MONOGRAFIES DE L'INSTITUT D'INVESTIGACIÓ
EN INTEL·LIGÈNCIA ARTIFICIAL
Number 2

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*

The Calculus of Refinements, a Formal Specification Model Based on Inclusions

Jordi Levy Díaz

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

Volume Author
Jordi Levy Díaz
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Institut d'Investigació
en Intel·ligència Artificial

ISBN: 84–00–07500–5
ISSN: 1135–4100
Dep. Legal: B–34741–95

Printed by Cardellach Còpies, – CBS, S.A.
Sant Pere, 40. 08221 Terrassa.

Als meus pares

Contents

Foreword

The research presented in this book started with a very practical aim: to investigate methods of incremental description. One approach to this has been the design of specification languages with good structuring operations. The approach taken in this book comes from a simple idea: to consider types as approximate values which can be specified by means of the order relation of its information content, an inclusion relation. To investigate in depth this idea, a formal model is proposed. The model can be seen, on one hand, from the point of view of type theory where the value-type relation has been substituted by an inclusion relation between types or approximate values. On the other hand, it can be seen as a higher order specification language where conventional equational specifications have been generalized to inclusion specifications.

The strength of the book lies in the creative use of existing techniques, extending them when necessary. In the domain of denotational semantics, a new model of λ-calculus has been found and a new rewriting technique has been introduced to give operational semantics to inclusion specifications. The attentive reader of this book I am sure will enjoy and learn from this long and intensive research effort, going from a simple idea to its accurate realization in a formal model.

Bellaterra, February 1996

Jaume Agustí
Head of the
Formal Methods Department
of the IIIA, CSIC

Preface

Moltes persones han contribuit d'una o altre manera a la realització d'aquesta tesi, i de ben segur que m'en oblidaré d'alguna. En primer lloc, vull agrair a en Jaume Agustí, el director d'aquesta tesi, el recolzament que m'ha donat durant aquests cinc anys i l'haver-se volgut embarcar en aquesta dificultosa aventura. A en Pere García, en Lluís Godo, en Ramon López de Mántaras, n'Enric Plaza, en Josep Puyol, en Carles Sierra i a tots els meus companys del Institut d'Investigació en Intel·ligència Artificial la seva col·laboració, suggeriments i ajuda. I en especial, a en Francesc Esteva l'haver tingut la paciència infinita de revisar moltes de les demostracions que aquí es presenten.

Tampoc voldria oblidar-me de donar gràcies a en Gabriel Ciobano, n'Hubert Comon, en Harald Ganzinger, en Claudio Hermida, l'Helene Kirchner, en Pierre Lescanne, en Karl Meinke, en José Meseguer, en Peter Mosses, en Roberto Nieuwenhuis, en Tobias Nipkow, en Fernando Orejas, en Mario Rodríguez Artalejo, en Klaus Schulz, en Gert Smolka, n'Andrzej Tarlecki i a totes les persones amb les que he tingut l'oportunitat de discutir algun aspecte d'aquesta tesi pels seus suggeriments i profitoses idees. Agraeixo especialment a la Jane Hesketh, en Dave Robertson, en Don Sannella i a tots els investigadors del *Department of Artificial Intelligence* i del *Department of Computer Science* de la Universitat d'Edinburgh la calurosa acollida amb que em varen rebre durant les meves estades.

La CICYT, la Generalitat de Catalunya i el CSIC han subvencionat part d'aquesta investigació mitjançant un contracte com a Titular Superior a càrrec del projecte SPES financiat per la CICYT (TIC 880j380), una Beca de Formació d'Investigadors de la Generalitat de Catalunya (DOGC 1638 de 28-8-1992) i una Beca para Ingenieros, Arquitectos y Licenciados en Informática del CSIC (BOE 14-11-1992). Esperem que ho continuin fent.

També vull donar gracies a tots els meus amics del Centre d'Estudis Avançats de Blanes i a la vila de Blanes per haver-me acollit durant la realització d'aquesta tesi. Molt especialment a en Felip Manyà, en Gabriel Valiente i en Lluís Vila per la seva amistat i tots els bons moments i sopars que hem pogut compartir, i dels que espero continuem gaudint.

Resto en deute amb la Clara Barroso per tot el que m'ha ensenyat i per la seva ajuda en els moments més difícils i importants d'aquesta tesi.

Tanmateix dono gracies a la Xus Grau per tot el que m'ha cuidat, mimat i ajudat al llarg de la nostra prolongada amistat, especialment en els moments de desesperació i flaquesa.

Finalment, als meus pares els he d'agrair tot, ja que sense ells res de tot això hauria estat possible.

Abstract

Programming in the large require the use of formal specification languages for describing program requirements and a method to test (automatically) such requirements. These methods can also be applied in other areas like complex system modeling. In this thesis we study the theoretical kernel of a formal specification language, named *Calculus of Refinements (COR)*, based on the use of monotonic inclusion relations. These relations are more general than equality relations, therefore inclusion specifications can be considered as a generalization of equational specifications. Moreover, we propose the substitution of the typing relation ":" by an inclusion relation, therefore, the Calculus of Refinements can also be considered as a new typing discipline. The theoretical study of the Calculus of Refinements consists of the definition of a denotational semantics and of an operational semantics for it. They are described on the two first parts of the thesis. In the third part we approach the specification of nondeterministic programs by means of inclusions.

In the first part of the thesis we describe the Calculus of refinements as a logic, giving its syntax, a set of inference rules and defining a class of models based on the class of environment models of the λ-calculus. We also study a concrete model where expressions are interpreted as order ideals. Such ideal domains have been used to give semantics to polymorphic types. On it we base the view of the Calculus of Refinements as a typing discipline.

In the second part we give an operational semantics based on rewrite techniques. We define a pair of rewriting systems, namely a *bi-rewriting system*, which implement the deduction on inclusion theories. The main idea is using one of the relations to rewrite terms into smaller terms, and the other one to rewrite terms into bigger terms. Using a bi-rewriting system is possible to implement an algorithm to test if an inclusion $a \subseteq b$ is deducible in a theory. We rewrite a into bigger terms, and b into smaller terms till we obtain a common term. We have studied such technique for first-order theories and linear second-order theories (where bindings bind one and only one variable occurrence).

In the third part, we propose the use of bi-rewriting systems for the verification of nondeterministic program specifications. We model nondeterministic computation by means of a relation satisfying, among others, the inclusion axioms. Therefore, the rewriting technique is sound (although not necessarily complete). We prove that adding more axioms to the specification such technique is also complete.

Chapter 1

Introduction

Development of large-scale programs require the use of a formal language for describing program requirements and a method to test (automatically) such requirements. After some early attempts to specify requirements for imperative programs (based on state pre- and post-conditions) it became clear that specification languages are closely related with programming languages, and both must have a clean and simple mathematical semantics. Nowadays, most of the specification languages are developed following an algebraic approach. Basic programming techniques, like *modularity* and *parametrization*, have also been applied to specification methodologies.

One of the most successful specification methodologies is the *stepwise refinement* discipline. It proposes the incremental program construction by gradually transforming an original specification till we obtain such a low-level specification that it can be considered as a program. We obtain in such way a sequence of specifications leading from the original specification to the final program. The correctness of the resulting program is ensured provided that each transformation step is correct. We say that a transformation step $SP \rightsquigarrow SP'$ is correct (namely, SP *refines* to SP', or SP' *implements* SP) if every model of SP' is a model of SP. The development of a program consist of a sequence of refinement steps $SP_0 \rightsquigarrow SP_1 \rightsquigarrow \cdots \rightsquigarrow SP_n$. The refinement relation $\rightsquigarrow$ must be transitive (*vertical composition*) and monotonic (*horizontal composition*). Vertical composability ensures the correctness of the final program SP_n w.r.t. the original specification SP_0 if every refinement step $SP_i \rightsquigarrow SP_{i+1}$ is correct. Horizontal composability guarantees that if we have a parameterized specification $P(SP)$ then any refinement P' of P and any refinement SP' of SP results on a refinement $P'(SP')$ of $P(SP)$.

This thesis describes a formal model for the development of a specification methodology within the stepwise refinement and algebraic specification disciplines. Algebraic specifications are mainly based on the use of equality relations. We generalize drastically such approach by introducing a more general kind of relations: *inclusion relations*. These relations share the same properties than an equality relation with the exception of the symmetry property. Their use is motivated by their expressiveness. Furthermore, we also propose the use of inclusion relations as a new typing relation (substituting each typing declaration $t : \tau$ by an inclusion axiom $t \subseteq \tau$). The practical consequences of these decisions are out of the scope of this thesis and are the matter of current and future research work. However, the future practical use of the specification model has guided us in all the technical decisions in its definition.

Our main objective, in the thesis, has been to develop the formal basis of this methodology. In order to achieve such objective we have defined the *Calculus of Refinements* (COR). We describe its *denotational semantics*, on which it has to be based the future specification language, and its *operational semantics*, which sets the techniques for the automation of the deduction in COR. These constitute the two main parts of the thesis. The Calculus of Refinements can also be seen as a first approach to a *computing with sets of values* model. In such computation model the user starts defining how functions work for big sets of values (for instance $\texttt{product(int, int)} \subseteq \texttt{int}$ for the product function), refines this specification defining how it works for smaller sets of values ($\texttt{product(even, odd)} \subseteq \texttt{even}$) and, finally, defines how it works for singletons ($\texttt{product}(2,3) = 6$). This methodology is closely related with the non-deterministic computation and specification paradigm. In the third part of the thesis we present a first relation between both paradigms.

1.1 A Specification Language Based on Inclusions

Our approach can be analyzed from the point of view of the following two paradigms:

- Algebraic specifications
- Type theories

Algebraic specifications make an intensive use of equality relations, although there have been proposed some extensions based on the use of other relations: inclusion relations (Mosses, 1989b; Mosses, 1989c), membership relations (Manca et al., 1990; Comon, 1993), transitive relations (Bachmair and Ganzinger, 1993b), rewrite relations (Meseguer, 1990; Meseguer, 1992). Our approach is based on the use of monotonic inclusion relations. Unlike Bachmair and Ganzinger, our transitive relations are monotonic, and in relation to Mosses we give an operational semantics to these relations. In our case this is based on bi-rewriting systems, a rewriting technique used to check when a term is included into another term.

Algebraic specifications also use to be based on many-sorted signatures, i.e. terms are built using a set $\mathcal{F}$ of function symbols and each symbol $f \in \mathcal{F}$ has an assigned type $f : s_1 \times \cdots \times s_n \to s$ where $s_1, \ldots, s_n, s$ are sorts (basic types). If we want terms to be higher-order then such types may be more complex including, at least, all those type expressions generated by the grammar $\tau ::= b \mid \tau \to \tau$ where

b stands for any base type. Many extensions to this simple type theory there have also been proposed, i.e. polymorphic types (Milner, 1978; Cardelli and Wegner, 1985; MacQueen et al., 1986), *Calculus of Constructions* (Coquand and Huet, 1988),...). They consist mainly on enlarging the set of type expressions being considered (and therefore, the set of typing rules). Most of them require any well-formed term t to have a *unique* type τ and the existence of an algorithm to find such type for any given term. On the contrary, we propose not to distinguish between types and values and substitute the typing relation ":" between values and types by an inclusion relation. Our approach can be seen as a language with a unique universe of types and a subtyping relation, where the typing relation $t : \tau$ is interpreted as a special case of subtyping relation $\{t\} \subseteq \tau$. Thus, the bi-rewriting method, implementing the inclusion relation, is also a kind of type checker.

1.2 Motivation

Our research work started five years ago with the practical study of some specifications languages such as Nuprl (Constable et al., 1986), LCF (Paulson, 1987), Z (Spivey, 1988) and, specially, Standard ML (Sannella and Tarlecki, 1984; Harper, 1986; Milner et al., 1990) and Extended ML (Sannella and Tarlecki, 1991a). Simultaneously, we also begun to study the application of specification techniques in areas other than program requirement specification, such as the specification of ecological models.[1] From this experience we concluded that all these languages share a often unnecessarily complex higher-order type theory, specially if we use them in knowledge representation applications (Robertson et al., 1993).

The following pair of examples have been included to show how program requirements can be captured with the use of inclusion relations. They are not developed in detail and must be considered only as a justification for the use of such relations. Let us consider the following toy example in Standard ML, consisting of a signature specification and a program.

```
signature SIG_TOY =
  sig
    type s
    val x : s
  end
```

$\rightsquigarrow$

```
structure STRUCT_TOY : SIG_TOY =
  struct
    type s = int
    val x = 3
  end
```

In COR we propose do not distinguish between types and values, and between signatures and programs. Thus, we can consider such example as a program refinement. In the first version we declare s as a type, i.e. as something smaller than top ($\top$), and x as something smaller than s. In the second version be assign concrete values to s and x. The same example in COR would be as follows.

[1] We started a collaboration with Dr. Sannella of the Laboratory for Foundation of Computer Science and Prof. Robertson of the Department of Artificial Intelligence, both at the University of Edinburgh.

$$\boxed{\begin{array}{l} s \subseteq \top \\ x \subseteq s \end{array}} \quad \rightsquigarrow \quad \boxed{\begin{array}{l} s = int \\ x = 3 \end{array}}$$

If we take into account that `3 : int`, i.e. $3 \subseteq int$ expressed as an inclusion, then we can prove that the models of the second specification are also models of the first one. Notice that whereas in Standard ML we have type and value declarations, in COR we have a unique kind of axioms (inclusions and equalities).

