}} \subseteq \mathcal{A}^n$ .

A diagram defines a predicate by stating the subsets or elements of the graphical set associated to the predicate. Such predicate has arguments as arrows pointing to the square box it represents. One argument is kept implicit. When defining an interpretation function of this box, we have to commit to some particular order of these arguments, and thus we introduce a  *tupling function*.

**7.2.5 Definition.** Given a set  $N$ , we define a *tupling function*

$$\tau : \mathcal{P}(N) \rightarrow \bigcup_{n \in \mathbb{N}} N^n$$

as follows: Let  $S \subseteq N$  with cardinality  $n \in \mathbb{N}$ .

$$\tau(S) = (s_1, \dots, s_n) \in N^n$$

such that, for all  $s \in S$ , there exists an  $s_i = s$ .

For each diagram  $D$ , we define an interpretation function  $\llbracket \cdot \rrbracket$  as follows:

**7.2.6 Definition.** Let  $D$  be a well-formed diagram, as defined in 7.2.3, let  $\tau$  be a tupling function, and let  $\rho : X \rightarrow \mathcal{A}$  be a valuation function of circles (which is analogous to a valuation function of variables). For every node  $w \in N$  of such diagram,

$$\llbracket w \rrbracket = \begin{cases} \rho(w) & \text{if } w \in X \\ l_R(w)_{\mathcal{A}}(\llbracket \tau(\text{conn}_E(w)) \rrbracket) & \text{if } w \in R \\ \{a \in \mathcal{A} \mid (\llbracket \tau(\text{conn}_E(w)) \rrbracket, a) \in l_S(w)_{\mathcal{A}}\} & \text{if } w \in S \cup \{\boxed{\phantom{x}}\} \end{cases}$$

where the interpretation function is applied over  $n$ -tuples in the following way:

$$\llbracket (s_1, \dots, s_n) \rrbracket = (\llbracket s_1 \rrbracket, \dots, \llbracket s_n \rrbracket)$$

**7.2.7 Satisfaction.** Diagrams can be viewed as conditional subset or membership assertions, where the containments in the highlighted box are conditioned by the rest of containments in the diagram. In order to formally state when an interpretation satisfies a sentence, we need to consider two containment relations. That is why we are distinguishing between  $C_1$  and  $C_2$ . We further partition the containment relation  $C_1$  distinguishing between containment of square boxes into the highlighted box, and between containment of round boxes and circles into the highlighted box. Thus,  $C_{11}$  consists of those pairs of  $C_1$  having only elements of  $S$  in its codomain, and  $C_{12}$  consists of those pairs of  $C_1$  having only elements of  $R \cup X$  in its codomain. An  $\Omega$ -model  $\mathcal{A}$  *satisfies* a sentence (diagram)  $D$ ,  $\mathcal{A} \models^{\text{DHL}} D$  if and only if, for every assignment  $\rho : X \rightarrow \mathcal{A}$ , we have

$$\begin{aligned} (\forall (v, u) \in C_2 \quad \llbracket u \rrbracket \in \llbracket v \rrbracket) \\ \Rightarrow (\forall (t', t) \in C_{11} \quad \llbracket t \rrbracket \subseteq \llbracket t' \rrbracket) \quad \wedge \quad (\forall (t', t) \in C_{12} \quad \llbracket t \rrbracket \in \llbracket t' \rrbracket) \end{aligned}$$

**7.2.8 Remark.** Though the visual notation is close to Venn diagrams, two important differences are worth noticing: First, the graphical sets of diagrams do not have a unique physical existence, i.e., the same set may appear in various places of the diagram. Second, in a diagram only graphical containment information is relevant, the absence of graphical containment is not taken into account, which means that graphically non-overlapping sets may still have elements in common.

**7.2.9 Insitution.** We deal with first-order signatures and models, thus they form the categories  $\mathbf{Sign}^{\text{DHL}}$  and  $\mathbf{Mod}^{\text{DHL}}(\Omega)$  respectively. Let  $sen^{\text{DHL}} : \mathbf{Sign}^{\text{DHL}} \rightarrow \mathbf{Set}$  be the functor assigning to each signature the set of its well-formed diagrams, and let  $\models^{\text{DHL}}$  be the logical consequence relation given above. Signatures, sentences, models, and logical consequence in the diagrammatic Horn logic form together the institution  $(\mathbf{Sign}^{\text{DHL}}, sen^{\text{DHL}}, \mathbf{Mod}^{\text{DHL}}, \models^{\text{DHL}})$ .

**7.2.10 A particular logic of special relations.** Let  $D$  be a well-formed diagram as defined in definition 7.2.3. Given a node  $n \in N$  we will denote with  $\hat{n}$  the first-order term constructed with the labels attached to the nodes of the graph rooted<sup>4</sup> at this node. Thus,  $\hat{n} = f(t_1, \dots, t_n)$ , where  $f \in \Sigma_n \cup \Pi_{n+1}$ <sup>5</sup>, and  $t_i \in \mathcal{T}_\Sigma(\mathcal{X})$ , where  $\mathcal{X}$  is a set of variable symbols in bijection to the set  $X$  of circles in  $D$ .

Of course we must determine the order of the parameters in the first-order terms in some way. We will do that by previously fixing a tupling function  $\tau$  as given in definition 7.2.5. As before, let us partition  $C_1$  into  $C_{11}$  and  $C_{12}$ , and let  $C_2 = \{(v_1, u_1), \dots, (v_n, u_n)\}$ . We map the diagrams  $D$  to a set of conditional sentences as follows:

$$\begin{aligned} \alpha(D) = \{ \hat{t} \subseteq \hat{t}' \text{ if } \hat{u}_1 \in \hat{v}_1 \wedge \dots \wedge \hat{u}_n \in \hat{v}_n \mid (t', t) \in C_{11} \} \\ \cup \{ \hat{t} \in \hat{t}' \text{ if } \hat{u}_1 \in \hat{v}_1 \wedge \dots \wedge \hat{u}_n \in \hat{v}_n \mid (t', t) \in C_{12} \} \end{aligned}$$

Observe, as already mentioned, that these sentences in the logic of special relations involve a couple of special relations  $\subseteq$  and  $\in$ . We will have to specify their properties explicitly, by means of the partial order of the relation-algebra structure, and also by means of the monotonicity properties of the function symbols. This will become clear as we look at how theories are mapped.

We map a diagrammatic logic program  $((\Sigma, \Pi), \Gamma)$  to a theory  $(\Omega', \Gamma')$  in the logic of special relations, where

- $\Omega' = (M, (\mathcal{S}^*, \Sigma'))$ ,
  - $M = \{s_1, s_2\}$  is a set of two sort symbols,
  - $\mathcal{S}^* = (S^*, ;, I', \smile, \sqsubseteq)$  being the smallest partially ordered free monoid with anti-involution generated over the set  $S = \{\subseteq, \in\}$  satisfying:

$$\begin{array}{ccccc} I' & \sqsubseteq & \subseteq \\ \subseteq; \subseteq & \sqsubseteq & \subseteq \\ \in; \subseteq & \sqsubseteq & \in \end{array}$$

<sup>4</sup>Perhaps the expression ‘rooted’ is not the most suitable one, because we build the term following the edges in their reverse direction; so for instance in figure 7.3, letting  $n$  be the node labeled by  $q$ , we have that  $\hat{n} = q(f(x), g(x, y))$ .

<sup>5</sup>Recall that boxes labelled with predicate symbols have one incoming arrow less than the arity of the predicate symbol.



- $\Sigma' = \Sigma \cup \Pi$  being the ranked alphabet of function and predicate symbols (reducing the arity of predicate symbols by 1), such that for all  $f$  in  $\Sigma'$ , no argument positions  $i$  in  $f$  is monotonic or antimonotonic with respect to any pair of special relations.

- $\Gamma' = \alpha(\Gamma) = \bigcup_{D \in \Gamma} \alpha(D)$

We write  $\Phi(\Omega, \Gamma) = (\Omega', \Gamma')$ , where  $\Phi : \mathbf{Th}^{\text{DHL}} \longrightarrow \mathbf{Th}^{\text{LSR}}$  is an  $\alpha$ -sensible functor.

A model  $\mathbf{A}$  of its corresponding theory  $\Phi(T)$  in the logic of special relations is a quasi-allegory whose subcategory of functions  $\text{Fun}(\mathbf{A})$  is Cartesian closed, with

- two objects  $\llbracket s_1 \rrbracket$  and  $\llbracket s_2 \rrbracket$  corresponding to the two sorts of  $M$ ,
- a function  $\llbracket f \rrbracket : \llbracket s_1 \rrbracket^n \longrightarrow \llbracket s_1 \rrbracket$  for each  $f \in \Sigma_n$ ,
- a function  $\llbracket p \rrbracket : \llbracket s_1 \rrbracket^{n-1} \longrightarrow \llbracket s_2 \rrbracket$  for each  $p \in \Pi_n$ ,
- arrows  $\llbracket \in \rrbracket : \llbracket s_1 \rrbracket \longrightarrow \llbracket s_2 \rrbracket$  and  $\llbracket \subseteq \rrbracket : \llbracket s_2 \rrbracket \longrightarrow \llbracket s_2 \rrbracket$ ,

such that

$$\begin{aligned} id_{\llbracket s_2 \rrbracket} &\leq \llbracket \subseteq \rrbracket \\ \llbracket \subseteq \rrbracket \cdot \llbracket \subseteq \rrbracket &\leq \llbracket \subseteq \rrbracket \\ \llbracket \subseteq \rrbracket \cdot \llbracket \in \rrbracket &\leq \llbracket \in \rrbracket \end{aligned} \tag{7.1}$$

and, for each sentence

$$\hat{t}(\bar{x}) \gamma \hat{t}'(\bar{x}) \text{ if } \hat{u}_1(\bar{x}) \in \hat{v}_1(\bar{x}) \wedge \cdots \wedge \hat{u}_n(\bar{x}) \in \hat{v}_n(\bar{x})$$

in  $\alpha(\Gamma)$ , we have that

$$\llbracket \hat{t}'(\bar{x}) \rrbracket \cdot \iota \leq \llbracket \gamma \rrbracket \cdot \llbracket \hat{t}(\bar{x}) \rrbracket \cdot \iota,$$

where  $\iota$  is the includer of the family of arrows  $(\llbracket \hat{v}_i(\bar{x}) \rrbracket, \llbracket \in \rrbracket \cdot \llbracket \hat{u}_i(\bar{x}) \rrbracket)_{i \in [1..n]}$  (see definition 3.4.2), and  $\gamma$  is either  $\subseteq$  or  $\in$ .

We define the map of institutions as follows: We take as universe  $\mathcal{A} = \text{Hom}(\mathbf{1}, \llbracket s_1 \rrbracket)$  in  $\text{Fun}(\mathbf{A})$ . For each  $f$  in  $\Sigma_n$  we define  $f_{\mathcal{A}} : \mathcal{A}^n \rightarrow \mathcal{A}$  as follows

$$f_{\mathcal{A}}(a_1, \dots, a_n) \stackrel{\text{def}}{=} \llbracket f \rrbracket \cdot \langle a_1, \dots, a_n \rangle \tag{7.2}$$

For each  $p$  in  $\Pi_n$  we define  $p_{\mathcal{A}} \subseteq \mathcal{A}^n$  as follows

$$p_{\mathcal{A}} \stackrel{\text{def}}{=} \{(a_1, \dots, a_n) \in \mathcal{A} \mid \llbracket p \rrbracket \cdot \langle a_1, \dots, a_{n-1} \rangle \leq \llbracket \in \rrbracket \cdot a_n\} \tag{7.3}$$

These interpretations can be extended as usual over terms as defined in 7.2.10. We map the model of  $\Phi(T)$ , i.e., quasi-allegory  $\mathbf{A}$ , to the first-order model  $\mathcal{A}$  constructed above, and write  $\beta(\mathbf{A}) = \mathcal{A}$ , where  $\beta : \mathbf{Mod}^{\text{LSR}} \cdot \Phi^{op} \longrightarrow \mathbf{Mod}^{\text{DHL}}$  is a natural transformation.

**7.2.11 Proposition.** *The map  $(\Phi, \alpha, \beta) : (\mathbf{Sign}^{\text{DHL}}, \text{sen}^{\text{DHL}}, \mathbf{Mod}^{\text{DHL}}, \models^{\text{DHL}}) \rightarrow (\mathbf{Sign}^{\text{LSR}}, \text{sen}^{\text{LSR}}, \mathbf{Mod}^{\text{LSR}}, \models^{\text{LSR}})$  is a map of institutions.*

PROOF: Let  $D = (G, L)$  be a diagram as defined in 7.2.3. Recall that a diagram translates along  $\alpha$  to a set of sentences, one for each containment in the highlighted box (i.e., one for each pair in  $C_1$ ). Therefore we are going to handle each of this translations individually by considering that  $C_1$  only has one unique pair  $(t', t)$  capturing the containment. Let  $C_2 = \{(v_1, u_1), \dots, (v_n, u_n)\}$ . We have two cases, when  $t \in R \cup X$  or  $t \in S$ .

1. If  $t \in R \cup X$ , i.e., the highlighted box contains a round box or a circle, then

$$\alpha(D) = \hat{t}(\bar{x}^m) \in \hat{t}'(\bar{x}^m) \text{ if } \hat{u}_1(\bar{x}^m) \in \hat{v}_1(\bar{x}^m) \wedge \dots \wedge \hat{u}_n(\bar{x}^m) \in \hat{v}_n(\bar{x}^m)$$

We reason:  $\mathbf{A} \models^{\text{LSR}} \alpha(D)$  if and only if

$$\llbracket \hat{t}'(\bar{x}^m) \rrbracket \cdot \iota \leq \llbracket \in \rrbracket \cdot \llbracket \hat{t}(\bar{x}^m) \rrbracket \cdot \iota ,$$

where  $\iota$  is the includer of the family of arrows

$$(\llbracket \hat{v}_i(\bar{x}^m) \rrbracket, \llbracket \in \rrbracket \cdot \llbracket \hat{u}_i(\bar{x}^m) \rrbracket)_{i \in [1 \dots n]} .$$

From definition 3.4.2 of includer, the latter holds, if and only if for all  $\zeta \in \mathcal{A}^m$ , and for all  $i \in [1 \dots n]$ , such that

$$\llbracket \hat{v}_i(\bar{x}^m) \rrbracket \cdot \zeta \leq \llbracket \in \rrbracket \cdot \llbracket \hat{u}_i(\bar{x}^m) \rrbracket \cdot \zeta ,$$

we have also that

$$\llbracket \hat{t}'(\bar{x}^m) \rrbracket \cdot \zeta \leq \llbracket \in \rrbracket \cdot \llbracket \hat{t}(\bar{x}^m) \rrbracket \cdot \zeta .$$

Let  $\zeta = \langle a_1, \dots, a_m \rangle$ . From (7.2) and (7.3) follows that

$$\llbracket \hat{v}_i(\bar{x}) \rrbracket \cdot \zeta \leq \llbracket \in \rrbracket \cdot \llbracket \hat{u}_i(\bar{x}) \rrbracket \cdot \zeta$$

if and only if

$$(a_1, \dots, a_{m-1}, \hat{v}_{i\mathcal{A}}(a_1, \dots, a_{m-1})) \in \hat{u}_{i\mathcal{A}} ;$$

but then from (7.2) and (7.3) follows that

$$(a_1, \dots, a_{m-1}, \hat{t}_{\mathcal{A}}(a_1, \dots, a_{m-1})) \in \hat{t}'_{\mathcal{A}} .$$

Therefore,  $\mathbf{A} \models^{\text{LSR}} \alpha(D)$  if and only if  $D$  is satisfied by  $\mathcal{A} = \beta(\mathbf{A})$ , for any assignment  $\rho : X \rightarrow \mathcal{A}$ , i.e.,  $\mathcal{A} \models^{\text{DHL}} D$ .