Let us consider another example in a more theoretical framework, the *Calculus of Constructions* (Coquand and Huet, 1988). This calculus uses three universes of expressions: contexts (Δ), types (P) and terms (t), being *contexts* a kind of "types of types". Judgments are built using a context (Γ), like in the sequent calculus, and may be of two forms: 1) *typing judgments*, built using a typing relation ":" between terms and types ($t : P$) or between types and contexts ($P : \Delta$); or 2) *conversion judgments* built using a conversion relation "$\cong$" between types ($P_1 \cong P_2$) or between terms ($t_1 \cong t_2$). There are also two kinds of variable bindings: products $[x : P_1]P_2$ and λ-abstractions $(\lambda x : P)t$. The following are the typing rule for λ-abstractions, assigning a product to each abstraction, and the conversion rules for products and for abstractions.

$$\frac{\Gamma[x : P_1] \vdash t : P_2}{\Gamma \vdash (\lambda x : P_1)t \ : \ [x : P_1]P_2}$$

$$\frac{\begin{array}{c}\Gamma \vdash P_1 \cong P_2 \\ \Gamma[x : P_1] \vdash P_3 \cong P_4\end{array}}{\Gamma \vdash [x : P_1]P_3 \cong [x : P_2]P_4} \qquad \frac{\begin{array}{c}\Gamma \vdash P_1 \cong P_2 \\ \Gamma[x : P_1] \vdash t_1 \cong t_2 \\ \Gamma[x : P_1] \vdash t_1 \ : \ P_3\end{array}}{\Gamma \vdash (\lambda x : P_1)t_1 \cong (\lambda x : P_2)t_2}$$

We can reformulate the Calculus of Constructions applying the principle of no distinction between types and values (in this case between contexts, types an values). No distinction is made then between λ-abstractions and products, the typing relation ":"is substituted by the inclusion relation $\subseteq$, and the conversion relation $\cong$ by the equality relation $=$. The later is interpreted as a pair of inclusions, i.e. $t = u \Leftrightarrow t \subseteq u \wedge u \subseteq t$. With such reformulation, all the above rules are subsumed by the following one:

$$\frac{\begin{array}{c}\Gamma \vdash t_1 = t_2 \\ \Gamma, x \subseteq t_1 \vdash t_3 \subseteq t_4\end{array}}{\Gamma \vdash \lambda x : t_1 \,.\, t_3 \subseteq \lambda x : t_2 \,.\, t_4}$$

that tells us when a λ-abstraction is included into another one. With this reformulation we obtain a higher-order type theory sharing some properties with the Calculus of Construction, in particular, if $\Gamma \vdash_{CC} t : P$ is provable in the Calculus of Constructions, then $\Gamma' \vdash_{COR} t' \subseteq P'$ is provable in the reformulated calculus, where Γ', t' and P' are respectively the reformulation of Γ, t and P. Similarly, $\Gamma \vdash_{CC} t_1 \cong t_2$ implies $\Gamma' \vdash_{COR} t'_1 = t'_2$. But as the three universes of terms are mixed, other properties of the Calculus of Constructions are lost, in particular, we lose the following *unique type* property:

If $\Gamma \vdash_{CC} t : P_1$ *and* $\Gamma \vdash_{CC} t : P_2$ *then* $\Gamma \vdash_{CC} P_1 \cong P_2$

(Coquand and Huet, 1988; lemma 5).

As the reader may assume, the many-sorted equational logic also admits a translation into such *inclusion formalism.*

We are aware that many important problems have been obviated in these examples; however our aim has been showing the expressive power of inclusion relations and the adequacy of using them in stepwise refinement disciplines. Another specification approach which also shares such conviction is the *Unified Algebras* (Mosses, 1989b; Mosses, 1989c; Mosses, 1989a). Thus we think that the advantage of using inclusion relations in program specifications, as well as in other kind of system specification do not require further discussion, from the point of view of simplicity, expressiveness and orthogonality of the resulting language.

1.3 Objectives and Contributions

Our main objective is to define and formalize a higher-order theory of monotonic inclusion relations. This objective has to consider the following three subobjectives:

1. define a syntax, a formal theory and a denotational semantics for the higher-order inclusion logic.
2. study the proof theory and automate the deduction for such logic.
3. find and propose application areas for such logic.

Although in this thesis we focus our attention on the first two objectives, we have also worked on some applications of the formalism (see chapter 6 and (Levy et al., 1992c; Robertson et al., 1993)).

Our principal contributions can be summarized as an answer to the following two questions:

1. To give denotational semantics to a higher-order logic of inclusions it seems sensible to interpret terms as sets of values and the inclusion relation as the inclusion relation of the set theory (as is usually done in type theories). Then, since our approach is higher-order, how can we interpret a function as a set of values?
2. The operational semantics of equational logic is usually based on rewriting techniques, i.e. on replacing terms by equal terms. If our principal relation is an inclusion relation, then can similar techniques be applied?

To answer the first question we define an ideal domain $\mathcal{J}(U)$ —that is, a subset of the power set of U— and an interpretation function

$$\mathsf{Fun} : \big(\mathcal{J}(U) \to \mathcal{J}(U)\big) \to \mathcal{J}(U)$$

(see subsections 3.3.3 and 3.3.4), which we used to map functions on such domain to elements of the domain, i.e. to interpret functions on sets as a sets of values.

To automate the deduction in the monotonic inclusion logic we use a pair of rewriting relations $\xrightarrow[R_{\subseteq}]{}$ and $\xrightarrow[R_{\supseteq}]{}$, one to replace subterms by bigger terms and the other to replace subterms by smaller terms. A sound and complete proof procedure is defined based on such pair of rewriting systems (named *bi-rewriting* system).

1.4 Related Work

This thesis makes use of very different techniques, therefore the related work come from very different areas, and we have decided to introduce them separately in each chapter. In this section we will only present the principal references we have used.

As usual, we use λ-calculus (i.e. the subject of (Barendregt, 1981) and (Hindley and Seldin, 1986)) as the skeleton of functional programming languages (Landin, 1964). It has also been used as a higher order parameterization mechanism in some specification languages (Sannella and Tarlecki, 1991b). Furthermore, we also view λ-calculus as a low-level specification formalism. The refinement relation we define can be compared with the subtype relation in (Reynolds, 1985; Cardelli, 1988), with the containment relation between types in (Mitchell, 1988) and in general with several typing systems (Martin-Löf, 1979; Constable et al., 1986; Coquand and Huet, 1988; Lampson and Burstall, 1988). In most of them (Martin-Löf, 1979; Coquand and Huet, 1988), however, value expressions and type expressions are rigorously distinguished and a type relation between values and types is formalized. Less rigorous is this distinction in Nuprl (Constable et al., 1986) and in Pebble (Lampson and Burstall, 1988). On the contrary, we drop such distinction from the very beginning and the type membership relation is replaced by a particular case of subtype relation. *Unified Algebras* (Mosses, 1989b; Mosses, 1989c), *Type Logic* (Manca et al., 1990) and *Rewrite Logic* (Meseguer, 1990; Meseguer, 1992) share the same principle.

The definition of the class of models of COR, in chapter 2, is based on the characterization of λ-calculus environment models of (Meyer, 1982; Koymans, 1982). There, we prove that the P_ω model of the λ-calculus (Scott, 1976) is also a model of COR

The use of order ideals as a semantic domain for types, as we propose in chapter 3, is not new and is also motivated in (Milner, 1978; MacQueen et al., 1986; Mitchell, 1988). The solution of the value recursive domain equation, in the definition of the ideal model, is based on the category-theoretic solution of such kind of equation formalized in (Smyth and Plotkin, 1982), other works also summarize such results (Plotkin, 1983; Schmidt, 1988; Pierce, 1991). The technique for constructing solutions of this kind of equations based on universal domains was originally presented in (Scott, 1976), and is summarized in (Gunter and Scott, 1990). Other papers describing the inverse limit construction without using category concepts are (Scott, 1972; Stoy, 1978).

The operational semantics of our language, described in chapters 4 and 5, is based on rewriting techniques. There is a huge number of papers on rewriting techniques. They can be found good introductions to the subject in (Knuth and Bendix, 1970; Huet, 1980; Klop, 1987; Dershowitz and Jouannaud, 1990). We extend our technique for rewriting on equivalence classes and for rewriting using a higher-order language. The techniques for rewriting on equivalence classes are described in (Huet, 1980; Peterson and Stickel, 1981; Kirchner, 1985a; Kirchner, 1985b; Jouannaud and Kirchner, 1986; Bachmair and Dershowitz, 1989a). Higher-order rewriting systems are defined in (Nipkow, 1991; Nipkow, 1992; Nipkow, 1993) and are based on a higher-order unification algorithm described in (Miller, 1991a; Miller, 1991b). An introduction to unification theory can be found in (Siekmann, 1989; Gallier and Snyder, 1990). In chapter 5 we define an unification procedure for our higher-order language. Other higher-order unification procedures are described in (Huet, 1975; Jensen and Pietrzykowski, 1976). The decidability of the unification problem we present remains as an open question. Related unification problems are the undecidable (Goldfarb, 1981) second-order unification problem and the decidable (Makanin, 1977) string unification problem. Comon (Comon, 1993) defines a language based on expression schemes quite similar to ours.

The nondeterministic specification approach we use in chapter 6 is introduced in (Kaplan, 1986a; Kaplan, 1988; Hussmann, 1991; Hussmann, 1992).

1.5 Overview of the Thesis

The thesis is organized in three parts:

Part I (chapters 2 and 3) introduces the Calculus of Refinements as a logic and defines its *denotational semantics.*

Part II (chapters 4 and 5) gives an *operational semantics* of the calculus.

Part III (chapter 6) shows a first *application* of the calculus.

The contents of each chapter are summarized in the following.

Chapter 2. We describe the Calculus of Refinements (COR) as a logic, giving its syntax and inference rules. The main contribution of the chapter is a characterization of COR-models. It is based on the characterization of λ-calculus models done by Meyer (Meyer, 1982) in terms of environment models and functional domains. We prove a soundness and completeness theorem for the COR-theories w.r.t. the COR-environment models. Finally, we show that two concrete classical λ-calculus models —the P_ω model of Scott (Scott, 1976) and the D_∞ model— are also COR-models. Any COR-domain has a lattice structure which we use for giving semantics to the inclusion (subtyping) relation $\subseteq$. The P_ω and D_∞ models have also such lattice structure, however in such cases the lattice order relation is also used for modeling the *computational ordering* (Scott, 1972; Scott, 1976). Therefore, in those models *computational* and *refinement* orderings are identified. This could cause problems and leads us to define a new concrete COR-model. This is the subject of chapter 3.

Chapter 3 Types are usually modeled by sets of values sharing a common structure, where *structure* represents notions like being a function or being a pair. These notions are modeled by a partial order $\prec$ defined by $a \prec b$ if *a is more structured than b*. Then, types are *ideals* of such ordering (Milner, 1978; MacQueen et al., 1986). The *structural* ordering $\prec$ and the *computational* ordering $\sqsubseteq$ are related. In this chapter we define the relation $\prec$ by $a \prec b$ iff $b \sqsubseteq a$. An order ideal model may be defined using such structural ordering. Then we interpret the refinement relation as the inclusion relation $\subseteq$ between ideals. We proof that if built the ideal domain over a functional domain of values, then the resulting ideal domain is also functional (a retract of its corresponding functional space). It means that the type domain is rich enough for modeling λ-calculus and the use of two distinct universes (types and values) is no longer necessary.

Chapter 4 The use of an inclusion relation between terms makes necessary to extend rewriting techniques to deal with non-symmetric relations. In this chapter we propose how deduction in an inclusion theory I may be automated using a pair of rewriting systems $\langle R_{\subseteq}, R_{\supseteq} \rangle$ (a *bi-rewriting system*). To prove that $I \vdash a \subseteq b$ we seek a common expression c such that $a \xrightarrow[R_{\subseteq}]{*} c$ and $b \xrightarrow[R_{\supseteq}]{*} c$. The decidability and completeness of the method is based on the commutation and termination of both rewriting relations. We extend such result for bi-rewriting modulo a set of inclusions and we show some concrete examples.

Chapter 5 The method proposed in chapter 4 is not free from problems. The use of the Knuth-Bendix completion process (Knuth and Bendix, 1970), in the presence of non-left linear rules may introduce rule schemes (resulting from orienting schemes of critical pairs, named *extended critical pairs*). In this chapter we propose the use of *linear second-order typed λ-calculus* to incorporate those rule schemes into the language. We show the adequacy of using such restricted second-order language as a new higher-order rewriting paradigm. A complete unification procedure is described for such language. However, the decidability of this unification problem remains as an open question, for which we think we will be able to prove an affirmative answer in the future.

Chapter 6 This is the only chapter devoted to applications of the Calculus of Refinements. We view non-deterministic specifications (Kaplan, 1986a; Hussmann, 1992) as a special case of inclusion specifications. Therefore, the bi-rewriting technique is a sound although not always complete deduction method for those specifications. We show how a non-deterministic specification may be completed (without modifying the underlieing intended models) in order to have also a completeness result.

Although we have tried to write a basically self-contained document, the great variety of techniques used to present this formal model makes impossible to be completely exhaustive in such purpose. Therefore, some previous basic knowledge on category theory and rewrite techniques is supposed. Anyway, we cite some basic literature on such subjects. What we call *examples* in the first part of the thesis are really alternative definitions or unsuccessful lines of argument. Examples shown in the second part have not been checked by a computer, thus they may be not free from errors.

Even though we use different theoretical formalisms along the thesis, we have tried to use an uniform notation. This is always introduced at the beginning of each chapter and can also be found in the index at the end of the thesis.

Chapter 2

The Class of Models

Abstract: The Calculus of Refinements (COR) presented here takes the idea of types as specifications and subtyping as refinement and pushes it to an extreme. Types and values are no longer distinguished; in COR we consider a unique hierarchy of objects. A good way to deal with this hierarchy of objects is to structure it as a complete lattice. If functions are to be considered as first class citizens in the hierarchy, then the lattice must be reflexive: it must have the space of functions (some of them) as a sublattice. To represent reflexive lattices, the most simple language is an extension of the λ-calculus with lattice operators: this is the language of COR. The aim of this chapter is to show that the results about the soundness and the completeness of λ-calculus w.r.t. the model of terms (Meyer, 1982; Koymans, 1982) can be extended without problems to COR. We show also that two classical λ-models, the P_ω and the D_∞ models, are also COR-models.

2.1 Introduction

Data and its classification into types are kept separated and used distinctively in most programming languages. Types are mainly used as a discipline that contributes to program correctness and computation is not done on types. Nevertheless types have also been considered as specifications in (Martin-Löf, 1979; Constable et al., 1986; Coquand and Huet, 1988; Lampson and Burstall, 1988; Cardelli and Longo, 1990). The subtyping can be seen as a kind of specification refinement defining a type hierarchy where programs are the leaves on which computation is done. The Calculus of Refinements (COR) presented here takes this idea of types as specifications and subtyping as refinement and pushes it to an extreme. Types and values are no longer distinguished; in COR we consider a unique hierarchy of objects without distinctions between leaves and the rest of nodes. The subtyping relation is the only relation between objects and it is called refinement: an object is a refinement of another if the latter is a more specified version of the former. A good way to deal with the hierarchy of objects is to structure it as a complete lattice. And if functions are to be considered elements of the hierarchy then the lattice must be reflexive: there must be an injection from a subset of the space of functions to the lattice. So complete reflexive lattices are the intended semantic domains we want to use to model specifications or types (upper objects in the hierarchy) and programs or values (lower objects in the hierarchy).

To represent reflexive lattices, the most simple language is an extension of the λ-calculus with lattice operators, this is the language of COR. The results about soundness and completeness of λ-calculus w.r.t. to the *term model* can be extended without problems to COR. Moreover, we show that many concrete models of the λ-calculus are also models of COR. Unfortunately, the partial order defining the lattice structure of such models represents the *computational order* between λ-terms. This computational order (introduced by Scott (Scott, 1972)) justifies, for instance, the use of continuous functions (w.r.t. the topology defined by such order) and can be interpreted in terms of information: $a \sqsubseteq b$ if a is *more undefined* (contents less information) than b. On the contrary, the *subtyping order* is usually interpreted in terms of containment: $a \subseteq b$ if the set of values represented by a is included into the set of values represented by b. Therefore, the identification of both orderings may leads to problems. We define in chapter 3 a concrete COR-model where such problems are considered.

This chapter proceeds as follows. In section 2.2 the syntax of COR is introduced, COR-terms and COR-formulas are defined and then the axioms and inference rules defining COR-theories are given. Section 2.3 describes the class of models of COR. The main result of the chapter is the extension of the soundness and completeness results of λ-calculus to COR, proved in section 2.4. Section 2.5 presents some concrete models of the λ-calculus, which are also models of COR.

2.2 COR-syntax and COR-theories

In the following we will be concerned with a denumerable signature $\mathcal{F}$ of constants and an infinity and denumerable set of variables $\mathcal{X}$. The set of COR-**terms** is defined inductively by the following grammar

$$\begin{aligned}\mathtt{term} \; ::= \;& c \mid x \mid \bot \mid \top \mid \mathtt{term} \cap \mathtt{term} \mid \mathtt{term} \cup \mathtt{term} \\ \mid \;& \lambda x.\mathtt{term} \mid \mathtt{term}(\mathtt{term})\end{aligned}$$

where $x \in \mathcal{X}$ stands for variables and $c \in \mathcal{F}$ for constants.

COR-**formulas** are defined by

$$\mathtt{formula} ::= \mathtt{term} \subseteq \mathtt{term} \mid \mathtt{term} = \mathtt{term}$$

Definition 2.1 *A* **COR-theory** $\mathcal{T}$ *over a set* $\mathcal{F}$ *of constants is a set of formulas closed under the following* **COR-inference rules**.

$$\frac{}{\bot \subseteq t}\ (\bot) \qquad \frac{}{t \subseteq \top}\ (\top)$$

$$\frac{}{t \subseteq t \cup u} \qquad \frac{}{u \subseteq t \cup u} \qquad \frac{t \subseteq v \quad u \subseteq v}{t \cup u \subseteq v}\ (\cup)$$

$$\frac{}{t \cap u \subseteq t} \qquad \frac{}{t \cap u \subseteq u} \qquad \frac{v \subseteq t \quad v \subseteq u}{v \subseteq t \cap u}\ (\cap)$$

$$\frac{}{t \subseteq t}\ (Reflex) \qquad \frac{t \subseteq u \quad u \subseteq v}{t \subseteq v}\ (Trans)$$

$$\frac{t \subseteq u \quad u \subseteq t}{t = u} \qquad \frac{t = u}{t \subseteq u} \qquad \frac{t = u}{u \subseteq t}\ (Antisym)$$

$$\frac{\{y \notin \mathcal{FV}(t)\}}{\lambda x.t = \lambda y.t[y/x]}\ (\alpha) \qquad \frac{}{(\lambda x.t)(u) = t[u/x]}\ (\beta)$$

$$\frac{t \subseteq u}{t(v) \subseteq u(v)}\ (Apl\ monot) \qquad \frac{t = u}{v(t) = v(u)}\ (Apl\ congr)$$

$$\frac{t \subseteq u}{\lambda x.t \subseteq \lambda x.u}\ (\lambda\ monot)$$

It is quasi-extensional if in addition it is closed under:

$$\frac{}{(\lambda x.t) \cap (\lambda x.u) \subseteq \lambda x.t \cap u}\ (\lambda\cap) \qquad \frac{}{(t \cup u)(v) \subseteq t(v) \cup u(v)}\ (Apl\cup)$$

$$\frac{\{x \notin \mathcal{FV}(t)\}}{t \subseteq \lambda x.t(x)}\ (\eta^*)$$

and extensional if it is also closed under:

$$\frac{}{\lambda x.t \cup u \subseteq (\lambda x.t) \cup (\lambda x.u)}\ (\lambda\cup) \qquad \frac{}{t(v) \cap u(v) \subseteq (t \cap u)(v)}\ (Apl\cap)$$

$$\frac{\{x \notin \mathcal{FV}(t)\}}{t = \lambda x.t(x)}\ (\eta)$$

Given a finite set of COR*-formulas* I*, the sets* $Th(I)$*,* $Th^{\mathrm{q-e}}(I)$ *and* $Th^{\mathrm{e}}(I)$ *denote respectively the minimum* COR*-theory, quasi-extensional* COR*-theory and extensional* COR*-theory containing* I*.*

The notation $I \vdash_{\mathrm{COR}} u \subseteq v$ means that the formula $u \subseteq v$ belongs to the theory $Th(I)$, i.e. there exists a deduction of $u \subseteq v$ from I. The same for $I \vdash_{\mathrm{COR^{q-e}}} u \subseteq v$ and $Th^{\mathrm{q-e}}(I)$ and for $I \vdash_{\mathrm{COR^{e}}} u \subseteq v$ and $Th^{\mathrm{e}}(I)$. We say that $u \subseteq v$ is a COR-theorem, noted $\vdash_{\mathrm{COR}} u \subseteq v$, if the formula $u \subseteq v$ belongs to any COR-theory.

The relation between terms u and v, defined by $u \sim v$ if $u = v$ belongs to a given theory, is an equivalence relation. The set of equivalence classes will be a key point used to prove the completeness theorem.

Antisymmetry rules prove the equivalence $u = v \dashv\vdash u \subseteq v \wedge v \subseteq u$, therefore, from now on we will consider $u = v$ as an abbreviation for $u \subseteq v \ \wedge \ v \subseteq u$.

It is not hard to prove from monotonicity rules that simultaneous substitution preserves formulas, i. e., if $\mathcal{T} \vdash u_i = v_i$ for every $i \in \{1, \ldots, n\}$ and $\mathcal{T} \vdash u_0 \subseteq v_0$, then $\mathcal{T} \vdash u_0[u_1/x_1, \ldots, u_n/x_n] \subseteq v_0[v_1/x_1, \ldots, v_n/x_n]$.

2.3 COR-models

In this section we define the class of COR-models, like the one defined in (Meyer, 1982; Koymans, 1982) for the λ-calculus. In section 2.4 we will prove the completeness and correctness of this class of models in relation to COR-theories.

Definition 2.2 *A* **value model** *of COR is a poset $(D, \leq)$ whose elements are named values, and a function $[\![\_]\!]_{\_}$ from* COR*-terms and valuation functions ρ to D such that if $[\![u_1]\!]_\rho \leq [\![v_1]\!]_\rho, \ldots, [\![u_n]\!]_\rho \leq [\![v_n]\!]_\rho$ then $[\![u_0]\!]_\rho \leq [\![v_0]\!]_\rho$ for every* COR*-inference rule $u_1 \subseteq v_1, \ldots, u_n \subseteq v_n \vdash u_0 \subseteq v_0$ and valuation ρ.*

Value models are a mere reformulation of COR-theories, it is an essentially syntactic notion which don't justify or explain the COR-inference rules. To capture the notion of COR-terms as descriptions of functions in a lattice, we need to introduce the notion of functional domain on which is based our definition of COR-models.

Definition 2.3 *Let D be a nonempty set. Given two functions*

$$\begin{array}{lcccc} \mathsf{Fun} & : & D & \longrightarrow & (D \to D) \\ \mathsf{Graph} & : & \mathsf{Fun}(D) & \longrightarrow & D \end{array}$$

the triplet $E = (D, \mathsf{Fun}, \mathsf{Graph})$ is said to be a **functional domain** *if $f = \mathsf{Fun}(\mathsf{Graph}(f))$ for every $f \in \mathsf{Fun}(D) \subseteq D \to D$, that is, $\mathsf{Fun}(D)$ is a retract of D via the retraction pair $(\mathsf{Fun}, \mathsf{Graph})$.*
It is extensional if $u = \mathsf{Graph}(\mathsf{Fun}(u))$ for every $u \in D$.

Notice that since Fun maps D onto $D \to D$, it follows from cardinality considerations that $D \to D \not\subseteq \mathsf{Fun}(D)$, the mapping Graph is not defined for any function. This characterization of D is enough for the λ-calculus, but in the COR-calculus we need an additional condition to enable the definition of the lattice operators $\top$, $\bot$, $\cup$ and $\cap$, and the inclusion relation $\subseteq$.

Definition 2.4 *The tuple $E = (D, \leq_D, \mathsf{Fun}, \mathsf{Graph})$ is said to be a* **COR-domain** *if*

(i) *$(D, \mathsf{Fun}, \mathsf{Graph})$ is a functional domain.*
(ii) *$(D, \leq_D)$ is a complete lattice with maximum $\top_D$ and minimum $\bot_D$.*
(iii) *If we use $\leq_D$ to define a pointwise ordering[1] in $D \to D$ then* Fun *and* Graph *are monotonous functions.*

[1]The pointwise ordering is defined by $f \leq_{D \to D} g$ if $\forall x \in D \,.\, f(x) \leq_D g(x)$.

It is **quasi-extensional** *if in addition* $u \leq_D \mathsf{Graph}(\mathsf{Fun}(u))$ *for every* $u \in D$*, and* **extensional** *if* $u = \mathsf{Graph}(\mathsf{Fun}(u))$.

Notice that if we use $\leq_D$ to define a pointwise ordering $\leq_{D\to D}$ in $D \to D$, then $(D \to D, \leq_{D\to D})$ is also a complete lattice with

$$\begin{array}{l}(f \sqcap_{D\to D} g)(u) = f(u) \sqcap_D g(u)\\ (f \sqcup_{D\to D} g)(u) = f(u) \sqcup_D g(u)\\ \bot_{D\to D}(u) = \bot_D\\ \top_{D\to D}(u) = \top_D\end{array}$$

for any $f, g \in D \to D$ and $u \in D$.

The same classification of models is given in (Sanchis, 1980), with little differences, but completely different purposes.

Lemma 2.5 *For any* $u, v \in D$ *and* $f, g \in D \to D$ *we have*

(i) *If* $(D, \mathsf{Fun}, \mathsf{Graph})$ *is quasi-extensional then*

$$\begin{array}{rcl}\mathsf{Fun}(u \sqcup_D v) & = & \mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)\\ \mathsf{Graph}(u \sqcap_{D\to D} v) & = & \mathsf{Graph}(u) \sqcap_D \mathsf{Graph}(v)\end{array}$$

(ii) *If* $(D, \mathsf{Fun}, \mathsf{Graph})$ *is extensional then*

$$\begin{array}{rcl}\mathsf{Fun}(u \sqcap_D v) & = & \mathsf{Fun}(u) \sqcap_{D\to D} \mathsf{Fun}(v)\\ \mathsf{Graph}(u \sqcup_{D\to D} v) & = & \mathsf{Graph}(u) \sqcup_D \mathsf{Graph}(v)\end{array}$$

Proof: Since Fun is monotonous we have $\mathsf{Fun}(u) \leq_D \mathsf{Fun}(u \sqcup_D v)$ and $\mathsf{Fun}(v) \leq_D \mathsf{Fun}(u \sqcup_D v)$, therefore $\mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v) \leq_D \mathsf{Fun}(u \sqcup_D v)$.

On the opposite direction, the monotonicity of Graph and the quasi-extensionality allow to prove $u \leq_D \mathsf{Graph}(\mathsf{Fun}(u)) \leq_D \mathsf{Graph}\big(\mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)\big)$, the same for v and therefore $u \sqcup_D v \leq_D \mathsf{Graph}\big(\mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)\big)$. Now, the monotonicity of Fun and the condition of being a functional domain allow to prove $\mathsf{Fun}(u \sqcup_D v) \leq_{D\to D} \mathsf{Fun}\big(\mathsf{Graph}\big(\mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)\big)\big) = \mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)$.

Concluding $\mathsf{Fun}(u \sqcup_D v) = \mathsf{Fun}(u) \sqcup_{D\to D} \mathsf{Fun}(v)$. For the rest of equalities the proof is quite similar. ■

This lemma concludes that if (η^*) is sound in a COR-domain, then $(\lambda\cap)$ and $(Apl\cup)$ are also sound; and if (η) is sound then $(\lambda\cup)$ and $(Apl\cap)$ are also sound.

In the context of λ-calculus, the assignment of values to free variables are referred to as *environments*. Formally, an environment ρ over D is a map of the set of variables $\mathcal{X}$ into D. The set of environments over D will be noted by D_Env. Given a COR-domain, the corresponding COR-model is defined as follows.

Definition 2.6 *Given a (quasi-extensional, extensional)* COR*-domain* $(D, \leq_D, \mathsf{Fun}, \mathsf{Graph})$ *and the valuation function*

$$\mathcal{V} : \text{COR-}terms \times D\_Env \to D$$

defined inductively as follows

(i) $\mathcal{V}[x]_\rho = \rho(x)$ *for all variables* $x \in \mathcal{X}$
(ii) $\mathcal{V}[\top]_\rho = \top_D$
(iii) $\mathcal{V}[\bot]_\rho = \bot_D$
(iv) $\mathcal{V}[u \cup v]_\rho = \mathcal{V}[u]_\rho \sqcup_D \mathcal{V}[v]_\rho$
(v) $\mathcal{V}[u \cap v]_\rho = \mathcal{V}[u]_\rho \sqcap_D \mathcal{V}[v]_\rho$
(vi) $\mathcal{V}[u(v)]_\rho = \mathsf{Fun}(\mathcal{V}[u]_\rho)(\mathcal{V}[v]_\rho)$
(vii) $\mathcal{V}[\lambda x.u]_\rho = \mathsf{Graph}(\lambda d : D \,.\, \mathcal{V}[u]_{\rho[d/x]})$

if the function $\lambda d : D \,.\, \mathcal{V}[u]_{\rho[d/x]}$ *belongs to* $\mathsf{Fun}(D)$ *for any term* u *and environment* ρ*, then the tuple* $M = (D, \leq, \mathsf{Fun}, \mathsf{Graph}, \mathcal{V})$ *is said to be a (quasi-extensional, extensional)* COR-**environment model**.

The condition $\lambda d : D \,.\, \mathcal{V}[u]_{\rho[d/x]} \in \mathsf{Fun}(D)$ in the previous definition is necessary to ensure that $\mathsf{Graph}(\lambda d : D \,.\, \mathcal{V}[u]_{\rho[d/x]})$ is defined (notice that the domain of Graph is the rang of Fun). This restriction defines some kind of closure condition on the set of functions $\mathsf{Fun}(D)$ used in functional domains. In concrete models, presented in section 2.5, $\mathsf{Fun}(D)$ is the set of functions on D continuous w.r.t. the computational ordering, i.e. a subset of the computable functions (Scott, 1972).

The notions of satisfaction and validity are defined as usual. A formula $u \subseteq v$ is said to be **satisfied** in a COR-model M, noted $M \models u \subseteq v$, if $\mathcal{V}[u]_\rho \leq \mathcal{V}[v]_\rho$ for every environment $\rho \in D\_Env$. A formula is said to be **valid**, noted $\models u \subseteq v$ if it is satisfied for every COR-model.