2. If  $t \in S$ , i.e., the highlighted box contains a square box, then

$$\alpha(D) = \hat{t}(\bar{x}^m) \subseteq \hat{t}'(\bar{x}^m) \text{ if } \hat{u}_1(\bar{x}^m) \in \hat{v}_1(\bar{x}^m) \wedge \dots \wedge \hat{u}_n(\bar{x}^m) \in \hat{v}_n(\bar{x}^m)$$

We reason in a similar way as before, with the only difference that now  $\mathbf{A} \models^{\text{LSR}} \alpha(D)$  if and only if

$$\llbracket \hat{t}'(\bar{x}^m) \rrbracket \cdot \iota \leq \llbracket \subseteq \rrbracket \cdot \llbracket \hat{t}(\bar{x}^m) \rrbracket \cdot \iota , \quad (7.4)$$

where  $\iota$  is the includer as before. Now

$$\hat{t}_{\mathcal{A}} \subseteq \hat{t}'_{\mathcal{A}}$$

if and only if  $(a_1, \dots, a_m) \in \hat{t}_{\mathcal{A}}$  implies  $(a_1, \dots, a_m) \in \hat{t}'_{\mathcal{A}}$ . But

$$(a_1, \dots, a_m) \in \hat{t}_{\mathcal{A}}$$

if and only if

$$\llbracket \hat{t}(\bar{x}^m) \rrbracket \cdot \langle a_1, \dots, a_{m-1} \rangle \leq \llbracket \in \rrbracket \cdot a_m ,$$

and this implies, by (7.1) and (7.4), that

$$\llbracket \hat{t}'(\bar{x}^m) \rrbracket \cdot \langle a_1, \dots, a_{m-1} \rangle \leq \llbracket \subseteq \rrbracket \cdot \llbracket \in \rrbracket \cdot a_m \leq \llbracket \in \rrbracket \cdot a_m ,$$

and thus

$$(a_1, \dots, a_m) \in \hat{t}'_{\mathcal{A}} .$$

Again,  $\mathbf{A} \models^{\text{LSR}} \alpha(D)$  if and only if  $D$  is satisfied by  $\mathcal{A} = \beta(\mathbf{A})$ , for any assignment  $\rho : X \rightarrow \mathcal{A}$ , i.e.,  $\mathcal{A} \models^{\text{DHL}} D$ .

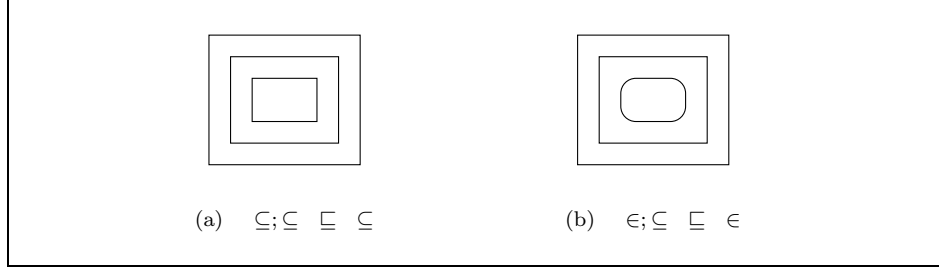
□

## 7.3 Diagram Chaining

**7.3.1 Spatial relations vs. special relations.** The map from the diagrammatic logic to the logic of special relations introduces explicit symbols — special relations  $\subseteq$  and  $\in$  — in order to express relationships implicitly captured by the *spatial* containment between boxes of a diagram. Here lies the power of diagrammatic notations with respect to equivalent textual ones, since they reflect much more its intended meaning. The *spatial* relationship of box containment is very close to the intended meaning of *special* relations  $\subseteq$  and  $\in$ . We want to take advantage of this resemblance in order to define an operational semantics for the diagrammatic logic programming language, which exploits the implicit properties of the spatial relations of a diagram in a computational way. So the properties expressed as partial order relations among compositions of special relations are implicitly given in the diagrammatic notation. See figure 7.5.

**7.3.2 Visual inferences.** Recall the chaining inference, generalized to arbitrary special relations:

$$\text{Chaining: } \frac{u \alpha s, C \quad t \beta v, T}{\sigma(u) \gamma \sigma(v), \sigma(C), \sigma(D)}$$



**Figure 7.5:** Spatially captured properties of special relations

where  $\sigma$  is a most general unifier of  $s$  and  $t$ , and  $\gamma$  is the smallest special relation such that  $\alpha; \beta \sqsubseteq \gamma$ . For our special case in our logic this rephrases as follows:

$$\frac{u \subseteq s, C \quad t \subseteq v, D}{\sigma(u) \subseteq \sigma(v), \sigma(C), \sigma(D)} \quad (7.5)$$

where  $\sigma$  is a most general unifier of  $s$  and  $t$ , and

$$\frac{u \in s, C \quad t \subseteq v, D}{\sigma(u) \in \sigma(v), \sigma(C), \sigma(D)}$$

where  $\sigma$  is a most general unifier of  $s$  and  $t$ .

Chaining is based on taking properties of special relations —expressed as a partial order— and building them into the inference system. In the above case they are

$$\subseteq; \subseteq \sqsubseteq \subseteq \quad \text{and} \quad \in; \subseteq \sqsubseteq \in$$

respectively. We now can take advantage of the fact that these properties are implicitly captured by spatial relations in our diagrams, in order to present the chaining inference diagrammatically. Figure 7.6 shows an example of a visual inference that corresponds to the chaining inference (7.5) above. We call such visual inference *diagram chaining*.

**7.3.3 Refutational proof calculus.** Since we end up proposing a diagram transformation process that resembles closely the refutation of negated queries, by means of a standard, but completely visual, resolution-based inference mechanism, we will be interested in visual forms of inferences that actually correspond to the *negative chaining* and *resolution* inferences of a chaining proof calculus:

$$\text{Negative Chaining: } \frac{s \not\subseteq t \quad u \subseteq v, C}{\sigma(s) \not\subseteq \sigma(u), \sigma(C)}$$

where  $\sigma$  is the most general unifier of  $t$  and  $v$ .  $C$  is the body of the Horn clause.

$$\text{Resolution: } \frac{s \not\subseteq t \quad u \in v, C}{\sigma(C)}$$

where  $\sigma$  is the most general unifier of  $s$  and  $u$ , and  $t$  and  $v$ , respectively. The corresponding diagrammatic inferences are shown in figure 7.7.

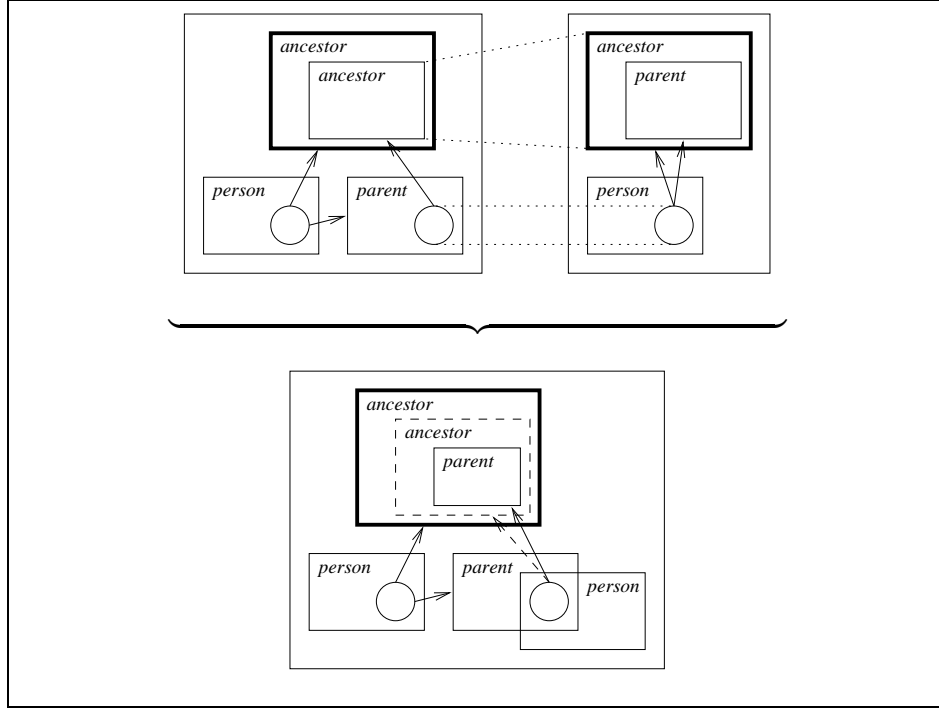
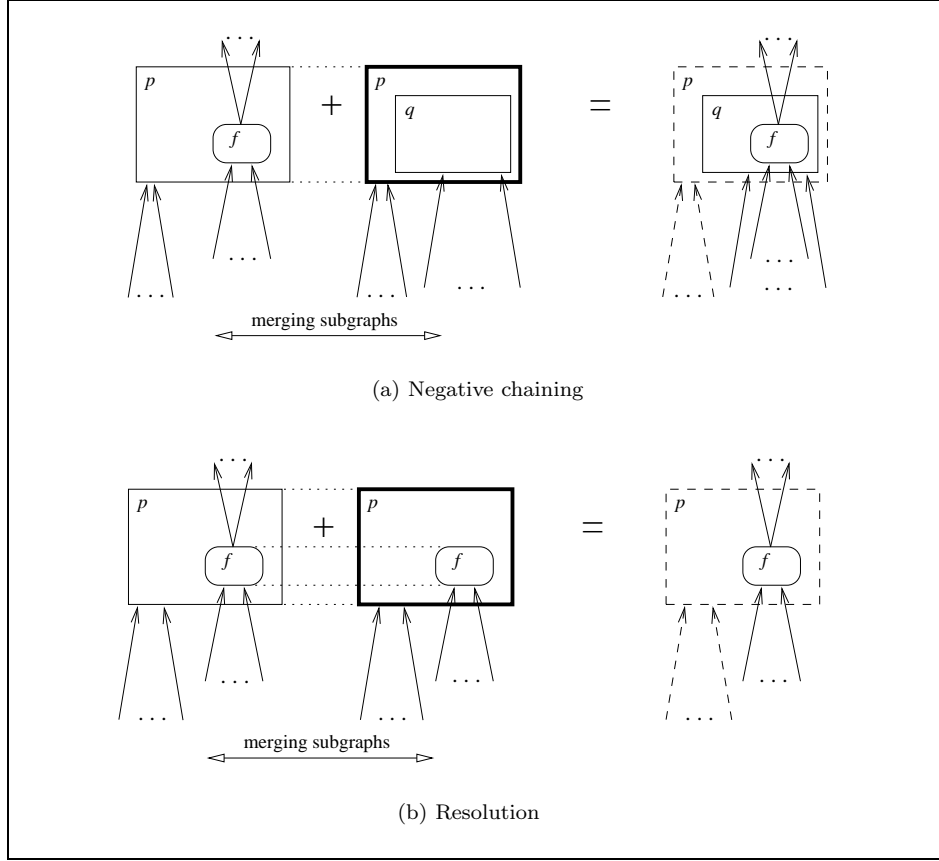


Figure 7.6: Diagram chaining

Notice, for instance, how nicely the negative chaining inference intuitively visualizes in what way the compositional property of membership with inclusion is exploited: The inference captures our attempt to prove the membership of  $f$  in  $p$  through first proving its membership in  $q$ . When we eventually proof this latter membership, the first one will follow trivially by transitivity, which is directly and intuitively captured in our diagrammatic notation.

The diagrammatic chaining inferences in figure 7.7 were originally called *subset rule* and *membership rule* in (Puigsegur et al., 1997), and they both involve a process of merging parts of the diagram, namely those subgraphs rooted on the matching nodes. Such merging process corresponds to a unification of visual terms, which Puigsegur and Agustí put forth in detail in (Puigsegur and Agustí, 1998).

**7.3.4 Discussion.** We hope that we have been able to show that conventional textual syntax not only is not indicative of the semantics of relations as sets, but tells nothing about the inference with Horn clauses. We believe diagrammatic syntax is more fit to make intuitive the resolution inference rule, which corresponds directly to the transitivity of inclusions, given for free in the diagrammatic notation. The intrinsic property that makes diagrammatic notation so useful as reasoning mechanism, exploiting a property of the intended models one is reasoning about, is called by Shimojima a ‘free ride’ (Shimojima,

**Figure 7.7:** Diagrammatic inferences

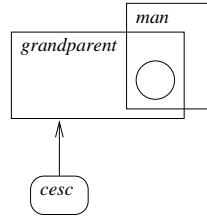
1996). For instance, the resolution inference rule applied to the two Horn clauses defining the *ancestor* relation considered above is made intuitive by the transitivity of box containment in the visual inference depicted in figure 7.6. The inferred clause corresponds to the special case of ancestors which are grandparents.

## 7.4 Diagram Transformation

Since visual inferences in general could become too complicated diagrammatically, we have explored, together with Puigsegur and Agustí, some advantages of these visual inferences in query answering (Puigsegur et al., 1997). We have customized the visual inferences discussed above to lay down a diagram transformation process for visually answering queries put on a diagrammatic logic program, as the one of figure 7.4. Here, we only want to sketch this transformation process, since all its details can be found in (Agustí et al., 1998b; Puigsegur

and Agustí, 1998) and in Puigsegr's forthcoming dissertation (Puigsegr, 2000).

**7.4.1 Query diagrams.** We provide a way to ask questions about a diagrammatic program by means of *query diagrams*. They are actually standard diagrams as described in section 7.1, with the difference that no box is highlighted with thick lines, and no box is contained within another, since no predicate is being defined. Instead query diagrams express a set of existential memberships to be proved. Suppose, given the visual program of figure 7.4, we want to know the grandfathers of Cesc. We will draw a query diagram as follows



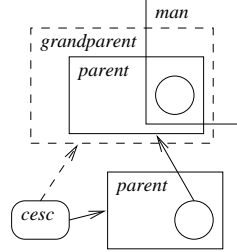
whose textual equivalent would be

$$\exists x \quad x \in \text{grandparent}(\text{cesc}) \wedge x \in \text{man} ?$$

The way we answer this query resembles the way Prolog would answer existentially quantified queries, namely by actually refuting their negation.

**7.4.2 Query answering.** Queries are answered by transforming *query diagrams* into *answer diagrams* applying visual inference rules given in section 7.3. The idea behind this diagram transformation is to complete the query with the trace of its proof, while instantiating circles (i.e., variables) by unification. As in conventional logic programming, we answer a query by exploring the definitions given in the program. We look for a definition diagram whose highlighted box has the same label as one of the boxes of the query diagram, and whose overall structures unify, and merge it into the query diagram. This will become clear by looking at an example.

Let us answer the simple query represented above. We have two different possibilities, since we can apply the subset rule (negative chaining) to the *grandparent* box or the membership rule (resolution) to the *man* box. Here, we do not consider specific deduction strategies, this is the subject of future work. Therefore we do not bother about the possible order in which boxes may be solved, and non-deterministically choose the first option. Recall that this means to unify the *grandparent* box with its definition in figure 7.4, transforming the query diagram by unifying the two circles—the argument of each *grandparent* box—and by adding to the query diagram all boxes of the definition of *grandparent*.



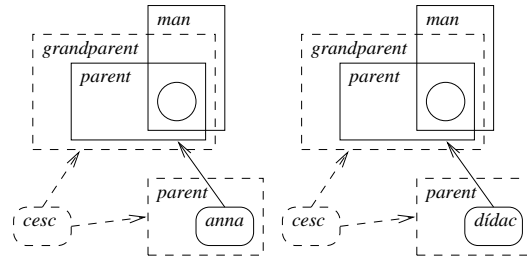
This inference step captures the idea that we are going to prove the existence of an element in the *grandparent* box by means of its membership in the *parent* box contained in it. Therefore we place the circle contained in the *grandparent* box also within the *parent* box.

**7.4.3 Proof tracing.** At this point, the first distinctive feature of a visual syntax with respect to a textual one arises. Note that the resulting query diagram reflects, both, the new inferred membership of the goal variable into the *parent* box and the old one into the *grandparent* box. This way, we can keep track of the original query and its proof during subsequent diagram transformations. The transitivity of the membership relation with respect to subset inclusion is clearly visualized by the diagram, since the circle contained in the *parent* box is obviously also within the *grandparent* box. This captures directly a basic kind of computational logic reasoning.

Keeping track of the original query forces us to distinguish goals that still need to be solved from the already solved ones. That is why we have drawn the solved *grandparent* box with dashed lines. This is how we represent solved goals. Boxes representing goals still to be solved remain drawn by solid lines.

The inclusion of circles into the *parent* boxes has to be checked before we can qualify the resulting diagram as a valid inference, otherwise we will have to backtrack. We choose to apply the membership rule by unifying the mostly instantiated *parent* box with its definition in figure 7.4.

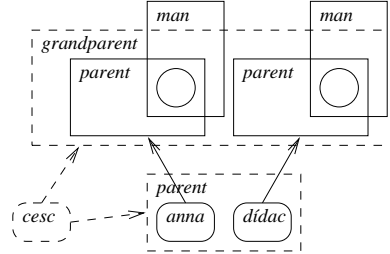
Since in this definition diagram *two* graphical items are contained in its highlighted box, we have two alternatives to instantiate the circle within *parent*, either by round box *anna* or *dídac*.



**7.4.4 Simultaneous representation of alternatives.** A new additional advantage of the visual syntax is that it easily allows us to represent both alter-



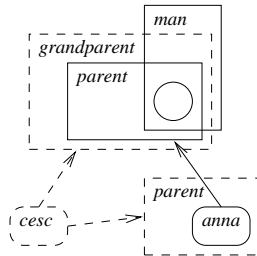
natives seen previously (*anna* and *didac*) in one single diagram:



Keeping several alternatives simultaneously is actually ‘multiply instantiating’ the circle within *parent* with *anna* and *didac*. Whenever such a multiple instantiation occurs we will have to duplicate all those boxes that also contained or had as arguments the variable (or term containing the variable) where the multiple instantiation has occurred. And again, when we duplicate these boxes, we might have to duplicate other boxes that share variables with them. A formal definition of this duplication operation can be seen in (Puigsegur and Agustí, 1998).

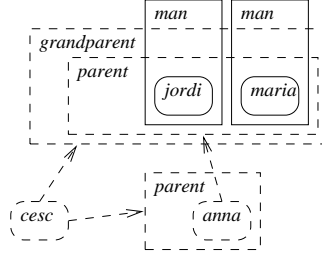
Different alternatives due to multiple instantiation introduce disjunction in the query, and consequently, we need to make explicit the logical relations between boxes of the query diagram. In our case, both, *parent* boxes and the *man* box need to be solved — they form a *conjunction*. But their duplications are different alternatives and therefore form a *disjunction*. It is well known in the diagrammatic reasoning community that disjunction is a representationally ‘thin’ concept, a concept that is difficult to capture diagrammatically (Barwise and Hammer, 1996). Puigsegur and Agustí choose an *AND-OR annotation tree* for this purpose, where the boxes of the query diagram are its leaves (Puigsegur and Agustí, 1998).

**7.4.5 Failure.** We cannot solve one of the two alternatives, because we do not dispose of a unifying definition diagram for finding a parent of *didac*. The box fails and has to be erased from the diagram, together with all other boxes attached to it by conjunction:

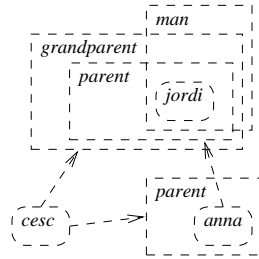


This failure doesn’t affect the solvability of the whole query, because there is still another alternative to be explored. Further transformation steps in our

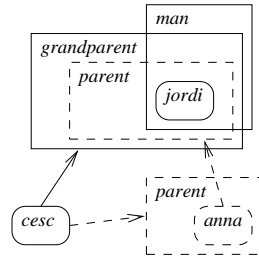
example involve a membership inference to solve the *parent* box yielding two new alternatives, *jordi* and *maria*.



The alternative *maria* fails in solving the remaining *man* box applying a membership inference, but this same inference succeeds for alternative *jordi*, eventually yielding



**7.4.6 Answer diagram.** The answer we have obtained to the original query is that Jordi is a grandfather of Cesc. In order to make the answer more readable, we may highlight the boxes of the original query diagram by drawing them with solid lines. We call such a diagram an *answer diagram*, the original query diagram plus a trace of its proof:



## 7.5 Conclusion

We have presented a completely visual inference mechanism for the visual programming language currently under development (Puigsegur, 2000). Although the rules are formally based on the chaining calculus, the visual inference mechanism is intrinsically different. Visual queries differ from textual ones, since

they involve different kinds of query transformations, providing a new and interesting framework for exploring and developing visual inference systems. The main difference with regard to a conventional calculus is that in Puigsegur and Agustí's diagrams we do not erase boxes (predicates) that have already been refuted, because we want to keep track of the trace of the proof. Another difference is when we are applying diagrams whose highlighted boxes contain more than one graphical item, and therefore various alternatives for solving a goal arise. Further development of this visual operational semantics has been done by Puigsegur and Agustí in (Puigsegur and Agustí, 1998).

The formalization we have done of Puigsegur and Agustí's visual Horn logic as customized higraphs also opens new interesting perspectives for further studying visual inference mechanisms: it provides the necessary framework for formally defining the visual inferences put forth in this chapter as a merging of two higraphs. But before we actually attempted to completely develop such formal definition, we first studied how diagrammatic inferences based on the merging of graphs actually work in the more general framework provided by category theory. This is the subject of the next chapter.



## Chapter 8

# On Diagrammatic Reasoning in Category Theory

Category theory is one of those branches of mathematics that heavily uses diagrams, not as an informal aid, but as a formally-defined mathematical object. The so called *commutative diagrams* are the category theorist's way of expressing equations. Categories itself are best understood having a graphical image in mind, since a category is actually a graph with some additional structure<sup>1</sup>, and can be therefore easily visualized as a graph. In addition, proofs of theorems in category theory are often constructed with a technique known as *diagram chasing*, based on pasting several commutative diagrams together. In fact, for many people proofs expressed in this way are much easier to understand than by applying equational reasoning inferences. The use of graphs and commutative diagrams for specifying mathematical structures has been thoroughly studied by means of the notion of *sketch*, first introduced by Ehresmann (Ehresmann, 1968). Sketches are a graph-based logic and therefore an alternative to traditional string-based logic for specifying mathematical structures (Bagchi and Wells, 1997a).

Furthermore, there is a well-known relationship between category theory and computer science. Category theory has provided theoretical foundations to the design of programming and specification languages, together with their semantic models. Recently, category theory, together with its diagrammatic notation, serves also as guideline for the emerging theory of information flow (Barwise and Seligman, 1997).

Very often diagrammatic notations capture a large amount of information in a much more compact and clear way than equivalent textual ones, because they somehow resemble what they represent in a much closer way than textual

---

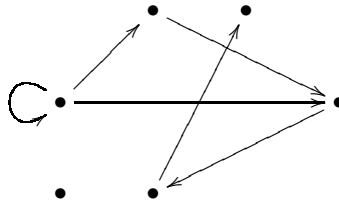
<sup>1</sup>To be rigorous we should talk about *small* categories, but we shall not worry about foundational issues here.

ones do. Barwise and Hammer suggests to provide a definition of resemblance by the use of a precise mathematical concept of *homomorphism* between an abstract notion of diagrammatic representations and an abstract notion of what they represent (Barwise and Hammer, 1996). We believe that the framework provided by general logics (see section 5.1) may be suitable for describing such resemblance, at least in cases for logical systems based on sketches. Recall that in this framework the syntax of a logic is presented as a category of signatures, abstracting the kind of symbols one actually uses to write down sentences. This abstraction does not restrict a signature to be of textual manner, but allows it to be of any kind, and therefore also one of diagrammatic nature.

In this chapter we motivate the use of visual representations, like graphs, and diagrammatic reasoning mechanisms, like diagram chasing, analyzing them from the perspective of ‘formal diagrammatic reasoning’, *i.e.* by putting the strength on how the manipulation of commutative diagrams by diagram chasing captures the reasoning process when proving theorems in category theory. Here we only formalize diagram chasing, and thus, we will ignore other important diagrammatic reasoning mechanisms common in category theory, for instance, for establishing the uniqueness of an arrow. We want to explore how visual aspects of category theory, not only diagram chasing, can inspire us in the design of visual declarative programming languages, and if these aspects can be useful from the operational point of view.

## 8.1 Graphs and Categories

What are the signatures —the abstract mathematical objects— mathematicians use when they specify and reason in category theory by means of diagrammatic representations? Let us first observe the following representation:



We may identify two kind of visual items, bullets and arrows. We also may observe that bullets and arrows are in relation to each other: For every arrow there is always a unique bullet on its source and also a unique bullet on its target.

The mathematical model we grasp by observing the representation above is a well-known one, namely a (directed) graph.

**8.1.1 Definition.** A *graph*  $\mathcal{G}$  consists of a set of nodes  $G_0$  and a set of arrows  $G_1$ , together with two total functions  $source : G_1 \rightarrow G_0$  and  $target : G_1 \rightarrow G_0$ .

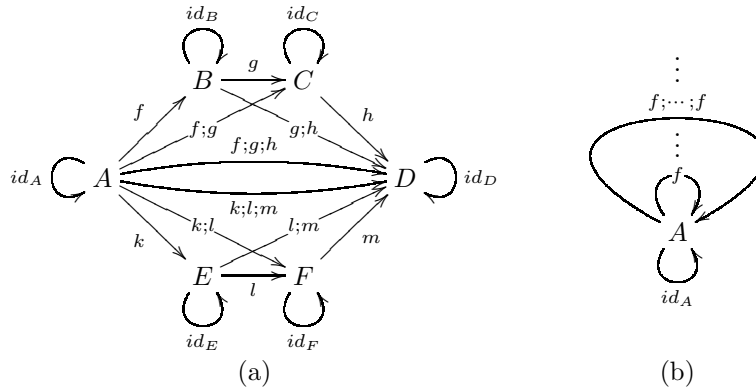
This should not surprise us at all, because there is a strong resemblance between the mathematical notion of graph and its physical representation. Since (small) categories are graphs with some additional structure, mathematicians use the same kind of representation for describing categories and reasoning about them. Let us analyze that a little bit closer:

**8.1.2 Definition.** A *category*  $\mathcal{C}$  is a graph<sup>2</sup> together with a function assigning to each pair of arrows  $f$  and  $g$  in  $C_1$ , with  $\text{target}(f) = \text{source}(g)$ , the composite arrow  $(f;g)$ <sup>3</sup>, with  $\text{source}(f;g) = \text{source}(f)$  and  $\text{target}(f;g) = \text{target}(g)$ , and a function assigning to each node  $A$  in  $C_0$  an identity arrow  $\text{id}_A$  in  $C_1$ , with  $\text{source}(\text{id}_A) = \text{target}(\text{id}_A) = A$ , such that, if  $f, g, h$  are arrows in  $C_1$ ,

$$\begin{aligned} (f;g);h &= f;(g;h) \\ \text{id}_{\text{source}(f)};f &= f \\ f;\text{id}_{\text{target}(f)} &= f \end{aligned}$$

In a category, nodes are often called *objects* and arrows *morphisms*.

By representing categories as graphs, we clearly indicate which arrows have a common source or target, much more so than if we use a linear listing of the arrows with the source and target given for each of them, instead. Unfortunately, due to the additional structure of categories (composite arrows and identity arrows), its representations by means of finite graphs —each object represented by a node and each morphism by an arrow— can be quite complicated even for very simple categories, or actually impossible at all, due to infinite number of distinct composite arrows:



But, we can represent the morphisms of a category by means of *paths* in a *graph* instead of directly by arrows.

<sup>2</sup>This way we only define small categories; large categories are captured by a more general notion of graph, where nodes and arrows form collections rather than sets.

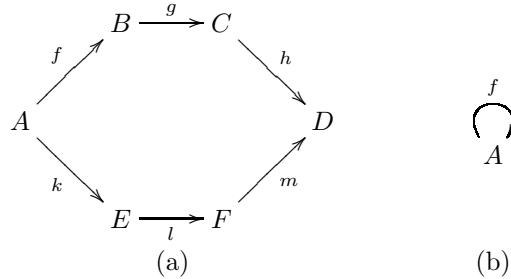
<sup>3</sup>Usually composition is written  $g \cdot f$ . We use its ‘diagrammatic’ notation instead. As its name indicates, it is closer to the diagrammatic representation of composition by means of paths in a graph as discussed in this section.

**8.1.3 Definition.** In a graph  $\mathcal{G}$ , a *path* from node  $A$  to node  $B$  of length  $n \geq 0$  is a sequence  $p = \langle f_1, f_2, \dots, f_n \rangle$  of arrows, for which

$$\begin{aligned} \text{source}(f_1) &= A \\ \text{target}(f_i) &= \text{source}(f_{i+1}) \quad \text{for } i = 1, \dots, n-1 \\ \text{target}(f_n) &= B \end{aligned}$$

We will denote a path  $p$  from  $A$  to  $B$  with  $A \rightsquigarrow^p B$ . If  $n = 0$ , we say that the path is *empty*. A path starting and ending at the same node is called a *cyclic path*.

Paths in a graph capture in an elegant way the additional structure of categories, and their properties. A path represents the composition of the morphisms represented by its arrows. An empty path on a node represents the identity morphism. The following graphs represent the categories above, using the notion of path:



The resulting graphs, though loosing on degree of ‘resemblance’ with respect to the categories they represent, they remain being highly resemblant, and in addition we obtain other advantages. We gain in clear representations of categories, and are able to express equality of morphisms by means of ‘diagrams’ in an elegant way, as we will see in the next section.

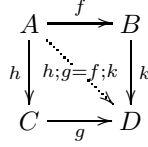
We can now answer the question put at the beginning of this section: When mathematicians specify and reason in category theory by means of diagrammatic representations, the kind of signatures they use are actually graphs: a set of object symbols, a set of morphism symbols, and relations between them determined by the sources and targets of morphisms.