2.4 Soundness and Completeness

In this section we prove a soundness and completeness theorem for the COR-theories w.r.t. the COR-environment models we have defined. We will prove that the inclusions valid in a COR-model form a COR-theory, and that for any COR-theory we can find a COR-environment model, namely the model of terms.

We can prove a substitution lemma, like in λ-calculus:

Lemma 2.7 $\mathcal{V}[u[v/x]]_\rho = \mathcal{V}[u]_{\rho[\mathcal{V}[v]_\rho/x]}$.

Proof: By induction on the structure of COR-terms. A complete proof can be found for the λ-calculus in (Stoy, 1978), and ours is quite similar. ■

Theorem 2.8 soundness. *The equations valid in a (quasi-extensional, extensional)* COR-*model form a (quasi-extensional, extensional)* COR-*theory. Moreover, if* $u \subseteq v$ *is a theorem* ($\vdash u \subseteq v$) *then* $u \subseteq v$ *is valid* ($\models u \subseteq v$).

Proof: The complete lattice structure of D corresponds to axioms and rules ($\bot$), ($\top$), ($\cap$), ($\cup$), (*Reflex*), (*Antisim*) and (*Trans*). Axioms (β) and (α) can be proved from the substitution lemma (for a complete proof see (Meyer, 1982)).

The Graph monotonicity and pointwise order in $D \to D$ can be used to prove (λ *monot*), and the Fun monotonicity and pointwise order to prove (*Apl monot*).

For a quasi-extensional model, using the definition of $\mathcal{V}$ it is easy to see

$$\begin{aligned}\mathcal{V}[(u \cup v)(w)]_\rho &= \mathsf{Fun}(\mathcal{V}[u]_\rho \sqcup_D \mathcal{V}[v]_\rho)(\mathcal{V}[w]_\rho)\\ \mathcal{V}[(u(w) \cup v(w)]_\rho &= \mathsf{Fun}(\mathcal{V}[u]_\rho)(\mathcal{V}[w]_\rho) \sqcup_D \mathsf{Fun}(\mathcal{V}[v]_\rho)(\mathcal{V}[w]_\rho)\end{aligned}$$

Now, the properties $\mathsf{Fun}(a \sqcup_D b) = \mathsf{Fun}(a) \sqcup_{D\to D} \mathsf{Fun}(b)$ of lemma 2.5 and $(f \sqcup_{D\to D} g)(a) = f(a) \sqcup_D g(a)$ can be used to prove the equality between both values, that is, the $(Apl\cup)$ axiom. Axiom $(\lambda\cap)$ can be proved from properties $\mathsf{Graph}(f \sqcap_{D\to D} g) = \mathsf{Graph}(f) \sqcap_D \mathsf{Graph}(g)$ (lemma 2.5) and $(f \sqcap_{D\to D} g)(a) = f(a) \sqcap_D g(a)$ in a similar way.

Using the definition of $\mathcal{V}$ we can see also $\mathcal{V}[\lambda x.u(x)]_\rho = \mathsf{Graph}\big(\lambda d : D.\, \mathsf{Fun}(\mathcal{V}[u]_\rho)(d)\big) = \mathsf{Graph}(\mathsf{Fun}(\mathcal{V}[u]_\rho))$ if x is not free in u. If the domain is quasi-extensional then $\mathcal{V}[u]_\rho \subseteq \mathsf{Fun}(\mathsf{Graph}(\mathcal{V}[u]_\rho))$ holds, and (η^*) can be proved.

For an extensional model $(Apl\cap)$, $(\lambda\cup)$ and (η) can be proved similarly to $(Apl\cup)$, $(\lambda\cap)$ and (η^*).

Finally, if $\vdash u \subseteq v$ is a theorem, then for any COR-theory, $\mathcal{T} \vdash u \subseteq v$ holds. In particular for the COR-theory formed by the satisfiable formulas of a COR-model M. Therefore $M \models u \subseteq v$ for any COR-model M. Thus $\models u \subseteq v$ is valid. ■

In the following we will prove the completeness theorem, showing that given a COR-theory $\mathcal{T}$ it can be constructed a COR-model $M_\mathcal{T}$ which valid equations are the ones deducible from the COR-theory. Such model is named **term model**.

Given a COR-theory $\mathcal{T}$, we note the set of $\mathcal{T}$-equivalence classes of terms by $\Im$ and the $\mathcal{T}$-equivalence class of u by $\overline{u}$. The relation $\leq_\Im \subseteq \Im \times \Im$ and the functions $\mathsf{Fun}_\Im : \Im \to (\Im \to \Im)$ and $\mathsf{Graph}_\Im : (\Im \to \Im) \to \Im$ defined by

$$\begin{array}{rcl}\overline{u} \leq_\Im \overline{v} & \text{iff} & \mathcal{T} \vdash u \subseteq v\\ \mathsf{Fun}_\Im(\overline{u}) & \stackrel{def}{=} & \lambda\overline{v} : \Im\,.\,\overline{u(v)}\\ \mathsf{Graph}_\Im(\mathsf{Fun}(\overline{u})) & \stackrel{def}{=} & \overline{\lambda x.u(x)}\ \text{ for } x \notin \mathcal{FV}(u)\end{array}$$

can be used to define a COR-domain.

Lemma 2.9 *The tuple* $(\Im, \leq_\Im, \mathsf{Fun}_\Im, \mathsf{Graph}_\Im)$ *is a* COR-*domain.*

Proof: It is easy to prove that $\mathsf{Fun}_\Im$ and $\mathsf{Graph}_\Im$ are well defined and that $\mathsf{Fun}_\Im \circ \mathsf{Graph}_\Im = Id$ holds. Therefore $(\Im, \mathsf{Fun}_\Im, \mathsf{Graph}_\Im)$ is a functional domain.

From the lattice inference rules $(\bot)$, $(\top)$, $(\cap)$ and $(\cup)$ it can be proved the lattice structure of $\Im$, where

$$\begin{array}{rcl}\bot_\Im & = & \overline{\bot}\\ \top_\Im & = & \overline{\top}\\ \overline{x} \sqcap_\Im \overline{y} & = & \overline{x \sqcap y}\\ \overline{x} \sqcup_\Im \overline{y} & = & \overline{x \sqcup y}\end{array}$$

The monotonicity of $\mathsf{Fun}_\Im$ and $\mathsf{Graph}_\Im$ can be proved from $(Apl\ monot)$ and $(\lambda\ monot)$. ■

Lemma 2.10 *The valuation function $\mathcal{V} : \text{COR-terms} \times \Im\_Env \to \Im$ corresponding to the COR-domain $(\Im, \leq_\Im, \mathsf{Fun}_\Im, \mathsf{Graph}_\Im)$ is defined by*

$$\mathcal{V}[u]_\rho = \overline{u[\rho(x_i)/x_i]}$$

where $\{x_i\}$ is the set of free variables of u.
Moreover, it satisfies $\lambda d : \Im \,.\, \mathcal{V}[u]_{\rho[d/x]} \in \mathsf{Fun}_\Im(\Im)$ for any term u and valuation ρ.

From this lemma we can prove the following completeness theorem:

Theorem 2.11 completeness *For every COR-theory $\mathcal{T}$, there exists a COR-environment model $\mathcal{M}_\mathcal{T}$ such that*

$$\mathcal{T} \vdash u \subseteq v \ \textit{ iff } \ \mathcal{M}_\mathcal{T} \models u \subseteq v$$

Moreover, if $u \subseteq v$ is valid ($\models u \subseteq v$), then $u \subseteq v$ is a theorem ($\vdash u \subseteq v$).

Proof: Given a COR-theory $\mathcal{T}$, let $\mathcal{M}_\mathcal{T}$ be the COR-model defined in lemmas 2.9 and 2.10. Suppose $\mathcal{T} \vdash u \subseteq v$ then $\mathcal{T} \vdash u_\rho \subseteq v_\rho$ for every assignment function ρ since simultaneous substitutions preserves equation. Therefore $\mathcal{V}[u]_\rho \subseteq \mathcal{V}[v]_\rho$ for every ρ, thus $\mathcal{M}_\mathcal{T} \models u \subseteq v$.

Suppose $\mathcal{M}_\mathcal{T} \models u \subseteq v$, then $\mathcal{V}[u]_\rho \subseteq \mathcal{V}[v]_\rho$ for every ρ. In particular for ρ_0 such that $\rho_0(x) = \overline{x}$. Then $\overline{u} = \mathcal{V}[u]_{\rho_0} \subseteq \mathcal{V}[v]_{\rho_0} = \overline{v}$. That is $\mathcal{T} \vdash u \subseteq v$.

Finally, if $\models u \subseteq v$ is valid, then for all COR-model $\mathcal{M}$, $\mathcal{M} \models u \subseteq v$. In particular for the COR-model $\mathcal{M}_\mathcal{T}$ obtained from any COR-theory $\mathcal{T}$, therefore $\mathcal{T} \vdash u \subseteq v$ for every theory. Thus $\vdash u \subseteq v$. ■

These two theorems prove the equivalence between $\models u \subseteq v$ and $\vdash u \subseteq v$, that is the weak soundness and completeness properties. To prove the (strong) soundness and completeness properties, i.e. the equivalence between $\Gamma \models u \subseteq v$ and $\Gamma \vdash u \subseteq v$, we would need the initiality property for the term model.

2.5 Concrete COR-models

Some of the models proposed classically for the λ-calculus have a lattice structure. The order relation $\subseteq$ defining this structure is based on computation considerations, i.e. $t \subseteq u$ if u is *more defined* than t, therefore $\perp$ is the *more undefined* term (that is, the program that never finishes). Such models take $\mathsf{Fun}(D) = [D \to D]$, being $[D \to D]$ the set of continuous functions from D to D. (Continuous functions are defined using the topology generated by the order relation $\subseteq$). These continuous functions include computable functions.

In this section we show how the P_ω and the D_∞ models of the λ-calculus are also models of the COR-calculus. However, we are interested in modeling a kind of *subtyping relation* with $\subseteq$, incompatible with the computation ordering. Because of that, we will develop a new model for COR (and therefore for the λ-calculus) in chapter 3. There we propose a relationship between both orderings.

2.5.1 The Model P_ω

In the P_ω model of the λ-calculus Scott (Scott, 1976) takes the set P_ω as functional domain and the functions $\mathsf{Fun} : P_\omega \longrightarrow [P_\omega \to P_\omega]$ and $\mathsf{Graph} : [P_\omega \to P_\omega] \longrightarrow P_\omega$, named `fun` and `graph` respectively, defined as follows

$$\begin{array}{lcl} \mathsf{Fun}(u) & \overset{def}{=} & \lambda v : P_\omega.\{m \mid \exists e_n \subseteq v.\langle n, m\rangle \in u\} \\ \mathsf{Graph}(f) & \overset{def}{=} & \{\langle n, m\rangle \mid m \in f(e_n)\} \end{array}$$

where $\langle \_, \_\rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a pair encoding function; $\{e_n\}_{n\in\mathbb{N}}$ is an enumeration of the finite subsets of P_ω; and $[P_\omega \to P_\omega]$ is the set of continuous functions from P_ω to P_ω.

The lattice structure of P_ω is evident, with $\perp$, $\top$, $\leq$, $\sqcup$ and $\sqcap$ defined as $\emptyset$, $\mathbb{N}$, $\subseteq$, $\cup$ and $\cap$ respectively. We have in addition, using the continuity of f

$$\mathsf{Fun}(\mathsf{Graph}(f)) = \lambda u : P_\omega.\{m \mid \exists e_n \subseteq u.\langle n, m\rangle \in f(e_n)\} = \lambda u : P_\omega.f(u) = f$$

and

$$\begin{array}{ll} \mathsf{Graph}(\mathsf{Fun}(u)) & = \{\langle n, m\rangle \mid m \in \{m' \mid \exists e_{n'} \subseteq e_n.\langle n', m'\rangle \in u\}\} = \\ & = \{\langle n, m\rangle \mid \exists e_{n'} \subseteq e_n.\langle n', m\rangle \in u\} \supseteq u \end{array}$$

The tuple $(P_\omega, \subseteq, \mathsf{Fun}, \mathsf{Graph})$ is a quasi-extensional COR-domain that allows to define a quasi-extensional COR-model defining $\mathcal{V}$ in the usual way. It has to be checked that $f = \lambda u : P_\omega \,.\, \mathcal{V}[u]_{\rho[u/x]}$ is a continuous function for any COR-term u. This proof can be done by induction in the structure of u.

If we restrict the domain to those elements u of P_ω such that

$$\langle n', m\rangle \in u \;\wedge\; e_{n'} \subseteq e_n \Rightarrow \langle n, m\rangle \in u$$

then $\mathsf{Fun}(\mathsf{Graph}(u)) = u$ and P_ω becomes an extensional COR-model.

2.5.2 The Model D_∞

This model is built solving the recursive domain equation

$$D \;\cong\; [D \to D]$$

in the category of lattices and continuous morphisms between lattices.[2] The solution is a lattice $(D_\infty, \leq_{D_\infty})$ and an isomorphism function

$$\Theta : D_\infty \to [D_\infty \to D_\infty]$$

The functions Fun and Graph are defined by $\mathsf{Fun} = \Theta$ and $\mathsf{Graph} = \Theta^{-1}$, and satisfy $\mathsf{Fun} \circ \mathsf{Graph} = Id_{D_\infty \to D_\infty}$ and $\mathsf{Graph} \circ \mathsf{Fun} = Id_{D_\infty}$. Therefore, the domain $(D_\infty, \leq_{D_\infty}, \Theta, \Theta^{-1})$ is an extensional model of COR.

[2] In section 3.2 we show how a similar equation can be solved in a given category.

2.6 Conclusions

A main point is worth noticing as a conclusion: the easiness with which environment models for λ-calculus have been extended to COR. COR-models are a subcategory of λ-calculus models keeping its main properties. Notice that, in principle, in these models functions need not to be continuous nor monotonous w.r.t. the subtyping order $\leq$. Nevertheless the concrete models presented take continuous functions as morphisms. This fact is related with the identification of the computational order and the subtyping order. That will be treated in detail in subsection 3.2.6.

Chapter 3

An Ideal Model for COR

Abstract: Types are usually modeled by sets of values sharing a common structure, where *structure* represents notions like being a function or being a pair. These structural notions are modeled using a partial order relation $\prec$ between terms, defined intuitively by $a \prec b$ if a is *more structured* than b. Then, types are interpreted as *order ideals* of such ordering (Milner, 1978; MacQueen et al., 1986). This *structural* ordering $\prec$ and the *computational* ordering $\sqsubseteq$, introduced by Scott (Scott, 1972; Scott, 1976), are related. In this chapter we identify the relation $\prec$ with the inverse of the relation $\sqsubseteq$, and we define an *ideal model* based on such structural ordering. We interpret the refinement relation between types as the set inclusion relation $\subseteq$ between ideals. We proof that if the ideal domain is built over a functional domain of values, then the resulting ideal domain is also functional (a retract of its corresponding functional space). It means that the type domain is rich enough for modeling λ-calculus and the use of two distinct universes of types and values is superfluous.