## 8.2 Commutative Diagrams

The basic task in categorical proof is to show the existence of a morphism, or to show the equality of two morphisms, when some other morphisms and objects are given. Category theorists use *diagrams* to state equality of morphisms, by a *semantic convention* on these diagrams, namely that all paths with the same source and the same target are considered to represent equal morphisms. By this semantic convention, the following diagram states that morphism  $f; k$ ,



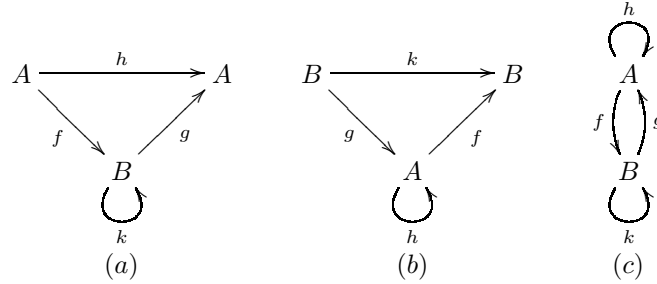
represented by path  $\langle f, k \rangle$ , and morphism  $h; g$ , represented by path  $\langle h, g \rangle$ , are equal, i.e.  $f; k = h; g$ :



We call such a diagram (without the dotted arrow), together with this semantic convention, a *commutative diagram*. A commutative composite diagram is a very economical way of showing several equalities simultaneously without duplication of subterms.

Although the representation above is that of a graph, the precise notion of *diagram* is much more subtle. In order to use commutative diagrams to represent equalities of morphisms, we need to express much more within one unique representation than a graph would allow.

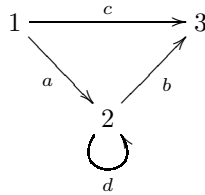
**8.2.1 Example.** Observe the following three diagrams:



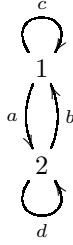
They are different diagrams, though for all three of them their nodes are  $A, B$ , and their arrows are  $f, g, h, k$  such that

$$\begin{aligned} \text{source}(f) &= \text{source}(h) = A \\ \text{source}(g) &= \text{source}(k) = B \\ \text{target}(g) &= \text{target}(h) = A \\ \text{target}(f) &= \text{target}(k) = B \end{aligned}$$

Diagrams (a) and (b) have the same *shape*, namely the one given by the following graph:



Nodes and arrows, though, are labeled differently. Diagram (c) above has a completely different shape, namely the one given by the following graph:



A diagram is, therefore, the drawing of a graph that determines the shape, and the labeling of nodes and arrows with respect to the graph we want to describe. This is captured by a graph homomorphism:

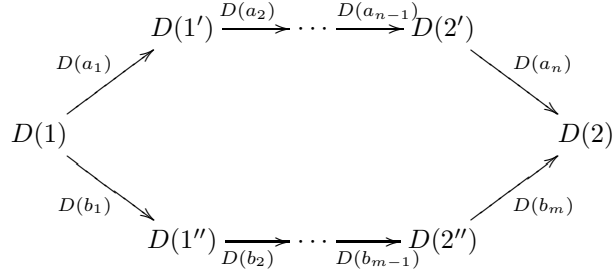
**8.2.2 Definition.** A *graph homomorphism*  $\phi : \mathcal{G} \rightarrow \mathcal{H}$  is a pair of functions  $\phi_0 : G_0 \rightarrow H_0$  and  $\phi_1 : G_1 \rightarrow H_1$  such that for every arrow  $f$  of  $\mathcal{G}$ ,

$$\begin{aligned} \text{source}_{\mathcal{H}}(\phi_1(f)) &= \phi_0(\text{source}_{\mathcal{G}}(f)) \\ \text{target}_{\mathcal{H}}(\phi_1(f)) &= \phi_0(\text{target}_{\mathcal{G}}(f)) \end{aligned}$$

**8.2.3 Definition.** Let  $\mathcal{I}$  and  $\mathcal{G}$  be graphs. A *diagram in  $\mathcal{G}$  of shape  $\mathcal{I}$*  is a graph homomorphism  $D : \mathcal{I} \rightarrow \mathcal{G}$ .

**8.2.4 Example (continued).** For diagram (a), the homomorphism is  $D(1) = D(3) = A$ ,  $D(2) = B$ ,  $D(a) = f$ ,  $D(b) = g$ ,  $D(c) = h$ , and  $D(d) = k$ , taking the first of the two shape graphs above. For diagram (b), the homomorphism is  $D(1) = D(3) = B$ ,  $D(2) = A$ ,  $D(a) = g$ ,  $D(b) = f$ ,  $D(c) = k$ , and  $D(d) = h$ , taking the first of the two shape graphs, too. For diagram (c), the homomorphism is  $D(1) = A$ ,  $D(2) = B$ ,  $D(a) = f$ ,  $D(b) = g$ ,  $D(c) = h$ , and  $D(d) = k$ , now taking the second of the two shape graphs above.

**8.2.5 Definition.** In a category  $\mathcal{C}$ , a diagram  $D : \mathcal{I} \rightarrow \mathcal{C}$  is *commutative* (or *commutes*) if for any pair of nodes 1 and 2 of  $\mathcal{I}$  and any two paths  $\langle a_1, \dots, a_n \rangle$  and  $\langle b_1, \dots, b_m \rangle$  from 1 to 2, we have that the two morphisms  $D(a_1); \dots; D(a_n)$  and  $D(b_1); \dots; D(b_m)$  obtained by composition in  $\mathcal{C}$  are the same:

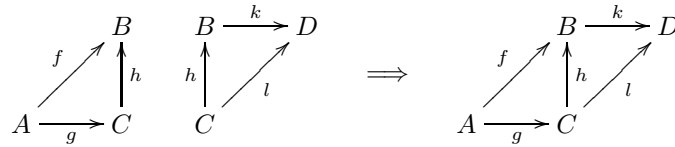


**8.2.6 Observation.** Having in mind that empty paths represent identity morphisms, observe the distinct assertions we are making with the diagrams of example 8.2.1, when considered to be commutative: Diagram (a) asserts that  $f;g = h$  and  $k = id_B$ . Diagram (b) asserts that  $g;f = k$  and  $h = id_A$ . Finally, diagram (c) asserts that  $f;g = h = id_A$  and  $g;f = k = id_B$ .

## 8.3 Reasoning by Diagram Chasing

A conventional style of proof in category theory is *diagram chasing*. Diagram chasing is the technique by which category theorists reason with commutative diagrams. It is an easy, visual, reliable style of proving equality of morphisms.

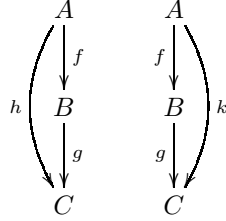
**8.3.1 Free rides.** Consider the following diagram chasing step:



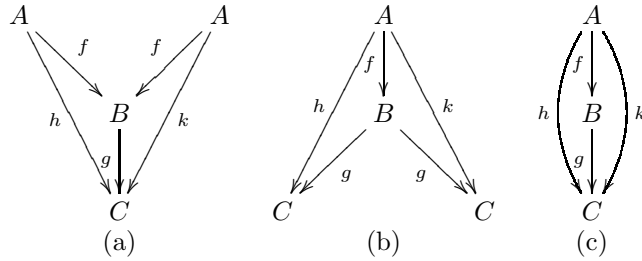
The two diagrams pictured to the left state that  $f = g;h$  and  $l = h;k$ . Diagram chasing consists of pasting both diagrams together along the common arrow  $h$  to obtain the new commutative diagram to the right. From this diagram, and by the semantic convention established on commutative diagrams discussed in section 8.2, we can grasp a new fact not previously stated, namely that  $f;k = g;l$ . This is a consequence of chasing the diagrams.

This form of diagrammatic reasoning provides us with *free rides*, as defined by Shimojima (1996): By chasing two diagrams, new paths with coinciding sources and targets may appear in the new diagram, without explicitly drawing them; we may get them for free! By the semantic convention of ‘commutation of a diagram’, these paths will be considered to represent equal morphisms.

**8.3.2 Correct and incorrect diagram chasings.** Diagram chasing is not a mere pasting together of two diagrams along common arrows, since such pasting must be done in a correct way. Consider the following two diagrams:

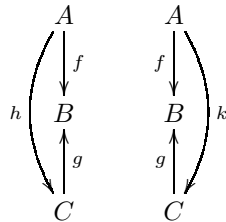


We may think of three different correct alternatives to chase the two original commutative diagrams together, namely along arrow  $f$ , arrow  $g$ , or both at once:

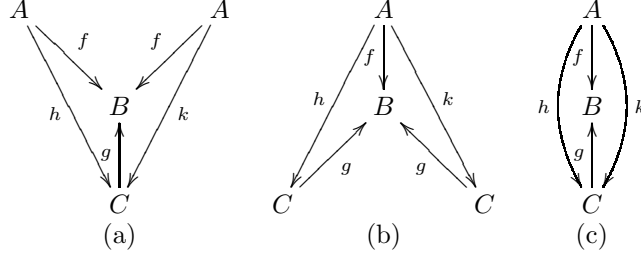


We do not obtain new equalities from diagrams (a) and (b), they do not provide us any ‘free ride’, but from diagram (c) we can grasp the equality of morphisms  $h = k$ . It seems, therefore, that in order to deduce as many new equalities thanks to ‘free rides’ provided by diagram chasing, we would have to paste diagrams along as many common arrows as possible. But this is not always the case.

**8.3.3 Observation.** Let us now chase the following diagrams, which are almost identical to the previous ones, but with the direction of arrow  $g$  reversed:

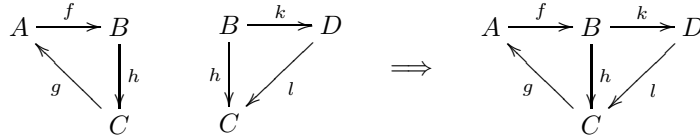


Observe the possible alternatives we might consider now:



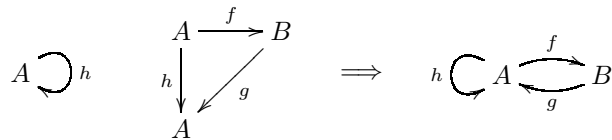
As before, from diagram (c) we would obtain that  $h = k$ , but now this cannot be deduced equationally from  $f = h; g$  and  $f = k; g$ . Instead, diagrams (a) and (b) are correct diagram chasings. In order for diagram chasing to provide correct ‘free rides’ we have to paste along arrows that form together a path, which is the key notion in the semantic convention.

**8.3.4 Cyclic paths in diagram chasing.** It is more subtle to state the correctness of diagram chasing when cyclic paths are involved, due to the semantic convention put on commutative diagrams that morphisms represented by paths beginning and ending at the same node are equal to the identity morphism of that node. Consider the following diagram chasing:



The resulting diagram asserts —beside other equalities— that  $l; g; f; k = id_D$ , which cannot be equationally deduced from the equations stated by the original diagrams. In general we will disallow diagram chasings on paths if some of their arrows belongs to a cyclic path, though this restriction can be relaxed for some cases.

The problem arises when a node not belonging to a cyclic path before the chasing takes place ends up within a cyclic path afterwards. This is what happened to node  $D$  in our example. By the following diagram chasing we obtain that  $g; f = id_B$ , which cannot be derived from the original equalities, because we have brought node  $B$  into a cyclic path, due to the loop  $h$  (which is an arrow forming a cyclic path by itself):



Notice that everything works fine when the affected nodes already belonged to cyclic paths. The diagram chasing below is correct, since the free ride  $f; g = id_B$

can be derived by equational reasoning. In this case node  $B$  was part of a cyclic path before the chasing took place:

$$\begin{array}{c}
 A \xrightarrow{h} A \\
 \text{---} \\
 A \xleftarrow{f} B \\
 \text{---} \\
 A \xrightarrow{g} B
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{c}
 A \xrightarrow{h} A \\
 \text{---} \\
 A \xleftarrow{f} B \\
 \text{---} \\
 A \xrightarrow{g} B
 \end{array}$$

**8.3.5 The formal definition.** Let us give a formal definition of diagram chasing. We will first define an equivalence relation on the set of arrows and set of nodes of both original shape graphs. The idea is to merge the two shape graphs into one, at the same time we are making equivalent those nodes and arrows on which the diagrams are chased:

**8.3.6 Definition.** Given two diagrams  $D : \mathcal{I} \rightarrow \mathcal{G}$  and  $D' : \mathcal{I}' \rightarrow \mathcal{G}$  and two non-empty paths,  $p = \langle a_1, \dots, a_n \rangle$  and  $p' = \langle a'_1, \dots, a'_n \rangle$  in  $\mathcal{I}$  and  $\mathcal{I}'$  respectively,  $n \geq 1$ , such that,

1. for all  $i \in [1 \dots n]$ ,  $D(a_i) = D'(a'_i)$ , and
2. no arrow  $a_i$  of  $p$  and no arrow  $a'_j$  of  $p'$  belongs to a cyclic path, neither in  $\mathcal{I}$  nor in  $\mathcal{I}'$ ,

we define the relation  $\sim$  on the set of arrows  $I_1 \cup I'_1$  and on the set of nodes  $I_0 \cup I'_0$  to be the minimal equivalence relation satisfying, for all  $i \in [1 \dots n]$ ,

$$\begin{aligned}
 a_i &\sim a'_i \\
 \text{source}(a_i) &\sim \text{source}'(a'_i) \\
 \text{target}(a_i) &\sim \text{target}'(a'_i) .
 \end{aligned}$$

We are now ready to give the actual definition of ‘diagram chasing’, using the equivalence relation  $\sim$  on nodes and arrows just defined.

**8.3.7 Definition.** We define the diagram  $D'' : \mathcal{I}'' \rightarrow \mathcal{G}$  obtained by chasing diagrams  $D$  and  $D'$  along paths  $p$  and  $p'$ , and write  $\{D, D'\} \vdash_0^{DC} D''$ , in the following way:

1. The set of nodes and set of arrows of its shape graph are

$$\begin{aligned}
 I''_0 &= (I_0 \cup I'_0) / \sim \\
 I''_1 &= (I_1 \cup I'_1) / \sim
 \end{aligned}$$

with

$$\begin{aligned}
 \text{source}'' &= (\text{source} \cup \text{source}') / \sim \\
 \text{target}'' &= (\text{target} \cup \text{target}') / \sim .
 \end{aligned}$$

2. The graph homomorphism is

$$\begin{aligned} D_0'' &= (D_0 \cup D_0') / \sim \\ D_1'' &= (D_1 \cup D_1') / \sim . \end{aligned}$$

**8.3.8 Warning.** Notice that we have made abuse of notation by defining *source''*, *target''*,  $D_0''$ , and  $D_1''$  as quotients by  $\sim$  in the obvious way. Recall from definition 8.3.6 that  $\sim$  depends on the paths we are chasing. It is easy to prove that  $D''$  is indeed a graph homomorphism.

## 8.4 Diagram Chasing vs. Equational Reasoning

We now formally prove that reasoning with diagrams and diagram chasing is indeed a correct alternative to traditional equational reasoning, by proving that every equality of morphisms derivable by diagram chasing, as defined in the previous section, is also derivable by conventional equational reasoning. We do this by defining a map of entailment systems (see section 5.1). We need therefore to first define the entailment systems of diagram chasing and equational reasoning, respectively.

**8.4.1 Entailment system.** In section 8.1 we saw that the signatures we are dealing with are graphs. The category **Grf** of graphs and graph homomorphisms is therefore the category of signatures and signature homomorphisms. Given a graph  $\mathcal{G}$ , the diagrams in  $\mathcal{G}$  play the role of well-formed sentences over this signature. Since diagrams in  $\mathcal{G}$  are graph homomorphisms from a shape graph to  $\mathcal{G}$ , they form a category, namely the slice category **Grf**/ $\mathcal{G}$ . We can define a functor  $diag : \mathbf{Grf} \rightarrow \mathbf{Set}$  assigning to each graph  $\mathcal{G}$  the set of all diagrams in  $\mathcal{G}$ , namely  $O(\mathbf{Grf}/\mathcal{G})$  where  $O : \mathbf{Cat} \rightarrow \mathbf{Set}$  is the object functor. This functor is therefore the one assigning to each signature the set of all well-formed sentences over it. Given a graph  $\mathcal{G}$ , diagram chasing determines an entailment relation  $\vdash^{DC}$  between sets of diagrams and single diagrams, by defining it as the reflexive, monotonic, and transitive closure of  $\vdash_0^{DC}$ . Consequently, we have that the category **Grf**, the functor  $diag$ , and the entailment relation  $\vdash^{DC}$ , all together constitute an *entailment system*  $(\mathbf{Grf}, diag, \vdash^{DC})$ , the entailment system of diagram chasing (see also section 5.1).

**8.4.2 Sketches.** *Theories* (or, to be more exact, *theory presentations*) are given by a signature together with a set of sentences over this signature. In our particular entailment system of diagram chasing, signatures are graphs, and sentences are diagrams in that graph. Theories are therefore *linear sketches*, as defined by Barr and Wells (Barr and Wells, 1995):

**8.4.3 Definition.** A *linear sketch* is a pair  $\mathcal{S} = (\mathcal{G}, \mathcal{D})$  with  $\mathcal{G}$  a graph and  $\mathcal{D}$  is a set of diagrams in  $\mathcal{G}$ .

Since, given an entailment system, theories form a category (Meseguer, 1989), we have that linear sketches do so, too. Let **LinSk** denote the category of linear sketches.

**8.4.4 Entailment system.** For the entailment system of equational reasoning, the category of signatures and signature homomorphisms is **Grf**, too, because we are dealing with a set of morphism symbols, a set of object symbols<sup>4</sup>, and relations between them determined by the sources and targets of these symbols. The sentences are equations, *i.e.* pairs of well-formed strings over these morphism and object symbols, plus the composition symbol<sup>5</sup>. We write equations with the infix symbol  $\approx$  for equality. The functor  $eq : \mathbf{Grf} \rightarrow \mathbf{Set}$  assigns to each signature the set of its well-formed equations. The entailment relation  $\vdash^{ER}$  is equational reasoning with strings, capturing the reflexivity, symmetry, transitivity, and the congruence properties of equality with respect to composition. The entailment system of equational reasoning is therefore  $(\mathbf{Grf}, eq, \vdash^{ER})$ <sup>6</sup>. Analogous to linear sketches (*i.e.* theories of the entailment system of diagram chasing), equational theories form a category, and we denote it **EqTh**.

**8.4.5 Map of entailment systems.** The following is an example of an equational reasoning process. It captures the diagram chasing given in 8.3.1:

$$\frac{\frac{f \approx g; h}{f; k \approx g; h; k} \quad \frac{\frac{l \approx h; k}{h; k \approx l}}{g; h; k \approx g; l}}{f; k \approx g; l}$$

The reasoning process done by diagram chasing, instead, spares us of many intermediate inference steps, compared to equational reasoning, because the semantic convention put on commutative diagrams—that different paths with coinciding sources and targets represent equal morphisms—implicitly captures ‘graphically’ the reflexivity, symmetry, transitivity, and congruence properties of equality with respect to composition, *without explicitly performing the inference steps*, as shown in figure 8.1.

Let us see these issues in a formal way, by studying the map from the entailment system of diagram chasing to the entailment system of equational reasoning. Given a graph  $\mathcal{G}$ , if  $D$  is a diagram in  $\mathcal{G}$  of shape  $\mathcal{I}$  and  $p = \langle a_1, \dots, a_n \rangle$  is a path in  $\mathcal{I}$ ,  $n \geq 0$ , then we denote with  $\hat{p}$  the string of morphism compositions  $D(a_1); \dots; D(a_n)$  represented by  $p$ . We map a diagram  $D$  to a set of equations, by defining  $\alpha_{\mathcal{G}}(D) = \{\hat{p} \approx \hat{p}' \mid source(p) = source(p') \wedge target(p) = target(p')\}$ . Being  $\mathcal{D}$  a set of diagrams,  $\alpha_{\mathcal{G}}(\mathcal{D}) = \bigcup_{D \in \mathcal{D}} \alpha_{\mathcal{G}}(D)$ . Theories (linear sketches) are mapped in the following way:  $\Phi(\mathcal{G}, \mathcal{D}) = (\mathcal{G}, \alpha_{\mathcal{G}}(\mathcal{D}))$ .

<sup>4</sup>Actually we are dealing with a family of morphism symbols  $id$ , subindexed by the object symbols.

<sup>5</sup>The strings are well-formed according to the sources and targets of the morphism symbols.

<sup>6</sup>To be more rigorous we should define the entailment system of equational reasoning within many-sorted equational logic with polymorphic operators (since composition of morphisms is polymorphic), but the above stated entailment system suffices for our purposes.



Reflexivity:	$\frac{}{p \approx p}$	$A \xrightarrow{p} B$
Symmetry:	$\frac{p \approx p'}{p' \approx p}$	$A \xrightarrow[p']{p} B$
Transitivity:	$\frac{p \approx p' \quad p' \approx p''}{p \approx p''}$	$A \xrightarrow[p'']{p} B$
Congruence (right):	$\frac{p \approx p'}{p; p'' \approx p'; p''}$	$A \xrightarrow[p']{p} B \xrightarrow{p''} C$
Congruence (left):	$\frac{p \approx p'}{p''; p \approx p''; p'}$	$C \xrightarrow{p''} A \xrightarrow[p']{p} B$

**Figure 8.1:** Equational inferences vs. commutative diagrams

Through the notion of ‘map of entailment systems’, we capture in a formal way the intuitive idea that every equality of morphisms derivable by diagram chasing is also derivable by equational reasoning.

**8.4.6 Proposition.** *The map  $(\Phi, \alpha) : (\mathbf{Grf}, \text{diag}, \vdash^{DC}) \rightarrow (\mathbf{Grf}, \text{eq}, \vdash^{ER})$  is a map of entailment systems.*

For  $(\Phi, \alpha)$  to be a map of entailment systems,  $\alpha : \text{diag} \Rightarrow \text{eq} \circ \Phi$  has to be a natural transformation, and  $\Phi : \mathbf{LinSk} \rightarrow \mathbf{EqTh}$  an  $\alpha$ -sensible functor<sup>7</sup>, such that  $\mathcal{D} \vdash_{\mathcal{G}}^{DC} D$  implies  $\alpha_{\mathcal{G}}(\mathcal{D}) \vdash_{\Phi(\mathcal{G}, \emptyset)}^{ER} \alpha_{\mathcal{G}}(D)$ . We refer to section 5.1 for further details on the general definition of a map of entailment systems. In the rest of this chapter we will drop the subscripts of  $\alpha$ ,  $\vdash^{DC}$ , and  $\vdash^{ER}$  when the graph  $\mathcal{G}$  is clear from the context.

That  $(\Phi, \alpha)$  is indeed a map of entailment systems is a direct consequence of lemma 8.4.11 below. But let us first prove several auxiliary lemmas, in which we use the following conventions:

**8.4.7 Conventions.** By a *path in a diagram* we actually think of a path in its shape graph. We extend *source* and *target* on arrows to paths in the following way: If  $p = \langle a_1, \dots, a_n \rangle$ ,  $\text{source}(p) = \text{source}(a_1)$  and  $\text{target}(p) = \text{target}(a_n)$ . We will also write  $p = \langle s_1, \dots, s_m \rangle$  to indicate that  $s_i$  are *subpaths* of  $p$ , such that for all  $i \in [1 \dots m - 1]$ ,  $\text{target}(s_i) = \text{source}(s_{i+1})$ . If  $D''$  is the diagram obtained by chasing diagrams  $D$  and  $D'$  along paths  $p$  and  $p'$ , respectively, we will call  $p$ ,  $p'$  and their resulting path  $p''$  in  $D''$  indistinguishably the *chased path*. We will say that a node  $n$  is *on the chased path* if it is the source or target of one of its arrows. Furthermore, if  $q = \langle a_1, \dots, a_n \rangle$  is a path in  $D''$ , we will say that

<sup>7</sup>  $\Phi$  is  $\alpha$ -sensible when the theorems of  $\Phi(\mathcal{G}, \mathcal{D})$  are completely determined by  $\Phi(\mathcal{G}, \emptyset)$  and  $\alpha_{\mathcal{G}}(\mathcal{D})$ .

$q$  is also a path in  $D$  (or in  $D'$ ) if, for all  $i \in [1 \dots n]$ , there exists an arrow  $a'_i$  in the equivalence class  $a_i$ , and  $\langle a'_1, \dots, a'_n \rangle$  is a path in  $D$  (or  $D'$ ). This latter can be also applied to single nodes and arrows.

**8.4.8 Lemma.** *Let  $D''$  be the diagram obtained by chasing diagrams  $D$  and  $D'$ . Every path  $p$  in  $D''$  can be decomposed in a sequence of subpaths  $p = \langle s_1, \dots, s_n \rangle$ , with  $n \geq 1$ , such that, for all  $i \in [1 \dots n - 1]$ , either  $s_i$  is a path in  $D$ , and then  $s_{i+1}$  is a path in  $D'$ , or else  $s_i$  is a path in  $D'$ , and then  $s_{i+1}$  is a path in  $D$ .*

PROOF: Because paths in  $D''$  are actually finite sequences of equivalence classes of arrows of the shape graphs of  $D$  and  $D'$  (see definitions 8.3.6 and 8.3.7), we can ‘isolate’ the subsequences of arrows that are completely in  $D$  or  $D'$  respectively.  $\square$

**8.4.9 Corollary.** *For all  $i \in [1 \dots n - 1]$ ,  $target(s_i)$  and  $source(s_{i+1})$  are nodes on the chased path.*

PROOF: If  $s_i$  is in  $D$ , then  $s_{i+1}$  is in  $D'$ , and since node  $target(s_i) = source(s_{i+1})$ , it must be both in  $D$  and  $D'$ , and consequently, on the chased path.  $\square$

**8.4.10 Lemma.** *Let  $D''$  be the diagram obtained by chasing diagrams  $D$  and  $D'$ . If  $p$  is a path in  $D''$ , such that its source and target is on the chased path, then there exists a subpath  $q$  of the chased path, such that  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{q} \approx \hat{p}$ .*

PROOF: By lemma 8.4.8 and corollary 8.4.9,  $p = \langle s_1, \dots, s_n \rangle$ , such that, for all  $i \in [1 \dots n]$ ,  $s_i$  is either in  $D$  or in  $D'$ , and  $source(s_i)$  and  $target(s_i)$  are on the chased path. Consequently, there is a subpath  $s'_i$  of the chased path between these two nodes, which must go also from  $source(s_i)$  to  $target(s_i)$ , otherwise it would violate the condition put on diagram chasing, namely that no arrow of the chased path belongs to a cyclic path (see definition 8.3.6). Since  $s_i$  is either in  $D$  or in  $D'$  and  $s'_i$  is in both, we have that  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{s}'_i \approx \hat{s}_i$ , for all  $i \in [1 \dots n]$ , and by the properties of equational reasoning  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{q} \approx \hat{p}$ .  $\square$

**8.4.11 Lemma.** *If  $D''$  is the diagram obtained by chasing diagrams  $D$  and  $D'$ , i.e.  $\{D, D'\} \vdash_0^{DC} D''$ , then  $\alpha(D) \cup \alpha(D') \vdash^{ER} \alpha(D'')$ .*

PROOF: Let  $u$  and  $v$  be two paths in  $D''$ , and therefore  $\hat{u} \approx \hat{v}$  is in  $\alpha(D'')$ . We have the following cases:

1.  $u$  and  $v$  are either both in  $D$  or else both in  $D'$ :  
Then, either equation  $\hat{u} \approx \hat{v}$  is in  $\alpha(D)$ , or else it is in  $\alpha(D')$ , and therefore  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{u} \approx \hat{v}$

2.  $u$  is in  $D$ , but  $v$  is not in  $D$  (or vice versa)<sup>8</sup>:

By lemma 8.4.8 and corollary 8.4.9, and by letting  $w_1$  be the empty path from  $source(v)$  to itself when  $source(v)$  is on the chased path, and  $w_3$  be the empty path from  $target(v)$  to itself when  $target(v)$  is on the chased path, we can decompose  $v$  in a sequence of three subpaths  $\langle w_1, w_2, w_3 \rangle$ , such that  $w_1$  and  $w_3$  are both in  $D$  and  $source(w_2)$  and  $target(w_2)$  are on the chased path. By lemma 8.4.10, there exists a subpath  $q$  of the chased path, such that  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{q} \approx \hat{w}_2$ . Since  $q$  is also a path in  $D$ ,  $\alpha(D) \vdash^{ER} \hat{u} \approx \hat{w}_1; \hat{q}; \hat{w}_3$ , and by the properties of equational reasoning  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{u} \approx \hat{v}$ .

3. Neither  $u$  nor  $v$  is in  $D$  or in  $D'$ :

By lemma 8.4.8 and corollary 8.4.9, we decompose  $u$  and  $v$  in the sequence of three subpaths  $\langle w_1, w_2, w_3 \rangle$  and  $\langle w'_1, w'_2, w'_3 \rangle$  respectively, such that  $w_1, w_3, w'_1$ , and  $w'_3$  are each either in  $D$  or  $D'$ , and  $source(w_2)$ ,  $target(w_2)$ ,  $source(w'_2)$ , and  $target(w'_2)$  are on the chased path. By lemma 8.4.10, there exist subpaths  $q$  and  $q'$  of the chased path, such that  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{q} \approx \hat{w}_2$  and  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{q}' \approx \hat{w}'_2$ .