3.1 Introduction

We interpret λ-calculus as a typing formalism which admits a precise notion of refinement or inclusion between λ-expressions. By type we mean here an intensional description of a set of values sharing some structure, where *structure* represents notions like being a function or being a pair. These notions are modeled by a *structural order* $\prec$ defined intuitively by $t \prec t'$ if t is *more structured* than t'. For instance, we say that completely undefined value $\bot$ is less structured than any other value $\forall x \,.\, x \prec \bot$, or that $\langle t_1, t_2\rangle$ is more structured than $\langle t_1', t_2'\rangle$ if t_1 is more structured than t_1' and t_2 more structured than t_2'. Lambda expressions can be considered types so far we make them denote not elements of a domain, as usual, but a particular set of them. Then an expression can be refined by giving another expression whose denotation is included in the denotation of the former, i.e. we model the refinement relation between expressions as the inclusion relation between sets.

Having in mind the interpretation of λ-expressions as types, we have taken *closed $\prec$-order ideals* (Milner, 1978; MacQueen et al., 1986) as denotations of λ-expressions. There are different reasons that support this decision. Ideals establish a coherent link between the ordering defined by the inclusion relation between ideals and the *structural order* of its elements. Milner (Milner, 1978) notes that any typing formalism must share the following two properties:

- If $t : \tau$ and $t' \prec t$ then $t' : \tau$. Types are inherited from less structured terms.
- If $t_1 \prec t_2 \prec \ldots t_i \prec \ldots$ is a sequence of τ-typed terms ($t_i : \tau$ for $i \in \mathbb{N}$) then $\sqcup_{i \in \mathbb{N}} t_i : \tau$. The least structured term of a sequence of terms sharing the same type, also shares such type.

These properties are perfectly modeled by closed $\prec$-order ideals. These and other similar reasons have also been given in (MacQueen et al., 1986) to model polymorphic types as ideals.

Given that we want λ-expressions to denote ideals and that the set of ideals is closed under union, intersection and cartesian product (see lemma 3.25) it seems natural to extend pure λ-calculus with these set operators. We want to give semantics to these expressions in a domain of closed order ideals $\mathcal{J}(U)$ built over a functional domain U, i.e. over the initial solution of a recursive domain equation like:

$$U \cong C + U \times U + U \to U$$

where C is any initially given domain and $U \to U$ stands for the set of continuous functions w.r.t. the *computational order* relation $\sqsubseteq$. The notion of computational order was introduced by Scott (Scott, 1972; Scott, 1976) and is defined intuitively by $t \sqsubseteq t'$ if t contains *less computational information* than t'. Our principal decision is to identify the structural order $\prec$ and the inverse of computational order $\sqsubseteq$, i.e. to define $t \prec t'$ iff $t' \sqsubseteq t$. We prove that if the value domain U is functional (it can be defined a retract from a subset of $U \to U$ to U), then the domain of $\prec$-order ideals $\mathcal{J}(U)$ is a semantic model of the extended λ-calculus (it can also be defined a retract from a subset of $\mathcal{J}(U) \to \mathcal{J}(U)$ to $\mathcal{J}(U)$).

This chapter proceeds as follows. We start summarizing the standard technique formalized by Smyth and Plotkin (Smyth and Plotkin, 1982) for solving recursive domain equations in a given category. They use some categorical concepts that we introduce (not in detail) just to make the chapter self-contained. The rest of section 3.2 is devoted to the search of an *"ideal"* functor —a functor mapping domains to domains of $\prec$-ideals. We study four candidates and we try to justify each one of the decisions leading to the final choice (the functor $\mathcal{S}$). These candidates are the following ones:

$\mathcal{I}(D)$ The structural and the computational orderings are identified ($\prec \overset{def}{=} \sqsubseteq$), the computational ordering between ideals is defined by $I_1 \sqsubseteq I_2$ if $I_1 \subseteq I_2$, and the domain is built using (not necessarily closed) $\prec$-order ideals over a cpo structure.

$\mathcal{J}(D)$ Like the previous one, but using closed order ideals and domains instead of cpos.

$\mathcal{J}^C(D)$ The relations $\prec$ and $\sqsubseteq$ are identified, but the computational ordering between ideals is defined by $I_1 \sqsubseteq I_2$ iff $I_2 \subseteq I_1$.

$\mathcal{S}(D)$ Finally, we decide to identify the structural order and the inverse of the computational order ($\prec \stackrel{def}{=} \sqsupseteq$) and define the computational order between ideals as $I_1 \sqsubseteq I_2$ iff $I_1 \supseteq I_2$.

This is not the shortest way to introduce the functor $\mathcal{S}$, but we think it reflects closely the way how we found it. We prove the local continuity property for such functor. This ensures the existence of an initial solution for a wide range of domain equations, like $U \;\cong\; C + U \times U + \mathcal{S}(U) \to U$, which we will also use. In section 3.3 we prove the main result of this chapter: the domain of $\prec$-order ideals $\mathcal{S}(U)$, built over a functional domain $U \cong \ldots + U \to U + \ldots$, is also a functional domain. It allows to interpret λ-expressions not as values (in U) but as types (in $\mathcal{S}(U)$). Finally, we show in section 3.4 how a semantic interpretation function can be defined for such ideal model.

3.2 Value Domain Construction

In this section we will prove in detail the existence of a least fixed point of the recursive domain equation:

$$D \quad\cong\quad F_\nu(D) \tag{3.1}$$

that is, the existence of an initial F_ν-algebra (U, α), where U is the initial fixed point of the domain equation, and α is the isomorphism

$$\alpha : F_\nu(U) \to U$$

Usually, the endofunctor F_ν, used to give semantics to values in a functional language, has the form $F_\nu(D) = C + (D \times D) + (D \to D)$ where $+$, $\times$ and $\to$ are well-known continuous functors. Techniques to solve the domain equation (3.1) in this case have been studied and do no require a further analysis (we summarize such technique in subsection 3.2.1). Such recursive domain is enough for interpreting constants (in C), pairs of values (in $U \times U$), and computable functions (in $U \to U$) as values (in U) by means of the isomorphism $\alpha : F_\nu(U) \to U$.

Scott (Scott, 1972) was the first to face the solution of such kind of recursive domain equations using the inverse limit construction. This was the first statement of the limit-colimit coincidence. Later he showed (Scott, 1976) how this construction can be avoided using an universal domain and a least fixed point construction. Wand (Wand, 1979) formalized the inverse limit construction in terms of enriched categories, the O-categories. Finally, Smyth and Plotkin (Smyth and Plotkin, 1982) studied the sufficient condition that a functor F and an O-category K have to satisfy in order to ensure the existence of a solution of the domain equation $D \cong F^E(D)$ in a category K^E, where K^E is the subcategory of K resulting from restricting arrows to be embeddings, and $F^E : K^E \to K^E$ is a covariant functor on K^E defined from F. Such formulation is based on the coincidence of colimits (inverse limits) in K^E and limits in K. This is the most general formulation of the limit-colimit coincidence that we know. As Smyth and Plotkin says (Smyth and Plotkin, 1982), "the relation between the *category-theoretic treatment* [based on the inverse-limit construction or limit-colimit coincidence] and the *universal domain method* has, until now, remained rather obscure". The relation between both methods still remains obscure, and this chapter contributes to close the gap between them. We will use the category-theoretic approach mainly to solve the valued domain equation (3.1), obtaining a functional value domain U, which we use to build over an ideal domain $\mathcal{J}(U)$. Such ideal domain is an *universal domain* used to found a solution to the type domain equation, using the universal domain method.

In the following we give some references describing the standard technique that we present in subsection 3.2.1. Pierce (Pierce, 1991) summarizes the Smith and Plotkin's paper and also introduces the necessary basic categorical definitions. The Plotkin's course notes on domains (Plotkin, 1983) and the Schmidt's book (Schmidt, 1988) are good introductions to denotational semantics and domain theory. Stoy (Stoy, 1978) also makes a good introduction to the inverse limit construction, however he does not use the categorical formulation. Gunter and Scott (Gunter and Scott, 1990) summarize the main techniques for the semantic domain construction based on the universal domain method.

In some cases we will want to solve the domain equation (3.1) using the endofunctor $F_\nu(D) = C + D \times D + (\mathcal{S}(D) \to D)$, where $\mathcal{S}$ is a functor mapping a domain D to a subset of its corresponding power set. Some kinds of powerdomains have already been studied, for instance, the Plotkin's powerdomain (Plotkin, 1976) and the Smyth's powerdomain (Smyth, 1978), but none of them satisfies our requirements. The concrete definition of the functor $\mathcal{S}$ will be motivated step by step in subsections from 3.2.2 to 3.2.6.

3.2.1 The Standard Technique

In this subsection we present some well-known definitions of category concepts and we summarize the Smyth and Plotkin results of (Smyth and Plotkin, 1982). They have been included just to make the chapter self-contained.

Definition 3.1 *A partial ordered set (poset) $(D, \sqsubseteq_D)$ is said to be a* **complete partial order**, *cpo for short, if*

(i) *D has a minimum (or bottom) element, noted by $\bot_D$, and*

(ii) *every increasing sequence $\{x_i\}_{i \in \mathbb{N}}$ has a least upper bound (lub) in D, noted by $\bigsqcup_{i \in \mathbb{N}} x_i$.*

If it exists, the greatest lower bound (glb) of a decreasing sequence is noted by $\sqcap_{i \in \mathbb{N}} x_i$.

Given two cpos $(D, \sqsubseteq_D)$ and $(E, \sqsubseteq_E)$, a map $f : D \to E$ is said to be **continuous** *if for every increasing sequence $\{x_i\}_{i \in \mathbb{N}}$ of elements of D we have*

$\bigsqcup_{i\in\mathbb{N}} f(x_i) = f(\bigsqcup_{i\in\mathbb{N}} x_i)$.[1]

Definition 3.2 *Let* **CPO** *be the category of cpos and continuous functions between cpos, and let D, E, D', E' be objects and $f : D \to D'$, $g : E \to E'$ be arrows of such category, the functors $\_ + \_, \_ \times \_ :$* **CPO** $\times$ **CPO** $\to$ **CPO** *and $\_ \to \_ :$* **CPO**^{op} $\times$ **CPO** $\to$ **CPO** *are defined as follows.*

(i) **coalesced sum**
$D + E \overset{def}{=} \{\langle d, \bot_E\rangle \mid d \in D\} \cup \{\langle \bot_D, e\rangle \mid e \in E\}$
where $\langle d, e\rangle \sqsubseteq_{D+E} \langle d', e'\rangle$ *iff* $d \sqsubseteq_D d'$ *and* $e \sqsubseteq_E e'$ *for objects, and*
$f + g(\langle x, y\rangle) \overset{def}{=} \langle f(x), g(y)\rangle$ *for arrows*
We also define $in_1(d) = \langle d, \bot_D\rangle$, $is_1(x) = true$ *iff* $x = \langle d, \bot_D\rangle$, $out_1(\langle d, \bot_D\rangle = d$, *and similarly* in_2, is_2 *and* out_2.
We have $\bot_{D+E} = \langle \bot_D, \bot_E\rangle$ *and* $\bigsqcup_{i\in\mathbb{N}}\langle a_i, b_i\rangle = \langle \bigsqcup_{i\in\mathbb{N}} a_i, \bigsqcup_{i\in\mathbb{N}} b_i\rangle$

(ii) **smash product**
$D \times E \overset{def}{=} \{\langle d, e\rangle \mid d \in D \wedge e \in E \wedge d = \bot_D \Leftrightarrow e = \bot_E\}$
$\langle d, e\rangle \sqsubseteq_{D\times E} \langle d', e'\rangle$ *iff* $d \sqsubseteq_D d'$ *and* $e \sqsubseteq_E e'$
$f \times g(\langle x, y\rangle) \overset{def}{=} \langle f(x), g(y)\rangle$
$\bot_{D+E} = \langle \bot_D, \bot_E\rangle$ *and* $\bigsqcup_{i\in\mathbb{N}}\langle a_i, b_i\rangle = \langle \bigsqcup_{i\in\mathbb{N}} a_i, \bigsqcup_{i\in\mathbb{N}} b_i\rangle$

(iii) **continuous function space**
$D \to E \overset{def}{=} \{f : D \to E \mid f \text{ is continuous}\}$
$f \sqsubseteq_{[D\to E]} g$ *if for any* $x \in D$ *we have* $f(x) \sqsubseteq_E g(x)$
$[f \to g](h) \overset{def}{=} g \circ h \circ f$
$\bot_{[D\to E]} = \lambda x . \bot_E$ *and* $\bigsqcup_{i\in\mathbb{N}} f_i = \lambda x . \bigsqcup_{i\in\mathbb{N}} f_i(x)$.

The same functors can be defined in the category **Dom** *of domains (see definition 3.16) and continuous functions between domains.*

Definition 3.3 *A* **fixed point** *of a recursive domain equation $X \cong F(X)$ in a category K, where $F : K \to K$ is an endofunctor, is as a pair (D, θ) where D is an object of K and $\theta : F(D) \to D$ is an isomorphism of K.*
It is an **initial fixed point** *if for any other fixed point (D', θ') there is a unique arrow $f : D \to D'$ such that $\theta' \circ F(f) = f \circ \theta$.*

The initiality of such solutions is important in order to ensure the validity of structural induction principles.

To understand the Smith and Plotkin (Smyth and Plotkin, 1982) category-theoretic formulation, for the solution of such kind of equations, it is useful to have in mind the well-known theorem that ensures that any continuous function f in a cpo has a least fixed point, and this one is $\bigsqcup_{n\in\mathbb{N}} f^n(\bot)$. Using this analogy, cpos would be equivalent to ω-categories, where the elements of the cpo are equivalent to the objects of the category and the $a \sqsubseteq b$ pairs are equivalent to the $a \to b$ arrows, a function would be equivalent to a functor, the bottom element of the cpo to the initial object of the category, an increasing sequence of elements to an ω-chain, the least upper bound of an increasing sequence to

[1]Note that f continuous implies f monotonic and, therefore, $\{f(X_i)\}_{i\in\mathbb{N}}$ is an increasing sequence.

the colimit of the corresponding ω-chain, and a continuous function to a ω-continuous functor. The formal definitions of the categoric concepts we have introduced are as follows.