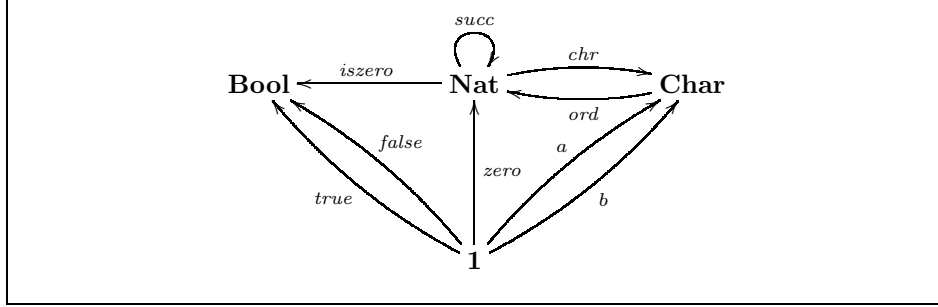
- (a) If  $q'$  is subpath of  $q$  (or vice versa), then there exists subpaths  $r_1, r_2$  such that  $q = \langle r_1, q', r_2 \rangle$ . Consequently,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{r}_1 \approx \hat{w}'_1$  and  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{r}_2; \hat{w}_3 \approx \hat{w}'_3$ . By the properties of equational reasoning,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{r}_1; \hat{q}'; \hat{r}_2; \hat{w}_3 \approx \hat{w}'_1; \hat{q}'; \hat{w}'_3$ , and consequently,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{u} \approx \hat{v}$ .
- (b) If  $q$  and  $q'$  overlap, but neither is subpath of the other, then there exist subpaths  $q'', r_1, r_2$  such that  $q = \langle r_1, q'', r_2 \rangle$  and  $q' = \langle q'', r_2 \rangle$  (or vice versa). Consequently,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{r}_1 \approx \hat{w}'_1$  and  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{r}_2; \hat{w}'_3 \approx \hat{w}_3$ . By the properties of equational reasoning,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{r}_1; \hat{q}''; \hat{w}_3 \approx \hat{w}'_1; \hat{q}''; \hat{r}_2; \hat{w}'_3$ , and consequently,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{u} \approx \hat{v}$ .
- (c) If  $q$  and  $q'$  do not overlap, then there exists a subpath  $q''$ , such that  $\langle q, q'', q' \rangle$  (or  $\langle q', q'', q \rangle$ ) is subpath of chased path. Consequently, and by the properties of equational reasoning,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{q}; \hat{q}''; \hat{q}'; \hat{w}'_3 \approx \hat{w}'_1; \hat{q}'; \hat{w}'_3$  and  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{w}_1; \hat{q}; \hat{w}_3 \approx \hat{w}_1; \hat{q}; \hat{q}''; \hat{q}'; \hat{w}'_3$ . Consequently,  $\alpha(D) \cup \alpha(D') \vdash^{ER} \hat{u} \approx \hat{v}$ .

□

**8.4.12 Remark.** From the diversity of cases one has to consider while proving lemmas 8.4.10 and 8.4.11, we can see the considerable amount of equational reasoning inferences required in order to ‘emulate’ one single diagram chasing inference  $\{D, D'\} \vdash_0^{DC} D''$ . This observation further highlights that commutative diagrams and diagram chasing somehow embed the properties of equational reasoning in a natural way.

---

<sup>8</sup> $v$  may be in  $D'$  or not, it doesn't matter.



**Figure 8.2:** Category providing types and basic operators

**8.4.13 Non-conservative map.** If by equational reasoning we cannot prove more equalities of morphisms than by diagram chasing, we say that the map of entailment system is *conservative*. Formally, the map  $(\Phi, \alpha)$  would be conservative if  $\alpha_{\mathcal{G}}(\mathcal{D}) \vdash_{\Phi(\mathcal{G}, \emptyset)}^{ER} \alpha_{\mathcal{G}}(D)$  implies  $\mathcal{D} \vdash_{\mathcal{G}}^{DC} D$ . Unfortunately, with the current formalization of diagram chasing given in definitions 8.3.6 and 8.3.7 it is not. This is due to the particular treatment we are giving to cyclic paths. Observe the following case:

$$\begin{array}{c}
 A \xrightarrow{f} B \\
 \downarrow h \quad \searrow g \\
 A
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{c}
 h \circlearrowleft A \xrightleftharpoons[g]{f} B
 \end{array}$$

We cannot deduce the equality of morphisms  $f;g$  and  $id_A$  by diagram chasing, because we disallow chasings along cyclic paths in order to avoid incorrect deductions, although  $f;g \approx id_A$  can be deduced by equational reasoning.

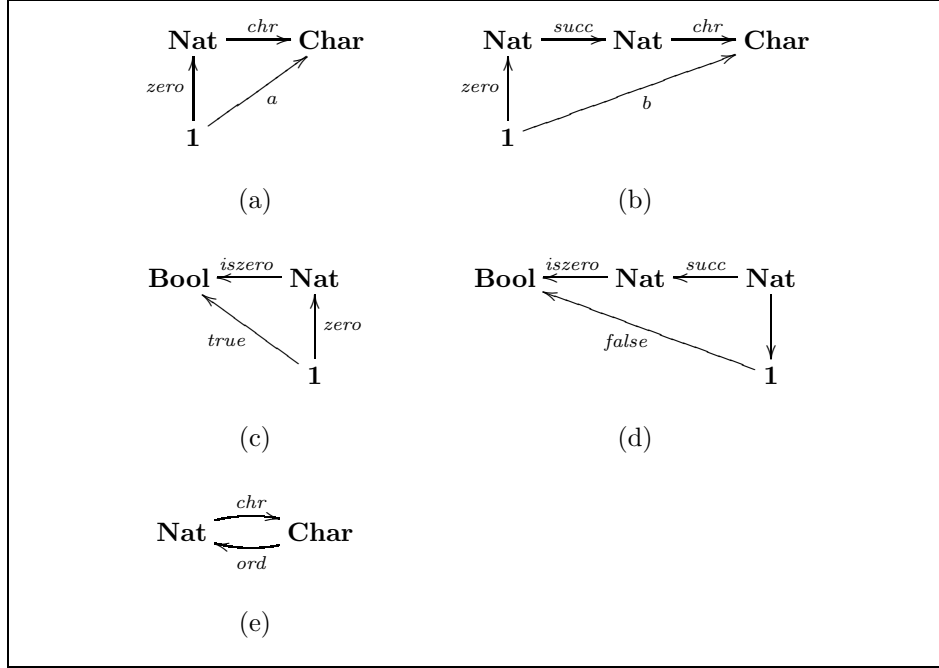
## 8.5 Computing by Diagram Chasing

We are interested in exploiting sketches as a computing device. As an example, let us see how we would graphically describe a very simple functional programming language.

**8.5.1 Example.** Figure 8.2 represents the category providing us with the basic operators of the language, together with its typing. Arrows originating at **1** are constants of their target type (e.g. *false* is a constant of type **Bool**).

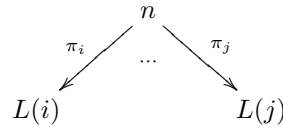
The behavior of these basic operators is specified by means of the commutative diagrams of figure 8.3, where the intended meaning of the operators in the obvious one.

**8.5.2 Cones.** Before going any further, we need to introduce another visual formalism frequently used in category theory, namely *cones*. Like diagrams, they are also a particular type of graph homomorphism:



**Figure 8.3:** Commutative diagrams specifying the behavior of operators

**8.5.3 Definition.** Let  $\mathcal{I}$  be a finite discrete graph and  $\mathcal{G}$  a graph. A *finite discrete cone in  $\mathcal{G}$*  is a graph homomorphism  $L : \mathcal{I} \rightarrow \mathcal{G}$ , a node  $n$  of  $\mathcal{G}$  and a collection of arrows  $\pi_i : n \rightarrow L(i)$ , one for each node  $i \in \mathcal{I}$ .



The node  $n$  is called the *vertex*, the diagram  $L$  the *base* of the cone.

**8.5.4 Example (continued).** We need to specify constants as nullary operators, by determining  $1$  to be a final object. This is done with the product cone with vertex  $1$  and empty base:

$1$

This description of the functional programming language is nothing else but a *finite product sketch*:

**8.5.5 Definition.** A *finite product sketch  $\mathcal{S}$*  is a triple  $(\mathcal{G}, \mathcal{D}, \mathcal{L})$  where  $\mathcal{G}$  is a graph,  $\mathcal{D}$  is a collection of diagrams in  $\mathcal{G}$ , and  $\mathcal{L}$  is a collection of cones.

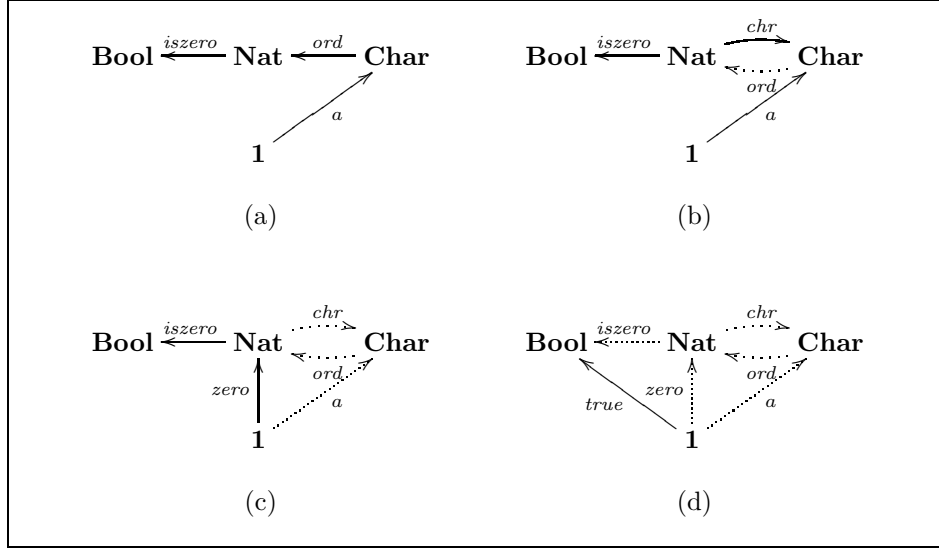


Figure 8.4: Computation by diagram chasing

We do not want to enter here in more details about sketches, because our purpose is to give the basic notions in order to present our argumentation. We refer to (Barr and Wells, 1995) for an exhaustive study on sketches.

**8.5.6 Computation.** Let us now see how computation with commutative diagrams by means of diagram chasing may look like. Suppose we want to know if the numeric code for character ‘a’ is zero or not, *i.e.* we want to know to which other morphism from **1** to **Bool** the composite morphism  $a; ord; iszero$  is equal.

We take diagram 8.4(a) and start to paste commutative diagrams taken from figure 8.3. By chasing diagram 8.3(e) on  $ord$  we get diagram 8.4(b). We indicate by dotted lines those arrows on which we have just pasted a diagram. The solid-line arrows show us those arrows on which we still have to paste other diagrams. Next, by chasing diagram 8.3(a) on  $a$  and  $chr$  we get diagram 8.4(c).

**8.5.7 Remark.** Notice that in this step we are not chasing along arrows of a common path. This would seem a violation of definitions 8.3.6 and 8.3.7. It is not, because what we are actually doing here is an ‘unchasing’ of diagrams, *i.e.* a reverse process of diagram chasing. You can check that the chasings done in the reverse direction (from (d) to (a)) are all in accordance to definitions 8.3.6 and 8.3.7. Finally, by chasing diagram 8.3(c) on  $zero$  and  $iszero$  we get diagram 8.4(d), from which we can read that the numeric code of character ‘a’ is indeed zero.

**8.5.8 Caveat.** Of course, this example is in no manner a complete analysis of how sketches and diagram chasing may be applied to computation. A detailed

study remains to be done, since we only attempted to give a glimpse of the kind of questions that bother us, and in which directions we are trying to pursue them.

On the other hand, we think it is sufficiently illustrative to demonstrate the intuitive gain obtained by using a visual notation based on diagrams, and that this is no way contradictory with the use of a formal language. We believe that diagrammatic representations and the reasoning done with them, as it is exploited in category theory, can be very useful in order to design in the future an intuitive high-level visual declarative programming language that may help non-specialists build prototypes of preliminary specifications in a completely formal way.





Part V

Conclusion



## Chapter 9

# Conclusions and Future Work

This thesis is the result of exploring the relationship existing between several distinct research fields, by trying to bring them together upon one key aspect: the role of special relations in specification and reasoning.

Therefore we have tailored a logic in which the pragmatics of special relations is highlighted, much more, we believe, than in standard first-order logic, since we use technicalities borrowed from relation algebra, as for instance identity relations, relation composition, and relation reversion, to better cope with the role these relations play in specification and reasoning. We further endowed the logic with a very abstract semantics based on allegories, the category theory of relations, in order to have a framework of specification general enough to cover many distinct paradigms of specification based on special relations.

We have shown that the proposed logic indeed suites well as a framework logic for specification paradigms by defining conservative maps of logics from several examples of specification frameworks. Thus we were able to map membership equational logic, rewriting logic, non-deterministic specifications with set-relations, and also a diagrammatic Horn logic to our logic of special relations. These maps, for sure, emphasise much more the pragmatics of special relations, i.e., the key role played by them in all these specification frameworks.

But we have also shown how this key role played by special relations can also be exploited computationally by laying down a very general notion of term rewriting along binary relations. The technique presented in this thesis goes further than previous generalisations by allowing rewrite steps to be done along any relational expression within a fragment of relation algebra, and also taking into account very general monotonicity properties of the function symbols in the signature. We have shown in detail all the tractability problems that arise in such general perspective of term rewriting, but we also were able to identify those key properties signatures have to satisfy for effectively dealing with a proof calculus based on our general notion of term rewriting. For this reason,

we have established the important role played by the notion of polarity for taming the general framework, and introduced concepts like ‘well-polarised and well-commuting signature’ or ‘polarisation relation of a signature’, which shed light on how properties of special relations influence the tractability of the term rewriting technique.

Our general term rewriting technique applied to specific specification paradigms led us to unify two notions for the decidability of term rewriting techniques in membership equational specifications, which up to now were considered distinct. We have namely shown that confluence and sort-decreasingness actually are the same general notion of local confluence, pointing out that sort-decreasingness is actually too strong a condition for decidability. We also were able to revise the redundancy notions exploited by a proof calculus for non-deterministic specifications with set relations, and which is based on a generalisation of term rewriting similar, but more restricted, than the one presented in this thesis. By paying attention to the subtleties that such generalisation carries, we showed that the framework suffers of non-well-polarisation, which is one of the key conditions for practicable application of such general techniques.

Finally, inspired by the proof techniques presented in this thesis, we laid down completely diagrammatic inference rules that constitute the operational semantics of a novel visual programming language currently under development, and that attempts to make formal specification closer to non-logicians. Motivated by the success of our approach we have explored diagram reasoning within the more general framework provided by category theory, and have formalised one of its reasoning techniques —diagram chasing— within the context and perspective established by the diagrammatic reasoning community, in order to figure out how category theory may be useful for defining general methods of visual representation and reasoning.

## 9.1 Related Research

During the research done while working on this thesis, and because of the many different aspects involved in it, we came in touch with many other lines of research sharing similar objectives, methodology or inspirational points. But unfortunately, we have not devoted them the required time they deserved, in order to incorporate their ideas and approaches into our work. Now we would like to briefly comment them here.

**9.1.1 Partial order programming.** Focusing on the role special relations play in declarative programming, several researchers chose specific algebraic structures, and in particular lattices, for a variety of declarative programming languages. For example, Aït-Kaci and Podelski make use of order-sorted feature terms as basic data structure of the programming language *LIFE* (Aït-Kaci and Podelski, 1993), generalising in this way the flat first-order terms normally used as unique data structure in logic programming. Jayaraman, Osorio, and

Moon also base their partial order programming paradigm on a lattice structure, and are specially interested on the complete lattice of finite sets (Jayaraman et al., 1995). In their paradigm they pursue the aim to integrate sets into logic programming, and to consider them as basic data structure on which the paradigm relies. But Parker was probably the first who advocated programming on non-symmetric transitive relations like preorder or partial order relations for generalising and subsequently combining several different programming paradigms (Parker, 1987; Parker, 1989). Another approach for integrating functional and logic programming, based on rewriting logic, but taking possibly non-deterministic lazy functions as the fundamental notion, has been done by González-Moreno et al. (1996; 1997). We have done a preliminary exploration on the use of rewrite techniques for declarative programming with special relations in (Schorlemmer and Agustí, 1996).