Definition 3.4 *Let K be a category, then*

(i) *$\perp_K$ is an* **initial object** *of K if for any object A there exists a unique arrow $f : \perp_K \to A$,*

(ii) *$\Delta = \langle A_n, f_n \rangle_{n \in \mathbb{N}}$ is a ω-**chain** if $f_n : A_n \to A_{n+1}$ for every $n \in \mathbb{N}$,*

(iii) *if $F : K \to K$ is an endofunctor, $F(\Delta)$ is the ω-chain defined by the arrows $F(f_n) : F(A_n) \to F(A_{n+1})$,*

(iv) *$\mu : \Delta \to A$ is a* **cone** *if there exists a sequence $\{\rho_n\}_{n \in \mathbb{N}}$ of arrows $\rho_n : A_n \to A$ such that $\rho_n = \rho_{n+1} \circ f_n$ for any $n \in \mathbb{N}$,*

(v) *$\mu : \Delta \to A$ is a* **colimiting cone** *if A is initial, that is, for any other cone $\mu' : \Delta \to A'$ there exists a unique arrow $\theta : A \to A'$ such that $\rho'_n = \theta \circ \rho_n$ for any $n \in \mathbb{N}$; here θ is named the* **mediating morphism,**

(vi) **terminal object**, *ω^{op}-**chain**,* **cocone** *and* **limit** *are respectively, the dual[2] definitions of initial object, ω-chain, cone and colimit.*

(vii) *K is a ω-**category** if it has an initial object $\perp_K$ and any ω-chain has colimit,*

(viii) *a functor $F : K \to K'$ is ω-**continuous** if whenever $\mu : \Delta \to A$ is a colimiting cone, $F(\mu) : F(\Delta) \to F(A)$ is also a colimiting cone.*

Notice that colimits of ω-chains are unique up to isomorphisms, that is, if A and A' are both colimits of the same ω-chain, then the (unique) arrow from A' to A is an isomorphism.

The following lemma is an adaptation of Smyth and Plotkin basic lemma 2.

Lemma 3.5 *(Smyth and Plotkin, 1982; lemma 2). Let K be a ω-category and let $F : K \to K$ be a ω-continuous functor, then there exists an initial fixed point of $X \cong F(X)$. The initial fixed point is (A, θ), where A is the colimit of the ω-chain defined by*

$$\perp_K \xrightarrow{f} F(\perp_K) \xrightarrow{F(f)} \cdots F^n(\perp_K) \xrightarrow{F^n(f)} F^{n+1}(\perp_K) \cdots$$

where $\perp_K$ is an initial object of the category and f is the unique arrow from this object $\perp_K$ to $F(\perp_K)$; and $\theta : F(A) \to A$ is the mediating morphism from $F(A)$ to the colimit A.

Proof: Notice that Δ and $F(\Delta)$ are the same ω-chain shifted one position to the right, therefore they have the same colimit up to isomorphism. Initiality is proved taking into account that we have built the ω-chain starting from an initial object $\perp_K$. The complete proof can be found in (Smyth and Plotkin, 1982). ■

[2] Where all arrows have been oriented in the opposite direction.

Unfortunately, the categories **Dom** and **CPO**, usually used to build such chains, are not ω-categories. Intuitively, using the analogy with cpos, the problem is that ω-chains are equivalent to sequences, and not every sequence has a lub. We have to restrict ω-chains to be *increasing*. The solution is *to restrict* the arrows we use to build a chain. It can be done defining a subcategory $\mathbf{Dom}^E$ of **Dom** or $\mathbf{CPO}^E$ of **CPO**. This is the main contribution of the Smyth and Plotkin's paper (Smyth and Plotkin, 1982).

First, to define K^E, the category K has to be an O-category. O-categories are enriched categories where a partial order between arrows with the same domain and codomain is defined.

Definition 3.6 *A category K is said to be a* **O-category** *(Wand, 1979) if*

(i) *for any pair of objects A and B we can define a cpo structure in the set of arrows $K(A,B)$ from A to B, and*

(ii) *composition is monotonic w.r.t. this order between arrows, that is $f \sqsubseteq f' \wedge g \sqsubseteq g' \Rightarrow f \circ g \sqsubseteq f' \circ g'$ and $\bigsqcup_{n\in\mathbf{N}}(f_n \circ g_n) = (\bigsqcup_{n\in\mathbf{N}} f_n)\circ(\bigsqcup_{n\in\mathbf{N}} g_n)$.*

If K is an O-category then we can consider a special kind of arrows named embeddings and projections.

Definition 3.7 *Let K be an O-category, then the K-arrow $f : A \to B$ is said to be an* **embedding** *and $f^R : B \to A$ to be a* **projection** *if*

(i) *$f^R \circ f = id_A$, and*

(ii) *$f \circ f^R \sqsubseteq id_B$.*

Given an embedding (or a projection) the corresponding projection (or embedding) is uniquely determined. We will note $f : D \lhd E$ when f is an embedding, $f : D \rhd E$ when it is a projection, and $f : D \cong E$ when it is an isomorphism.

The category K^E has as objects the objects of K and as arrows those arrows of K that are embeddings. It is easy to prove that if K is an O-category then K^E is a subcategory of K. Smith and Plotkin prove also the following result.

Theorem 3.8 *(Smyth and Plotkin, 1982; theorems 1 and 2) Let K be an O-category, then*

(i) *if $\perp$ is a terminal object in K, then it is also an initial object in K^E,*

(ii) *if every ω^{op}-chain in K has limit, then every ω-chain in K^E has colimit.*

This lemma will be used to prove that $\mathbf{Dom}^E$ is a ω-category. The coincidence between limit and colimits stated by the previous theorem was already discovered by Scott (Scott, 1972).

Secondly, we have to translate any functor $F : K \to K$ into a functor $F^E : K^E \to K^E$, and study which conditions F has to satisfy in order to be F^E ω-continuous. This translation is made in such way that contravariant, covariant, and mixed functors are all them translated into covariant functors.

Definition 3.9 *Let $F : K_1 \times K_2 \to K$ be a functor contravariant in its first argument and covariant in the second, then we define $F^E : K_1^E \times K_2^E \to K^E$ by*

(i) *$F^E(A, B) = F(A, B)$ for objects,*
(ii) *$F^E(f, g) = F(f^R, g)$ for arrows.*

This definition can be made completely general to functors with several co-variant or contravariant arguments.

Definition 3.10 *Let $F : K_1 \times K_2 \to K$ be a functor contravariant in its first argument and covariant in the second, then it is said to be* **locally continuous** *if for any increasing sequence $\{f_n\}_{n\in\mathbb{N}}$ in K_1^{op} and any increasing sequence $\{g_n\}_{n\in\mathbb{N}}$ in K_2 we have $F(\bigsqcup_{n\in\mathbb{N}} f_n, \bigsqcup_{n\in\mathbb{N}} g_n) = \bigsqcup_{n\in\mathbb{N}} F(f_n, g_n)$.*

Smith and Plotkin prove also the following result.

Theorem 3.11 *(Smyth and Plotkin, 1982; theorem 3) Let K be an O-category where any ω^{op}-chain has a limit, and $F : K \times \ldots \times K$ be a locally continuous functor, then F^E is a ω-continuous functor.*

Theorems 3.8 and 3.11 prove the premises of theorem 3.5, and this one proves the existence of an initial fixed point of a recursive domain equation $X \cong F(X)$ in a given category K.

It is well-known (Smyth and Plotkin, 1982; Plotkin, 1983) that the categories **CPO** and **Dom** have terminal object (the domain with a unique point $\perp$), and any ω^{op}-chain has a limit. As far the category **CPO** has not categorical sums, we have to take $\mathbf{CPO}_\perp$, i.e. the category of cpos and continuous strict functions between cpos (a function is strict if $f(\perp) = \perp$). It makes no difference because embeddings and projections are strict functions, therefore $\mathbf{CPO}^E = \mathbf{CPO}^E_\perp$. Then it has been proved that the coalesced sum $\_ + \_$ (the disjoint union of two cpos where the bottom elements are identified), the smash product $\_ \times \_$ (the cartesian product where pairs $\langle a, \perp\rangle$ and $\langle \perp, a\rangle$ with $a \neq \perp$ are not considered), and the strict continuous function space $[\_ \to \_]$ defined in 3.2 are all them locally continuous functors.

Therefore, the only point that remains to be proved —and this is our original contribution in this section— is the local continuity of any functor used in the definition of F_ν apart from these three ones.

3.2.2 The Functor $\mathcal{I}$

As we have said, in some cases we would need to solve the domain equation (3.1) using an endofunctor $F_\nu(D) = \ldots + \mathcal{S}(D) \to D + \ldots$ where functions from types to values are also considered. In such definition, if D is the domain of values, $\mathcal{S}(D)$ will be the domain of types of values. Given a domain of values there is not a unique way of building its corresponding domain of types. One of the proposed constructions are order ideals (see (MacQueen et al., 1986)) defined as follows.

Definition 3.12 *Given a cpo $(D, \sqsubseteq_D)$, a subset $I \subseteq D$ is said to be an* **order ideal**, *noted $I \in \mathcal{I}(D)$, if*

(i) *$I \neq \emptyset$ and*
(ii) *whenever $y \sqsubseteq_D x$ and $x \in I$ we have $y \in I$.*
An order ideal $I \subseteq D$ is said to be a **closed ideal**, *noted $I \in \mathcal{J}(D)$, if in addition*
(iii) *for every increasing sequence $\{x_i\}_{i \in \mathbb{N}}$ in I we have $\bigsqcup_{i \in \mathbb{N}} x_i \in I$.*

It makes sense to use (not necessarily closed) order ideals and defining a $\mathcal{I}$ functor as follows.

Definition 3.13 *The functor $\mathcal{I} : \mathbf{CPO} \to \mathbf{CPO}$ is defined by*

$$\mathcal{I}(D) = \{I \subseteq D \mid I \text{ is an order ideal}\}$$

with the order relation given by $I \sqsubseteq_{\mathcal{I}(D)} I'$ iff $I \subseteq I'$, for any object D (cpo), and

$$\mathcal{I}(f) = \lambda I \,.\, \{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E f(x)\}$$

for any arrow $f : D \to E$ (continuous function between cpos).

Proof: We have to prove that $\mathcal{I}$ is really a functor.

If D is a cpo, it is easy to prove that the set of order ideals of D with the set inclusion order relation is a cpo with bottom element $\bot_{\mathcal{I}(D)} = \{\bot_D\}$ and least upper bound $\bigsqcup_{n \in \mathbb{N}} I_n = \bigcup_{n \in \mathbb{N}} I_n$.

We have to prove that if $f : D \to E$ is a continuous function between cpos, then $\mathcal{I}(f) : \mathcal{I}(D) \to \mathcal{I}(E)$ is also a continuous function between cpos. Trivially, for any set I, the set $\{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E f(x)\}$ is an order ideal of E. Let's prove now that $\mathcal{I}(f)$ is continuous.

$$\begin{array}{rcl}
\mathcal{I}(f)(\bigsqcup_{n \in \mathbb{N}} I_n) & = & \{y \in E \mid \exists x \in \bigcup_{n \in \mathbb{N}} I_n \,.\, y \sqsubseteq_E f(x)\} \\
& = & \bigcup_{n \in \mathbb{N}} \{y \in E \mid \exists x \in I_n \,.\, y \sqsubseteq_E f(x)\} \\
& = & \bigcup_{n \in \mathbb{N}} \mathcal{I}(f)(I_n)
\end{array}$$

Finally, we have to prove the compositional properties:

$$\mathcal{I}(id_D)(I) = \{y \in D \mid \exists x \in I \,.\, y \sqsubseteq_E x\} = I$$

and

$$\begin{array}{rcl}
\mathcal{I}(g) \circ \mathcal{I}(f)\ (I) & = & \{z \in F \mid \exists y \in \{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E f(x)\} \,.\, z \sqsubseteq_F g(y)\} \\
& = & \{z \in F \mid \exists x \in I \,.\, \exists y \in E \,.\, y \sqsubseteq_E f(x) \wedge z \sqsubseteq_F g(y)\}
\end{array}$$

we have $f(x) \in E$ for any $x \in I \subseteq D$, thus we can take $y \overset{def}{=} f(x)$ and then

$$\begin{array}{rcl}
\ldots & = & \{z \in F \mid \exists x \in I \,.\, z \sqsubseteq_F g(f(x))\} \\
& = & \mathcal{I}(g \circ f)(I)
\end{array}$$

■

We could define the cpo $\mathcal{I}(D)$ using the opposite order relation, that is, $I \sqsubseteq_{\mathcal{I}(D)} I'$ iff $I' \subseteq I$. In this case the bottom element would be D, and the lub of an increasing sequence $\{I_n\}_{n \in \mathbb{N}}$ would be $\bigcup_{n \in \mathbb{N}} I_n$. We discuss this possibility in subsection 3.2.5.

Lemma 3.14 *The functor $\mathcal{I}$ is locally monotonic, but it is not locally continuous.*

Proof: It is easy to prove that it is monotonic and therefore

$$\bigsqcup_{n\in\mathbb{N}}\mathcal{I}(f_n) \sqsubseteq \mathcal{I}(\bigsqcup_{n\in\mathbb{N}} f_n)$$

However the inclusion in the opposite direction does not hold. We have

$$\begin{array}{rcl}\mathcal{I}(\bigsqcup_{n\in\mathbb{N}} f_n)(I) & = & \{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E \bigsqcup_{n\in\mathbb{N}} f_n(x)\} \\ \bigsqcup_{n\in\mathbb{N}}\mathcal{I}(f_n)(I) & = & \bigcup_{n\in\mathbb{N}}\{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E f_n(x)\}\end{array}$$

For any $x \in I$ we can define the sequence $\{f_n\}_{n\in\mathbb{N}}$ such that $\{f_n(x)\}_{n\in\mathbb{N}}$ is an increasing sequence with $\bigsqcup_{n\in\mathbb{N}} f_n(x) \neq f_m(x)$ for any $m \in \mathbb{N}$. Then, $\bigsqcup_{n\in\mathbb{N}} f_n(x) \in \mathcal{I}(\bigsqcup_{n\in\mathbb{N}} f_n)(I)$ but $\bigsqcup_{n\in\mathbb{N}} f_n(x) \notin \bigcup_{n\in\mathbb{N}}\mathcal{I}(f_n)(I)$. ■

As we have said, to prove the existence of a solution of a recursive equation $X \cong F(X)$, based on an initial fix point theorem, it is essential the local continuity of the functor F (see theorem 3.11). The non-continuity of $\mathcal{I}$ impossibilities such prove.

The problem is due to the non-closure of $\bigcup_{n\in\mathbb{N}}\mathcal{I}(f_n)(I)$. (The lub $\bigsqcup_{n\in\mathbb{N}} f_n(x)$ belongs to $\mathcal{I}(\bigsqcup_{n\in\mathbb{N}} f_n)(I)$ but not to $\bigcup_{n\in\mathbb{N}}\mathcal{I}(f_n)(I)$). The solution to such problem comes from using *closed order ideals*. In subsection 3.2.4 we define the functor $\mathcal{J}$ mapping domains to the corresponding set of closed order ideals. This functor maps continuous functions between domains $f : D \to E$ to a function $\mathcal{J}(f) : \mathcal{J}(D) \to \mathcal{J}(E)$. The function $\mathcal{J}(f)$ has to map any closed ideal $I \in \mathcal{J}(D)$ to the minimum closed ideal containing $\mathcal{J}(f)(I)$. It requires to close $\mathcal{J}(f)(I)$ by smaller elements and by lub of increasing sequences. The first closure was already studied for $\mathcal{I}$. The second one requires the definition of a *closure operator* for lub of increasing sequences, that will be discussed in next subsection.