**9.1.2 Relational programming and program construction.** Staying within the area of declarative programming, but moving to more abstract frameworks, we have found several attempts to integrate functional and logic programming by basing computation on the calculus of relations. Some directions are Broome and Lipton's combinatory logic programming (Broome and Lipton, 1994), which extends traditional database query formalisms by translating Horn clauses into relational expressions, and subsequently solving queries with a variable-free calculus based on equational term rewriting with the axioms of a relation algebra.

Motivated by the usefulness of a relational calculus to provide a natural treatment of partiality or non-determinism, and also by the opinion that the leading declarative programming paradigms, functional and logic programming, and the integration of both, lack of an elegant computational model which complicates the reasoning about the behaviour of functional logic programs, McPhee and de Moor propose to pursue a relational programming language which is purely compositional, and which is based on the theory of allegories (McPhee and de Moor, 1996). Furthermore Bird and de Moor have recently written an introductory textbook on the algebra of programming where they put special emphasis on using categorical methods from the theory of allegories for formal program construction (Bird and de Moor, 1997).

**9.1.3 Relational specifications.** From the algebraic specification point of view, theories in our logic of special relations can actually be seen as customised relational specification as put forth by Berghammer and Schmidt (1993). Their framework is very similar to ours from the model theoretic point of view, but it is more general, because it uses the whole expressiveness provided by relation algebra. The main difference relies in the proof theoretic aspect, since we were after providing rewriting-based proof calculi by approaching a general notion of term rewriting, while they focused mainly on machine-supported proof techniques for proving theorems in a relation algebra (Behnke et al., 1997).

**9.1.4 Tile logic.** Recently, we have discussed with Gaducci and Bruni the relationship existing between tile logic (Gaducci and Montanari, 1996) and the logic of special relations. It seems like a term rewriting system over a logic of special relations can be seen as a particular tile rewriting system, where our special relations are observations in that framework. Since there exists a conservative map of tile logic back to rewriting logic (Bruni et al., 1998), tile rewriting systems can be implemented in specification languages based on rewriting logic like *Maude* (Clavel et al., 1996), so that perhaps our general term rewrite technique can be eventually implemented in this language.

**9.1.5 Combining algebraic and set-theoretic specifications.** Rewriting set expression has been approached from an other quite distinct perspective by Kirchner and Mosses, who attempt to apply term rewriting techniques coming from membership equational specifications to a Z-like specification framework, thus bringing the model-oriented and property-oriented approach to specification together (Hintermeier et al., 1996; Kirchner and Mosses, 1999). Since we have not been aware of their work until very recently, we still do not know to which extend our framework and theirs may profit from mutual results.

## 9.2 Future Work

Beside exploring the relationship of our framework to all the research mentioned in the previous section, we also can further work on many aspects treated in this thesis.

**9.2.1 Logic of special relations.** We just tailored a logic to our needs, and it turned out to suite well for capturing several specification frameworks, and to highlight the role played by special relations. From the conclusion drawn in chapter 6, we firmly believe that this logic is a good starting point for further investigations of algebraic specification frameworks. But of course, one important aspect is still missing, namely that it is indeed complete with respect to our inference mechanism and to our proof method based on term rewriting with special relations. We know that, in convergent term rewriting systems, a decision algorithm based on rewriting can prove all sentences that the rewrite systems entails, but we do not know if it proves all sentences satisfied by its models. As we already mentioned in 3.3.13, we believe that completeness will be achieved, once we have established the relationship of our logic of special relations with Bruno, Gaducci, and Montanari's closely related tile logic (Gaducci and Montanari, 1996; Bruni et al., 1998).

**9.2.2 Term rewriting beyond equality.** Despite the discouraging situation laid down by the researchers that went before us in generalising term rewriting, namely that the development of term rewriting techniques and *Knuth-Bendix*-like completion procedures involving non-symmetric relations was not

worth any deeper investigation, because of its inherent difficulties for automation (Bachmair and Ganzinger, 1998; Struth, 1997), in this thesis we have shown that by using the notion of polarity we can tame the general inference mechanism, though the resulting conditions put on theories turn out to be very restrictive. Term rewriting captures many other special relations in addition to transitive and congruence ones, and this property of term rewriting needs to be further exploited.

Although maybe further studying general techniques of term rewriting, as presented in this thesis, is not worth the effort, it is necessary to think about more specific interesting theories with general special relations where our rewriting technique based on polarities for logics with special relations can be applied. Since by generalising term rewriting to binary relations we have been able, at least, to unify several distinct decidability conditions for specific theories in our logic with special relations, we are convinced that we will obtain more interesting results in the future by applying our rewriting techniques. For this reason, first, we will need to extend the framework to cover conditional term rewriting, in a similar way equational rewriting has been extended to the conditional case (Kaplan, 1984; Dershowitz and Okada, 1990; Ganzinger, 1991). Such extension might also be helpful to avoid the generation of critical atoms due to the overlap on variable positions as discussed in section 4.7.

**9.2.3 Specification and programming with special relations.** Specification paradigms are usually based on several special relations. While many-sorted specification has slowly shifted to treat sorts semantically, instead of syntactically, the interplay of special relations in their deduction systems has become more and more important. A unifying view at all levels, in their models and in their deduction systems is achieved in our logic with special relations.

We have also emphasised in section 9.1 that the gain in expressiveness using relations for specification and also for programming has been advocated by numerous researchers, and several proposals have been given. Specification paradigms and term-rewriting techniques, as studied in this thesis, may definitely be useful for further exploration of the use of relations for specification and programming. An interesting open problem would be to apply term-rewriting techniques to specific relation algebras, and to use term rewriting along binary relations to execute relational programs in a spirit similar to (Broome and Lip-ton, 1994).

**9.2.4 Diagrammatic reasoning and declarative programming.** Barwise and Hammer show the validity of diagrammatic reasoning as a completely formal deduction technique in logical systems. In particular, we have shown in this thesis how a particular novel visual language is just a diagrammatic logic with special relations that can be intuitively captured by visual cues, and how the diagrammatic reasoning done in this logic corresponds to specific chaining inferences.

In order to bring this observation a little bit further, and since models of our

logical theories are based on the category theoretic aspects of relation algebras, we attempted to formally define a fragment of diagrammatic reasoning in category theory. It would be of great interest to extend the formal analysis done on commutative diagrams and diagram chasing together with its relationship to standard equational reasoning, to semi-commutative diagrams and the diagrammatic Godement calculus of functors and natural transformations. It would capture a diagrammatic alternative to inequational reasoning as investigated in this thesis in its non-visual aspect.

Although the relationship between category theory and computer science has been of central importance for the formal specification discipline, a thorough investigation on how to use its diagrammatic tools for formal specification and high-level declarative programming remains to be done. Recent work in this direction, exploiting the category-theoretic notion of *sketch* for computation, includes Bagchi and Wells's thorough study on a *graph-based logic* constructed over the notion of sketch. They affirm it is "a first step towards a theory that is directly implementable for the purposes of computation" (Bagchi and Wells, 1997a; Bagchi and Wells, 1997b). Duval and Reynaud notice, also, that first-order structures are not well adapted to computation. They provide "a definition of sketches to deal explicitly with 'programs' (i.e. with operational semantics)" (Duval and Reynaud, 1994a; Duval and Reynaud, 1994b). We would like to further explore to which extent sketches *fit well* for the purpose of computation, by taking advantage of the experience of the diagrammatic reasoning community.



# Bibliography

- AGUSTÍ, J., PUIGSEGUR, J., AND ROBERTSON, D. 1998a. A visual syntax for logic and logic programming. *Journal of Visual Languages and Computation* 9:399–427.
- AGUSTÍ, J., PUIGSEGUR, J., ROBERTSON, D., AND SCHORLEMMER, W. M. 1996. Visual logic programming through set inclusion and chaining. In CADE-13 Workshop on Visual Reasoning, New Brunswick NJ, USA.
- AGUSTÍ, J., PUIGSEGUR, J., AND SCHORLEMMER, W. M. 1997. Towards specifying with inclusions. *Mathware & Soft Computing* IV:281–297.
- AGUSTÍ, J., PUIGSEGUR, J., AND SCHORLEMMER, W. M. 1998b. Query answering by means of diagram transformation. In Flexible Query Answering Systems, volume 1495 of *Lecture Notes in Artificial Intelligence*, pp. 15–28. Springer.
- AGUSTÍ, J., ROBERTSON, D., AND PUIGSEGUR, J. 1995. GRASP: A GRaphical SPecification language for the preliminary specification of logic programs. Research Report IIIA 95/13, Institut d’Investigació en Intel·ligència Artificial (CSIC).
- AÏT-KACI, H. AND PODELSKI, A. 1993. Towards a meaning of LIFE. *Journal of Logic Programming* 16:195–234.
- BAADER, F. AND NIPKOW, T. 1998. Term Rewriting and All That. Cambridge University Press.
- BACHMAIR, L. 1991. Canonical Equational Proofs. Birkhäuser.
- BACHMAIR, L., DERSHOWITZ, N., AND HSIANG, J. 1986. Orderings for equational proofs. In IEEE Symposium on Logic in Computer Science, pp. 346–357. IEEE Computer Society Press.
- BACHMAIR, L., DERSHOWITZ, N., AND PLAISTED, D. A. 1989. Completion without failure. In Resolution of Equations in Algebraic Structures, volume 2. Academic Press.
- BACHMAIR, L. AND GANZINGER, H. 1990. On restrictions of ordered paramodulation with simplification. In 10th International Conference on Automated Deduction, volume 449 of *Lecture Notes in Computer Science*, pp. 427–441. Springer.
- BACHMAIR, L. AND GANZINGER, H. 1994a. Ordered chaining for total orderings. In Automated Deduction — CADE’12, volume 814 of *Lecture Notes in Artificial Intelligence*, pp. 435–450. Springer.

- BACHMAIR, L. AND GANZINGER, H. 1994b. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation* 4:1–31.
- BACHMAIR, L. AND GANZINGER, H. 1994c. Rewrite techniques for transitive relations. In 9th IEEE Symposium on Logic in Computer Science, pp. 384–393. IEEE Computer Society Press.
- BACHMAIR, L. AND GANZINGER, H. 1994d. Rewrite techniques in theorem proving. CADE-12 Tutorial, Nancy, France.
- BACHMAIR, L. AND GANZINGER, H. 1998. Ordered chaining calculi for first-order theories of transitive relations. *Journal of the ACM* 45:1007–1049.
- BACHMAIR, L., GANZINGER, H., LYNCH, C., AND SNYDER, W. 1995. Basic paramodulation. *Information and Computation* 121:172–192.
- BAGCHI, A. AND WELLS, C. 1997a. Graph-based logic and sketches I: The general framework. Available from [www.cwru.edu/artsci/math/wells/pub/papers.html](http://www.cwru.edu/artsci/math/wells/pub/papers.html).
- BAGCHI, A. AND WELLS, C. 1997b. Graph-based logic and sketches II: Finite-product categories and equational logic. Available from [www.cwru.edu/artsci/math/wells/pub/papers.html](http://www.cwru.edu/artsci/math/wells/pub/papers.html).
- BARR, M. AND WELLS, C. 1995. Category Theory for Computing Science. Prentice-Hall.
- BARWISE, J. AND HAMMER, E. 1996. Diagrams and the concept of logical system, pp. 49–78. In *Logical Reasoning with Diagrams*, chapter III. Oxford University Press.
- BARWISE, J. AND SELIGMAN, J. 1997. Information Flow, volume 44 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.
- BEHNKE, R., BERGHAMMER, R., AND SCHNEIDER, P. 1997. Machine support of relational computations: The Kiel RELVIEW System. Technical Report 9711, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel.
- BERGHAMMER, R. AND SCHMIDT, G. 1993. Relational specifications. In *Proc. of the XXXVIII Banach Center Semester on Algebraic Methods in Logic and their Computer Science Applications*, volume 28 of *Banach Center Publications*, pp. 167–190.
- BERGSTRA, J. A. AND KLOP, J. W. 1986. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences* 32:323–362.
- BIDOIT, M., KREOWSKI, H.-J., LESCANNE, P., OREJAS, F., AND SANNELLA, D. 1991. Algebraic System Specification and Developement: A Survey and Annotated Bibliography, volume 501 of *Lecture Notes in Computer Science*. Springer.
- BIRD, R. AND DE MOOR, O. 1997. Algebra of Programming. Prentice-Hall.
- BLED SOE, W. AND HINES, L. 1980. Variable elimination and chaining in a resolution-based prover for inequalities. In 5th Conference on Automated Deduction, volume 87 of *Lecture Notes in Computer Science*, pp. 70–87. Springer.
- BLED SOE, W., KUNEN, K., AND SHOSTAK, R. 1985. Completeness results for

- inequality provers. *Artificial Intelligence* 27:255–288.
- BOROVANSKY, P., KIRCHNER, C., KIRCHNER, H., MOREAU, P., AND VITTEK, M. 1996. ELAN: A logical framework based on computational systems. In *Rewriting Logic and its Applications*, First International Workshop, volume 4 of *Electronic Notes in Theoretical Computer Science*, pp. 35–50. Elsevier Science.
- BOUHOULA, A., JOUANNAUD, J.-P., AND MESEGUER, J. 1997a. Specification and proof in membership equational logic. In *TAPSOFT'97*, volume 1214 of *Lecture Notes in Computer Science*, pp. 67–92. Springer.
- BOUHOULA, A., JOUANNAUD, J.-P., AND MESEGUER, J. 1997b. Specification and proof in membership equational logic. Unpublished draft.
- BRAND, D. 1975. Proving theorems with the modification method. *SIAM Journal of Computing* 4:412–430.
- BROOME, P. AND LIPTON, J. 1994. Combinatory logic programming: Computing in relation calculi. In *1994 International Logic Programming Symposium*. MIT Press.
- BRUNI, R., MESEGUER, J., AND MONTANARI, U. 1998. Internal strategies in a rewriting implementation of tile systems. In *Second International Workshop on Rewriting Logic and its Applications*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pp. 95–116. Elsevier Science.
- CLAVEL, M., EKER, S., LINCOLN, P., AND MESEGUER, J. 1996. Principles of Maude. In *Rewriting Logic and its Applications*, First International Workshop, volume 4 of *Electronic Notes in Theoretical Computer Science*, pp. 65–89. Elsevier Science.
- COMON, H. 1998. Completion of rewrite systems with membership constraints, Part I: Deduction rules. *Journal of Symbolic Computation* 25:397–419.
- DERSHOWITZ, N. 1987. Termination of rewriting. *Journal of Symbolic Computation* 3:69–116.
- DERSHOWITZ, N. AND JOUANNAUD, J.-P. 1990. Rewrite systems. In *Handbook of Theoretical Computer Science*, volume B. Elsevier Science.
- DERSHOWITZ, N., MARCUS, L., AND TARLECKI, A. 1988. Existence, uniqueness, and construction of rewrite systems. *SIAM Journal of Computing* 17:629–639.
- DERSHOWITZ, N. AND OKADA, M. 1990. A rationale for conditional equational programming. *Theoretical Computer Science* 75:111–138.
- DEVILLE, Y. 1990. *Logic Programming*. Addison-Wesley.
- DIACONESCU, R., FUTATSUGI, K., ISHISONE, M., NAKAGAWA, A. T., AND SAWADA, T. 1998. An overview of CafeOBJ. In *Second International Workshop on Rewriting Logic and its Applications*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pp. 75–88. Elsevier Science.
- DUVAL, D. AND REYNAUD, J.-C. 1994a. Sketches and computation — I: basic definitions and static evaluation. *Mathematical Structures in Computer Science* 4:185–238.
- DUVAL, D. AND REYNAUD, J.-C. 1994b. Sketches and computation — II: dynamic evaluation and applications. *Mathematical Structures in Computer*