3.2.3 The Category of Domains

Solutions to isomorphism equations like (3.1) are usually found using cpos. However, it is rather difficult to face the solution of this equation using only the properties of cpos. For instance, the definition 3.20 of the functor $\mathcal{J}$, the proof of its continuity (theorem 3.29), and the definition 3.46 of the encoding function $\mathsf{Code}_{\to}$ rely on having a constructive way to calculate the least upper bound (lub) of increasing sequences of ideals, and the minimum closed ideal that contains a given set. There is no a constructive way to find these ideals working with the set of closed ideals $\mathcal{J}(U)$ of a cpo U, even knowing that $\mathcal{J}(U)$ is a complete lattice and, therefore, such ideals exist. As has been shown in (MacQueen et al., 1986) when ideals are involved the suitable structure to work with is the category **Dom** of domains and continuous functions between domains.

If we close a set A adding all the lub of increasing sequences of elements from A, we obtain a set $\overline{A}$ with new elements which makes possible to define new increasing sequences and makes necessary to close the set again $\overline{\overline{A}}$. The natural property of cpo ensuring that $\overline{\overline{A}} = \overline{A}$ is the ω-algebraic property. However, the function space construction does not preserve ω-algebraic property (see (Plotkin, 1983) for a counter-example). Fortunately, this problem can be avoided if we require cpos to be also consistently complete. Consistently complete ω-algebraic cpos are usually named *domains*.

In this subsection we introduce the standard definitions of ω-algebraic and consistent completeness, that is, of domains. Here *domain* has a precise meaning: is a cpo with some additional properties. Basically, it is a cpo with a denumerable basis of elements, such that any other element is the lub of an increasing sequence of elements from the base.

Definition 3.15 *Let D be a cpo.*

(i) *We say that $x \in D$ is ω-**finite** if for any increasing sequence $\{a_i\}_{i\in\mathbb{N}}$ with $x \sqsubseteq \bigsqcup_{i\in\mathbb{N}} a_i$ there exists $k \in N$ such that $x \sqsubseteq a_k$.*

(ii) *A countable subset B of D is a ω-**basis** of D if for any $x \in D$ the set $B(x) \overset{def}{=} \{y \in B \mid y \text{ is } \omega\text{-finite } \wedge\ y \sqsubseteq_D x\}$ is directed and $\sqcup B(x) = x$.*

Definition 3.16 *Let D be a cpo.*

(i) *We say that D is **consistently complete** if every upper bound set $X \subseteq D$ has least upper bound.*

(ii) *We say that D is ω-**algebraic** if D has a ω-basis of finite elements.*

(iii) *And, we say that D is a **domain** if D is consistently complete and D is ω-algebraic.*

From now on, **Dom** will be the category of domains and continuous functions between domains.

The following lemma gives an alternative definition of ω-algebraic and justifies the intuition we have given: ω-algebraic elements of a domain are a countable basis that generates all the domain elements using only the least upper bound operator $\sqcup$.

Lemma 3.17 *Let D be a ω-algebraic cpo. For every $x \in D$ there exists an increasing sequence $\{x_i\}_{i\in\mathbb{N}}$ of ω-finite elements of D with $\bigsqcup_{i\in\mathbb{N}} x_i = x$.*

Proof: The set of ω-finite elements less than x is countable and directed. Let $\{a_1, \ldots, a_n, \ldots\}$ be this set. Because it is directed, we can construct the increasing chain $a_1 \sqsubseteq c_{1,2} \sqsubseteq \ldots \sqsubseteq c_{1,2,\ldots,n} \sqsubseteq \ldots$ where $c_{1,2,\ldots,n}$ is an upper bound of $\{c_{1,\ldots,n-1}, a_n\}$ belonging to the set of ω-finite elements less than x, that is, belonging to $\{a_1, \ldots, a_n, \ldots\}$. Thus, it's easy to see that $\bigsqcup_{i\in\mathbb{N}} c_{1,\ldots,i} = x$. ∎

3.2.4 The Functor $\mathcal{J}$

In this section we define the functor $\mathcal{J}$ in the category **Dom** of domains and continuous functions between domains.

The set of closed ideals of a cpo, like the set of order ideals, with the order relation defined by the inclusion relation form a cpo. However, the calculation of the lub of an increasing sequence of ideals is not straightforward (see lemma 3.25). The set of closed ideals of a domain, with the same order relation, form a domain (see lemma 3.26). In this case, the lub of increasing sequences is easily calculable (see lemma 3.19) thanks to the existence of a closure operator. As we have said, the definition of the $\mathcal{J}$ functor also relies on such closure operator. This operator is defined as follows.

Definition 3.18 *Let D be a domain, and let $X \subseteq D$ be a subset. We define*

$$\begin{array}{lcl} \mathcal{I}_{[X]} & \overset{def}{=} & \{x \in D \mid \exists y \in X \,.\, x \sqsubseteq y\} \\ X^{o} & \overset{def}{=} & \{x \in X \mid x \text{ is } \omega\text{-finite}\} \\ \overline{X} & \overset{def}{=} & \{\bigsqcup_{i \in \mathbb{N}} a_i \mid \{a_i\}_{i \in \mathbb{N}} \text{ is an increasing sequence in } X\} \end{array}$$

We can prove the following properties of these operators.

Lemma 3.19 *Let D be a domain, and $X \subseteq D$ a subset, then*
(i) *the set $\mathcal{I}_{[X]}$ is an order ideal,*
(ii) *if X is finite then $\mathcal{I}_{[X]}$ is also a closed order ideal,*
(iii) *the map $C : \mathcal{I}(D) \to \mathcal{J}(D)$ defined by $C(X) = \overline{X}$ is a closure operator.*
(iv) *for any $I \in \mathcal{J}(D)$ we have $\overline{I^o} = I$.*
(v) *for any increasing sequence $\{I_i\}_{i \in \mathbb{N}}$ in $\mathcal{J}(D)$ we have*

$$\bigsqcup_{i \in \mathbb{N}} I_i = \overline{\bigcup_{i \in \mathbb{N}} I_i^o}$$

Many other properties, as $\forall I_1, I_2 \in \mathcal{J}(D) \,.\, I_1 = I_2 \Leftrightarrow I_1^o = I_2^o$ or $(\bigsqcup_{i \in \mathbb{N}} I)^o = \bigcup_{i \in \mathbb{N}} I^o$, of closed ideals of a domain are not mentioned. Some of them are used in the proof of the following lemmas and theorems, and may be easily proved.

We define the functor $\mathcal{J}$ as follows.

Definition 3.20 *The functor $\mathcal{J}$ is defined in the category of domains and continuous function between domains by:*

$$\mathcal{J}(D) = \{I \subseteq D \mid I \text{ is a closed ideal}\}$$

for objects, and by

$$\mathcal{J}(f) = \lambda I \,.\, \overline{\{y \in E \mid \exists x \in I \,.\, y \sqsubseteq_E f(x)\}}$$

for arrows (continuous function $f : D \to E$ between domains).

Notice that $\mathcal{J}(f)(I)$ is the minimum closed order ideal containing $f(I)$.

In the following we will prove that $\mathcal{J}$ is really a functor in the category of domains.

In order to prove that the set of closed order ideals of a domain is a domain, we characterize the set of w-finite elements of $\mathcal{J}(D)$, that is, the set $\mathcal{J}(D)^o$. We need to introduce the following definition.

Definition 3.21 *A set X is said to be* **maximal complete** *if for every element $x \in X$, there exists a maximal element $m \in X$ such that $x \sqsubseteq m$. Here, $m \in X$ being maximal means that $\forall y \in X \,.\, m \not\sqsubseteq y$.*

The following lemma relates the set of maximal elements of an ideal and the ideals generated by a finite set of elements.

Lemma 3.22 *Any ideal $\mathcal{I}_M$ generated by a finite set M is maximal complete and the set of maximal elements of $\mathcal{I}_M$ is a subset of M.*
Any maximal complete ideal I is generated by the set M of its maximal elements $I = \mathcal{I}_M$.

These properties allow us to characterize the ω-finite ideals of $\mathcal{J}(D)$.

Lemma 3.23 *If D is a ω-algebraic cpo, then a closed ideal $I \in \mathcal{J}(D)$ is ω-finite ($I \in \mathcal{J}(D)^o$) if, and only if, it is maximal complete and the set of maximal elements is finite and contains only ω-finite elements of D.*

Proof:

$\Leftarrow$) We have to prove that if $I \subseteq \bigsqcup_{i\in\mathbb{N}} A_i$ then there exists a k such that $I \subseteq A_k$. Let $\{a_1, \ldots, a_n\}$ be the finite set of maximal elements of I. We have $I \subseteq \bigsqcup_{i\in\mathbb{N}} A_i = \overline{\bigcup_{i\in\mathbb{N}} A_i^o}$, therefore, for any maximal elements a_r we can say $a_r \in \overline{\bigcup_{i\in\mathbb{N}} A_i^o}$. But, as far as a_r is an ω-finite element there exists $k_r \in \mathbb{N}$ such that $a_r \in A_{k_r}^o$. There are a finite number of maximal elements, thus there exists $k = \max\{k_1, \ldots, k_n\}$ such that $a_r \in A_k$ for all $r = 1, \ldots, n$. Using lemma 3.22 it is easy to see that $I \subseteq A_k$.

$\Rightarrow$) As far as D is ω-algebraic, I^o must be countable. Let $I^o = \{a_i \mid i \in N\}$ be an enumeration of I^o, and $\mathcal{J}_{[\{a_0\}]} \sqsubseteq \mathcal{J}_{[\{a_0,a_1\}]} \sqsubseteq \cdots \sqsubseteq \mathcal{J}_{[\{a_0,\ldots,a_n\}]} \sqsubseteq \cdots \sqsubseteq I$ be the increasing sequence of ideals generated by $\underline{\{a_0\}}, \underline{\{a_0,a_1\}}, \ldots, \underline{\{a_0,\ldots,a_n\}}, \ldots$ An easy computation shows that $I = \overline{I^o} = \overline{\bigcup_{i\in\mathbb{N}}\{a_0,\ldots,a_i\}} = \overline{\bigcup_{i\in\mathbb{N}} \mathcal{J}^o_{[\{a_0,\ldots,a_i\}]}} = \bigsqcup_{i\in\mathbb{N}} \mathcal{J}_{[\{a_0,\ldots,a_i\}]}$. Now, if I is ω-algebraic then there exists $k \in \mathbb{N}$ such that $I = \mathcal{J}_{[\{a_0,\ldots,a_k\}]}$. Lemma 3.22 ensures that $\mathcal{J}_{[\{a_0,\ldots,a_k\}]}$ is maximal complete, and the set of maximal elements is a subset of $\{a_0, \ldots, a_k\}$ (there are finite many of them, and they are ω-finite). ■

Lemma 3.24 *If D is ω-algebraic, $\mathcal{J}(D)$ is also ω-algebraic.*

Proof: First, there are countably many ω-finite ideals because they are characterized by a finite number of ω-finite elements from D, and the ω-finite elements of D are denumerable.

Second, the set of ω-finite ideals less than a given closed ideal $I \in \mathcal{J}(D)$ is directed. The proof is an easy consequence of the equality

$$\mathcal{J}_{[\{a_1,\ldots,a_n\}]} \cup \mathcal{J}_{[\{b_1,\ldots,b_m\}]} = \mathcal{J}_{[\{a_1,\ldots,a_n,b_1,\ldots,b_m\}]}$$

Third, the lub of such set is I. If D is ω-algebraic then

$$I = \overline{I^o} = \overline{\bigcup_{i\in\mathbb{N}}\{a_0,\ldots,a_i\}} = \overline{\bigcup_{i\in\mathbb{N}} \mathcal{J}^o_{[\{a_0,\ldots,a_i\}]}} = \bigsqcup_{i\in\mathbb{N}} \mathcal{J}_{[\{a_0,\ldots,a_i\}]}$$

where $I^o = \{a_i\}_{i\in\mathbb{N}}$. The ideals $\mathcal{J}_{[\{a_0,\ldots,a_i\}]}$ are ω-algebraic, which proves that I is less than the lub of ω-algebraic ideals less that I. ■

To prove that the set $\mathcal{J}(D)$ is consistently complete we use the following lemma.

Lemma 3.25 *Let $(\mathcal{J}(D), \sqsubseteq_{\mathcal{J}(D)})$ be the cpo of closed ideals of D. If X,$Y \in \mathcal{J}(D)$ then $X \cup Y$, $X \cap Y$, $X \otimes Y \in \mathcal{J}(D)$. Moreover $(\mathcal{J}(D), \bigcup, \bigvee)$ is a complete lattice where the meet operator is the usual intersection and the joint operator is defined by $\bigsqcup_{i \in R} X_i \overset{def}{=} \bigcup\{I \in \mathcal{J}(D) \mid \forall i \in R \,.\, X_i \subseteq I\}$.*
Consequently, any bound subset of $\mathcal{J}(D)$ has a least upper bound (given by the joint operator).

Notice that in general, the infinite union of closed ideals is not a closed ideal. This fact makes necessary the definition of the previous *non-calculable* joint operator. However, if D is ω-algebraic then $\bigsqcup_{i \in R} X_i = \overline{\bigcup_{i \in R} X_i^o}$ which is *computable.*

Lemmas 3.24 and 3.25 may be summarized in the following lemma.

Lemma 3.26 *If D is a domain, then $\mathcal{J}(D)$ is also a domain.*

It can be proved easily that $\mathcal{J}(f)$ maps closed ideals over a domain to closed ideals.

The following lemma gives an alternative definition for $\mathcal{J}(f)$. This lemma is used widely to prove the following theorems.

Lemma 3.27

$$\mathcal{J}(f)(A) = \overline{\{y \in E \mid \exists x \in A \,.\, y \sqsubseteq f(x)\}} = \overline{\{y \in E^o \mid \exists x \in A^o \,.\, y \sqsubseteq f(x)\}}$$

Proof: Using $\overline{I} = \overline{I^o}$, that we have proved in 3.19, we have

$$\overline{\{y \in E \mid \exists x \in A \,.\, y \sqsubseteq f(x)\}} = \overline{\{y \in E^o \mid \exists x \in A \,.\, y \sqsubseteq f(x)\}}$$

Now, if x is not ω-finite, there exists an increasing sequence $\{x_i\}_{i \in \mathbb{N}}$ of ω-finite elements with $x = \bigsqcup_{i \in \mathbb{N}} x_i$. Using the continuity of f, $y \sqsubseteq f(x) = f(\bigsqcup_{i \in \mathbb{N}} x_i) = \bigsqcup_{i \in \mathbb{N}} f(x_i)$, and using the ω-finiteness of y, there exists $n \in N$ such that $y \sqsubseteq f(x_n)$, with $x_n \in A^o$, because $x_n \sqsubseteq x$ and A is an ideal set. Therefore, we can say that if there exists $x \in A$ with $y \sqsubseteq f(x)$, then there exists $x' \in A^o$ with $y \sqsubseteq f(x') \sqsubseteq f(x)$, which ensures:

$$\overline{\{y \in E^o \mid \exists x \in A \,.\, y \sqsubseteq f(x)\}} = \overline{\{y \in E^o \mid \exists x \in A^o \,.\, y \sqsubseteq f(x)\}}$$

■

Theorem 3.28 *The mapping $\mathcal{J}$ is a functor in the category of domains.*

Proof: We know that $\mathcal{J}$ maps domains to domains (lemma 3.26). It is easy to prove that if $f : D \to E$ is a function between domains then $\mathcal{J}(f)$ maps closed ideals of D to closed ideals of E.