- Science* 4:239–271.
- EHRESMANN, C. 1968. Esquisses et types des structure algébriques. *Bul. Inst. Polit. Iași* XIV.
- EHRIG, H. AND MAHR, B. 1985. Fundamentals of Algebraic Specifications 1: Equations and Initial Semantics. Springer.
- EKER, S. AND MESEGUER, J. 1997. Design of the equational proving tools for Cafe. Slides of a talk given at the CafeOBJ Workshop, Japan.
- FREESE, R., JEŽEK, J., AND NATION, J. 1993. Term rewrite systems for lattice theory. *Journal of Symbolic Computation* 16:279–288.
- FREYD, P. J. AND SCEDROV, A. 1990. Categories, Allegories. North-Holland.
- GADUCCI, F. AND MONTANARI, U. 1996. Tiles, rewriting rules, and CCS. In First International Workshop on Rewriting Logic and its Applications, volume 4 of *Electronic Notes in Theoretical Computer Science*, pp. 1–19. Elsevier Science.
- GANZINGER, H. 1989. Order-sorted completion: The many-sorted way. In TAPSOFT '89, volume 351 of *Lecture Notes in Computer Science*, pp. 244–258. Springer.
- GANZINGER, H. 1991. A completion procedure for conditional equations. *Journal of Symbolic Computation* 11:51–81.
- GANZINGER, H., NIEUWENHUIS, R., AND NIVELA, P. 1995. The Saturate system. Available from [www.mpi-sb.mpg.de/SATURATE/Saturate.html](http://www.mpi-sb.mpg.de/SATURATE/Saturate.html).
- GNAEDIG, I., KIRCHNER, C., AND KIRCHNER, H. 1988. Equational completion in order-sorted algebras. In CAAP '88, volume 299 of *Lecture Notes in Computer Science*, pp. 165–184. Springer.
- GOGUEN, J. AND BURSTALL, R. 1984. Introducing institutions. In Logics of Programms, volume 164 of *Lecture Notes in Computer Science*. Springer.
- GONZÁLEZ-MORENO, J. C., HORTALÁ-GONZÁLEZ, T., LÓPEZ-FRAGUAS, F., AND RODRÍGUEZ-ARTALEJO, M. 1996. A rewriting logic for declarative programming. In Programming Languages and Systems — ESOP '96, volume 1058 of *Lecture Notes in Computer Science*. Springer.
- GONZÁLEZ-MORENO, J. C., HORTALÁ-GONZÁLEZ, T., AND RODRÍGUEZ-ARTALEJO, M. 1997. A higher order rewriting logic for functional logic programming. In Logic Programming: The 14th International Conference. MIT Press.
- HAREL, D. 1988. On visual formalisms. *Communications of the ACM* 31:514–530.
- HINES, L. M. 1992. The central variable strategy of str+ve. In CADE-11, *Lecture Notes in Computer Science*, pp. 35–49.
- HINTERMEIER, C., KIRCHNER, H., AND MOSSES, P. D. 1996. Combining algebraic and set-theoretic specifications. In Recent Trends in Data Type Specification, volume 1130 of *Lecture Notes in Computer Science*, pp. 255–273. Springer.
- HSIANG, J. AND RUSINOWITCH, M. 1987. On word problems in equational theories. In Proc. 14th Int. Colloquium on Automata, Languages and Programming, volume 267 of *Lecture Notes in Computer Science*, pp. 54–71.

- Springer.
- HSIANG, J. AND RUSINOWITCH, M. 1991. Proving refutational completeness of theorem proving strategies: The transfinite semantic tree method. *Journal of the ACM* 38:559–587.
- HUET, G. 1980. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM* 27:797–821.
- HUET, G. 1981. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computation and System Sciences* 23:11–21.
- HUET, G. AND OPPEN, D. C. 1980. Equations and rewrite rules. A survey. Technical Report STAN-CS-80-785, Stanford Verification Group.
- HULLOT, J. M. 1980. Canonical forms and unification. In Proc. 4th International Conference on Automated Deduction, volume 87 of *Lecture Notes in Computer Science*.
- INVERARDI, P. 1995. Rewriting for preorder relations. In Conditional and Typed Rewriting Systems, 4th International Workshop, CTRS-94, volume 968 of *Lecture Notes in Computer Science*, pp. 223–234. Springer.
- JAYARAMAN, B., OSORIO, M., AND MOON, K. 1995. Partial order programming (revisited). In Algebraic Methodology and Software Technology, AMAST'95, volume 936 of *Lecture Notes in Computer Science*, pp. 561–575. Springer.
- JOUANNAUD, J.-P. AND KIRCHNER, H. 1986. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing* 15:1155–1194.
- KAPLAN, S. 1984. Conditional rewrite rules. *Theoretical Computer Science* 33:175–193.
- KIRCHNER, C., KIRCHNER, H., AND MESEGUER, J. 1988. Operational semantics of OBJ-3. In ICALP '88, volume 317 of *Lecture Notes in Computer Science*. Springer.
- KIRCHNER, C., KIRCHNER, H., AND VITTEK, M. 1995. Designing constraint logic programming languages using computational systems. In Principles and Practice of Constraint Programming. MIT Press.
- KIRCHNER, H. AND MOSSES, P. D. 1999. Algebraic specifications, higher-order types, and set-theoretic models. In Algebraic Methodology and Software Technology, AMAST'98, volume 1548 of *Lecture Notes in Computer Science*, pp. 373–388. Springer.
- KLOP, J. W. 1992. Term rewriting systems, pp. 1–116. In Handbook of Logic in Computer Science, volume 2. Oxford University Press.
- KNUTH, D. E. AND BENDIX, P. B. 1970. Simple word problems in universal algebras, pp. 263–297. In Computational Problems in Abstract Algebra. Pergamon Press.
- KRIAUCIUKAS, V. 1999. Email message to W. Marco Schorlemmer.
- KRIAUCIUKAS, V. AND WALICKI, M. 1995. Reasoning and rewriting with set-relations I: Ground completeness. In Computer Science Logic, 8th Workshop, CSL'94, volume 933 of *Lecture Notes in Computer Science*, pp. 264–278. Springer.
- KRIAUCIUKAS, V. AND WALICKI, M. 1996. Rewriting and reasoning with set-

- relations II: The non-ground case completeness. *In* Recent Trends in Data Type Specification, volume 1130 of *Lecture Notes in Computer Science*, pp. 306–321. Springer.
- LAMBEK, J. 1979. Subequalizers. *Canad. Math. Bull.* 13:337–349.
- LANKFORD, D. S. 1975. Canonical inference. Technical Report ATP-32, Department of Mathematics and Computer Science, University of Texas.
- LANKFORD, D. S. AND BALLANTYNE, A. 1977. Decision procedures for simple equational theories with permutative axioms: Complete sets of permutative reductions. Technical Report ATP-37, Department of Mathematics and Computer Science, University of Texas.
- LEVY, J. 1993. Second-order bi-rewriting systems. Research Report IIIA 93/23, Institut d'Investigació en Intel·ligència Artificial (CSIC).
- LEVY, J. 1995. The Calculus of Refinements: A Formal Specification Model Based on Inclusions. Number 2 in IIIA Monographs. Institut d'Investigació en Intel·ligència Artificial (CSIC).
- LEVY, J. AND AGUSTÍ, J. 1992. Implementing inequality and nondeterministic implementations with bi-rewriting systems. *In* Recent Trends in Data Type Specification, volume 758 of *Lecture Notes in Computer Science*, pp. 252–267. Springer.
- LEVY, J. AND AGUSTÍ, J. 1993. Bi-rewriting, a term rewriting technique for monotonic order relations. *In* Rewriting Techniques and Applications, volume 690 of *Lecture Notes in Computer Science*, pp. 17–31. Springer.
- LEVY, J. AND AGUSTÍ, J. 1996. Bi-rewrite systems. *Journal of Symbolic Computation* 22:279–314.
- LEVY, J. AND VILLARET, M. 1999. On the decidability of context unification. Technical report, Institut d'Investigació en Intel·ligència Artificial (CSIC).
- MADDUX, R. D. 1991. The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica* 50:421–455.
- MANCA, V., SALIBRA, A., AND SCOLLO, G. 1990. Equational type logic. *Theoretical Computer Science* 77:131–159.
- MANNA, Z. AND WALDINGER, R. 1986. Special relations in automated deduction. *Journal of the ACM* 33:1–59.
- MANNA, Z. AND WALDINGER, R. 1992. The special-relation rules are incomplete. *In* Automated Deduction – CADE-11, volume 607 of *Lecture Notes in Artificial Intelligence*, pp. 492–506. Springer.
- MARTÍ-OLIET, N. AND MESEGUER, J. 1993. Rewriting logic as logical and semantic framework. Technical Report SRI-CSL-93-05, Computer Science Laboratory, SRI International.
- MCALLESTER, D., GIVAN, B., AND FATIMA, T. 1989. Taxonomic syntax for first order inference. *In* Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning, pp. 289–300.
- MCPHEE, R. AND DE MOOR, O. 1996. Compositional logic programming. *In* JICSLP'96 post conference workshop on multi-paradigm logic programming, pp. 1–12.
- MÉGRELIS, A. 1992. Partial algebra + order-sorted algebra = galactic algebra.

- In Logical Foundations of Computer Science*, volume 620 of *Lecture Notes in Computer Science*, pp. 314–325. Springer.
- MESEGUER, J. 1989. General logics. *In Logic Colloquium '87*, pp. 275–329. Elsevier Science.
- MESEGUER, J. 1992. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science* 96:73–155.
- MESEGUER, J. 1998. Membership algebras as a logical framework for equational specification. *In Proc. of WADT'97*, volume 1376 of *Lecture Notes in Computer Science*, pp. 18–61. Springer.
- MITCHELL, J. C. AND SCEDROV, A. 1993. Notes on scoping and relators. *In Computer Science Logic '92, Selected Papers*, volume 702 of *Lecture Notes in Computer Science*, pp. 352–378. Springer.
- MOSSÉS, P. 1989. Unified algebras and institutions. *In Principles of Programming Languages Conference*, pp. 304–312. ACM Press.
- MOSSÉS, P. 1993. The use of sorts in algebraic specification. *In Recent Trends in Data Type Specification*, volume 655 of *Lecture Notes in Computer Science*, pp. 66–91. Springer.
- NIEUWENHUIS, R. AND RUBIO, A. 1995. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computation* 19:321–351.
- NIVELA, P. AND NIEUWENHUIS, R. 1993. Saturation of first-order (constrained) clauses with the Saturate system. *In Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pp. 436–440. Springer.
- PARKER, D. S. 1987. Partial order programming. Technical Report CSD-870067, UCLA Computer Science Department.
- PARKER, D. S. 1989. Partial order programming. *In POPL'89: 16th ACM Symposium on Principles of Programming Languages*, pp. 260–266. ACM Press.
- PETERSON, G. E. 1983. A technique for establishing completeness results in theorem proving with equality. *SIAM Journal of Computing* 12:82–100.
- PETERSON, G. E. AND STICKEL, M. E. 1981. Complete sets of reductions for some equational theories. *Journal of the ACM* 28:233–264.
- PLAISTED, D. A. 1993. Equational reasoning and term rewriting systems, pp. 273–364. *In Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1. Oxford University Press.
- PUIGSEGUR, J. 2000. Visual Declarative Programming by Diagrammatic Reasoning on Set Inclusions. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya. Forthcoming.
- PUIGSEGUR, J. AND AGUSTÍ, J. 1998. Visual declarative programming by means of diagram transformations. *In Proc. of the 1998 Joint Conference on Declarative Programming APPIA-GULP-PRODE'98*, pp. 311–328, A Coruña, Spain.
- PUIGSEGUR, J., AGUSTÍ, J., AND ROBERTSON, D. 1996. A visual logic programming language. *In 12th Annual IEEE Symposium on Visual Languages*. IEEE Computer Society Press.
- PUIGSEGUR, J., SCHORLEMMER, W. M., AND AGUSTÍ, J. 1997. From queries to answers in visual logic programming. *In 13th Annual IEEE Symposium*

- on Visual Languages. IEEE Computer Society Press.
- ROBERTSON, D. AND AGUSTÍ, J. 1999. Software Blueprints: Lightweight Uses of Logic in Conceptual Modelling. Addison-Wesley.
- ROBERTSON, D., AGUSTÍ, J., HESKETH, J., AND LEVY, J. 1994. Expressing program requirements using refinement lattices. *Fundamenta Informaticæ* 21:163–183.
- ROBINSON, G. AND WOS, L. 1969. Paramodulation and theorem-proving in first-order theories with equality, pp. 135–150. *In* Machine Intelligence 4. American Elsevier.
- SANNELLA, D. AND TARLECKI, A. 1997. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing* 9:229–269.
- SCHORLEMMER, W. M. 1996. Bi-rewriting rewriting logic. *In* First International Workshop on Rewriting Logic and its Applications, volume 4 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science.
- SCHORLEMMER, W. M. 1998. Rewriting logic as a logic of special relations. *In* Second International Workshop on Rewriting Logic and its Applications, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science.
- SCHORLEMMER, W. M. 1999. Term rewriting logic in a logic of special relations. *In* Algebraic Methodology and Software Technology, AMAST'98, volume 1548 of *Lecture Notes in Computer Science*, pp. 178–195. Springer.
- SCHORLEMMER, W. M. AND AGUSTÍ, J. 1995. Theorem proving with transitive relations from a practical point of view. Research Report IIIA 95/12, Institut d'Investigació en Intel·ligència Artificial (CSIC).
- SCHORLEMMER, W. M. AND AGUSTÍ, J. 1996. Inclusional theories in declarative programming. *In* Proc. of the 1996 Joint Conference on Declarative Programming APPIA-GULP-PRODE'96, pp. 167–178, Donostia, Spain.
- SCHORLEMMER, W. M., AGUSTÍ, J., AND PUIGSEGUR, J. 1998. On reasoning in category theory by diagram chasing. *In* LICS'98 Workshop on Logic and Diagrammatic Information, Indianapolis IN, USA.
- SHIMOJIMA, A. 1996. Operational constraints in diagrammatic reasoning, pp. 27–48. *In* Logical Reasoning with Diagrams, chapter II. Oxford University Press.
- SLAGLE, J. R. 1972. Automated theorem proving for theories with built-in theories including equality, partial orderings and sets. *Journal of the ACM* 19:120–135.
- STRUTH, G. 1997. On the word problem for free lattices. *In* Rewriting Techniques and Applications, volume 1232 of *Lecture Notes in Computer Science*, pp. 128–141. Springer.
- WADGE, W. W. 1982. Classified algebras. Research Report 46, Department of Computer Science, University of Warwick.
- WIERINGA, R. J. 1996. Requirements Engineering. John Wiley & Sons.
- WIRSING, M. 1990. Algebraic specification. *In* Handbook of Theoretical Computer Science, volume B, chapter 13. North-Holland.