We have to prove now that if $f : D \to E$ is continuous then so it is $\mathcal{J}(f) : \mathcal{J}(D) \to \mathcal{J}(E)$.

$$\begin{aligned}
\mathcal{J}(f)(\textstyle\bigsqcup_{n\in \mathbb{N}} I_n) &= \overline{\{y\in E^o \mid \exists x\in [\textstyle\bigsqcup_{n\in \mathbb{N}} I_n]^o \,.\, y\sqsubseteq f(x)\}}\\
&= \overline{\{y\in E^o \mid \exists x\in \textstyle\bigcup_{n\in \mathbb{N}} I_n^o \,.\, y\sqsubseteq f(x)\}}\\
&= \overline{\textstyle\bigcup_{n\in \mathbb{N}}\{y\in E^o \mid \exists x\in I_n^o \,.\, y\sqsubseteq f(x)\}}\\
&= \overline{\textstyle\bigcup_{n\in \mathbb{N}}\mathcal{J}(f)(I_n)} = \textstyle\bigsqcup_{n\in \mathbb{N}}\mathcal{J}(f)(I_n)
\end{aligned}$$

Let's proof now the compositional equalities.

$$\mathcal{J}(Id_D)(I) = \overline{\{x\in D \mid \exists y\in I \,.\, x\sqsubseteq y\}} = I$$

$$\begin{aligned}
[\mathcal{J}(g)\circ\mathcal{J}(f)](A) &= \overline{\{z\in F^o \mid \exists y\in \mathcal{J}(f)(A)^o \,.\, z\sqsubseteq g(y)\}}\\
&= \overline{\{z\in F^o \mid \exists y\in \{y'\in E^o \mid \exists x\in A^o \,.\, y'\sqsubseteq f(x)\} \,.\, z\sqsubseteq g(y)\}}\\
&= \overline{\{z\in F^o \mid \exists y\in E^o \,.\, \exists x\in A^o \,.\, y\sqsubseteq f(x) \text{ and } \,.\, z\sqsubseteq g(y)\}}\\
&= \ldots
\end{aligned}$$

But, because $z\sqsubseteq g(y)$ and $y\sqsubseteq f(x)$, by the monotonicity of g we can ensure that $z\sqsubseteq g(f(x))$. Then

$$\ldots \subseteq \overline{\{z\in F^o \mid \exists x\in A^o \,.\, z\sqsubseteq g(f(x))\}} = \mathcal{J}(g\circ f)(A)$$

In the other direction, from $z\sqsubseteq g(f(x))$ we have to find an ω-finite element y such that $z\sqsubseteq g(y)$ and $y\sqsubseteq f(x)$. We already know that x and z are ω-finite. If $f(x)$ is ω-finite we can take $y \overset{def}{=} f(x)$. If this is not the case, then there exists an increasing sequence $\{t_i\}_{i\in\mathbb{N}}$ of ω-finite elements such that $f(x) = \bigsqcup_{i\in\mathbb{N}} t_i$. Using the continuity of g we can say $z\sqsubseteq g(f(x))\sqsubseteq g(\bigsqcup_{i\in\mathbb{N}} t_i) = \bigsqcup_{i\in\mathbb{N}} g(t_i)$. And using the ω-finiteness of z, there exists $n\in N$ such that $z\sqsubseteq g(t_n)$. We can take then $y \overset{def}{=} t_n$, which ensures $y\sqsubseteq f(x)$ and the ω-finite of y, therefore we can say

$$\ldots \supseteq \overline{\{z\in F^o \mid \exists x\in A^o \,.\, z\sqsubseteq g(f(x))\}} = \mathcal{J}(g\circ f)(A)$$

■

Theorem 3.29 *The functor $\mathcal{J}$ is locally continuous.*

Proof: For any increasing sequence $\{f_i\}_{i\in\mathbb{N}}$ and any ideal $I\in\mathcal{J}(D)$ we have to prove that $\mathcal{J}(\bigsqcup_{i\in\mathbb{N}} f_i)(I) = \bigsqcup_{i\in\mathbb{N}}\mathcal{J}(f_i)(I)$.

Using lemma 3.27 we have

$$\mathcal{J}(\textstyle\bigsqcup_{i\in\mathbb{N}} f_i)(I) = \overline{\{y\in E^o \mid \exists x\in I^o \,.\, y\sqsubseteq \textstyle\bigsqcup_{i\in\mathbb{N}} f_i(x)\}} = \ldots$$

Using now that y is ω-finite, we can say that there exists $n\in\mathbb{N}$ such that $y\sqsubseteq f_n(x)$. Then

$$\overline{\{y\in E^o \mid \exists x\in I^o \,.\, \exists n\in N \,.\, y\sqsubseteq f_n(x)\}} = \overline{\textstyle\bigcup_{i\in\mathbb{N}}\mathcal{J}(f_i)(I)^o} = \textstyle\bigsqcup_{i\in\mathbb{N}}\mathcal{J}(f_i)(I)$$

■

Using domains and closed ideals we have got a locally continuous functor which makes possible the solution of (3.1).

Example. Alternatively, we could define the functor $\mathcal{J}$ for each arrow $f : D \longrightarrow E$ as follows

$$\mathcal{J}(f) = \lambda I : \mathcal{J}(D) . \{y \mid \exists x \in I . y \sqsubseteq_E f(x)\}$$

Such definition does not make use of the closure operator. However, as we show in figure 3.1, such definition is not correct because in general the set $\{y \mid \exists x \in I . y \sqsubseteq_E f(x)\}$ is non-closed. Notice that the set $\{\bot_D, a_0, a_1, \ldots, a_n, \ldots\}$ is a closed ideal, whereas its image $\{\bot_E, b_0, b_1, \ldots, b_n, \ldots\}$ is an open ideal. We have to take its closure $\{\bot_E, b_0, b_1, \ldots, b_n, \ldots, \bigsqcup_{i\in\mathbb{N}} b_i\}$ as image to obtain a closed ideal.

$$\begin{tikzcd}
& & & \bigsqcup_{i\in\mathbb{N}} b_i \\
& & & \vdots \\
& & & b_n \\
& & & \vdots \\
& & & b_1 \\
\cdots a_n \arrow[rrruuu, "f : D \longrightarrow E"] & \cdots a_1 \arrow[rru] & a_0 \arrow[r] & b_0 \arrow[u] \\
& \bot_D \arrow[ul] \arrow[u] \arrow[ur] \arrow[rr] & & \bot_E \arrow[u]
\end{tikzcd}$$

where $\forall n \in \mathbb{N} . f(a_n) = b_n$
and $\forall n \in \mathbb{N} . \bigsqcup_{i\in\mathbb{N}} b_i \neq b_n$

Figure 3.1: A counter-example to an alternative definition of the functor $\mathcal{J}$.

3.2.5 The Contravariant Functor $\mathcal{J}^C$

The properties proved for the functor $\mathcal{J}$ allow to find an initial fix point of our recursive domain equation. However, the solution is not completely satisfactory. Since Scott (Scott, 1972) proposed the use of lattices to find solutions to such kind of equations, the order relation $\sqsubseteq$ had an intended semantics in terms of information containment. The inequality $a \sqsubseteq b$ is interpreted as *b has more (computational) information than a* or *a is a computational approximation of b*. In the same way, if there exists an embedding $f : D \longrightarrow E$ between two cpos, then it is said that elements of D are approximations of those elements of E. Thus, the embedding f maps elements of D to elements of E *with the same information*, and the corresponding projection f^R maps elements of E to *their best approximation* in D. The equality $f^R \circ f(x) = x$ is interpreted as "the best approximation to any element containing the same information than x is x itself"; and $f \circ f^R(x) \sqsubseteq x$ is interpreted as "an element, containing the same information than the best approximation of x, contains less information than x".

We interpret terms as sets of values (as closed order ideals), therefore, the most sensible thing would be to consider a smaller set as containing more information than a bigger set. (The term $1 \cup 2$ contains less information than the term 1). The bottom element, the term not containing any information at all, is the term interpreted as the hole domain of values.

If we choose this interpretation, then the computational order between ideals we have proposed is not the correct one. We would have to define

$$I_1 \sqsubseteq_{\mathcal{J}(D)} I_2 \;\Leftrightarrow\; I_2 \subseteq I_1$$

Fortunately, the set of closed ideals with the relation $\subseteq$ have a complete lattice structure. Thus, if we take the inverse order we also obtain a cpo with

$$\begin{array}{rcl} \bot_{\mathcal{J}(D)} & = & D \\ \bigsqcup_{n\in\mathbb{N}} I_n & = & \bigcup_{n\in\mathbb{N}} I_n \end{array}$$

The infinite intersection of closed ideals is also a closed ideal, so by the moment, we do not need the ω-algebraic property.

Even if f is a continuous function $\mathcal{J}(f)$, as was defined in the previous subsection, is not necessarily a continuous function w.r.t. the new order relation, and $\mathcal{J}$ is no longer a functor. Fortunately, it is possible to define an ω-continuous functor $\mathcal{J}$ in $\mathbf{CPO}^E$. The idea is to define a contravariant functor $\mathcal{J}^C : \mathbf{CPO}^{op} \to \mathbf{CPO}$ such that $(\mathcal{J}^C)^E : \mathbf{CPO}^E \to \mathbf{CPO}^E$ would be ω-continuous.

The functor $\mathcal{J}^C$ may be introduced in a very natural fashion. We know that any embedding of $\mathbf{CPO}^E$ has a corresponding projection. The projection fitting the embedding $\mathcal{J}(f)$ is given by:

$$\begin{array}{rcl} \mathcal{J}(f)^R & = & \mathcal{J}(f^R) \\ & = & \lambda I : \mathcal{J}(E) \,.\, \overline{\{x \in D \mid \exists y \in I \,.\, x \sqsubseteq_D f^R(y)\}} \\ & = & \lambda I : \mathcal{J}(E) \,.\, \overline{\{x \in D \mid \exists y \in I \,.\, f(x) \sqsubseteq_E y\}} \\ & = & \lambda I : \mathcal{J}(E) \,.\, \overline{\{x \in D \mid f(x) \in I\}} \\ & = & \lambda I : \mathcal{J}(E) \,.\, \{x \in D \mid f(x) \in I\} \end{array}$$

where we use the implication $\{x \in D \mid f(x) \in I\}$ is a closed ideal if f is continuous and I is a closed ideal, the equivalence $x \sqsubseteq f^R(y) \Leftrightarrow f(x) \sqsubseteq y$, and the property $F(f)^R = F(f^R)$ for functors.

We have already said that with the order relation $\supseteq$, the function $\mathcal{J}(f)$ is not continuous and $\mathcal{J}$ is not a functor. Conversely, with the opposite order relation $\subseteq$, the function $\mathcal{J}(f)^R$ is not necessarily continuous. However, notice that if f is an embedding, then $\mathcal{J}^C(f) \stackrel{def}{=} \mathcal{J}(f)^R$ is the projection corresponding to the embedding $\mathcal{J}(f)$, and therefore it is continuous.

The above equality suggests the definition of the following contravariant functor.

Definition 3.30 *The functor $\mathcal{J}^C : \mathbf{CPO}^{op} \to \mathbf{CPO}$ is defined by*

$$\mathcal{J}^C(D) \stackrel{def}{=} \{I \mid I \textit{ is a closed ideal, w.r.t. the order relation defined by } \sqsubseteq_{\mathcal{J}^C(D)} \stackrel{def}{=} \supseteq\}$$

for any object D, and by

$$\mathcal{J}^C(f) \stackrel{def}{=} \lambda I : \mathcal{J}(E) \,.\, \{x \in D \mid f(x) \in I\}$$

for any arrow $f : D \to E$.

Theorem 3.31

(i) *$\mathcal{J}^C$ is a contravariant functor*

(ii) *$\mathcal{J}^C$ is locally continuous.*

Proof: It is a straightforward exercise. ■

We have $\mathcal{J}(D) \neq \mathcal{J}^C(D)$ for objects. Notice also that $\mathcal{J}^E \neq (\mathcal{J}^C)^E$ (!!) because although $\mathcal{J}(f^R) = \mathcal{J}^C(f)$ by definition, we have $\mathcal{J}^E(f) = \mathcal{J}(f) \subseteq \mathcal{J}^C(f^R) = (\mathcal{J}^C)^E(f)$ but in general $\mathcal{J}^E(f) = \mathcal{J}(f) \not\supseteq \mathcal{J}^C(f^R) = (\mathcal{J}^C)^E(f)$.

We can concluded that a ω-continuous functor $\mathcal{J}^E : \mathbf{CPO}^E \to \mathbf{CPO}^E$ can be defined without using the properties of domains, and considering the order relation between closed ideals given by $\sqsubseteq_{\mathcal{J}^C(D)} \overset{def}{=} \supseteq$.

3.2.6 The Functor $\mathcal{S}$

In the previous subsection we have motivated the following definition of computational ordering between types (sets of values): a type I_2 is a better approximation than other type I_1, (written $I_1 \sqsubseteq I_2$) if it contains less values (that is if $I_2 \subseteq I_1$). However, there are two different orders involved which we have misled, the *computational order* $\sqsubseteq$ (used to define continuous functions, the only ones that are computable), and the *typing order* $\prec$ (used to define the order ideals). Both orderings relate with the structure of the terms. For instance, the term $(\bot, \bot)$ is a computational approximation of any other pair —$(\bot,\bot) \sqsubseteq (x,y)$—, or the term (nat, nat) is the type of any pair of natural numbers —if $1 \prec nat$ and $2 \prec nat$ then $(1,2) \prec (nat, nat)$—. It means that there is some kind of relationship between both orderings. However, if we do not want to distinguish between both orders,[3] the most sensible thing would be defining

$$x \prec y \text{ iff } y \sqsubseteq x$$

For instance, 1 has type *nat*, written $1 \prec nat$ iff *nat* is an approximation to 1, written $nat \sqsubseteq 1$. Then, $\prec$-order ideals, used to give semantics to types, become $\sqsubseteq$-order filters. In other words, if a term t has type τ (that is, $t \in \tau$ where τ is a $\prec$-order ideal), then any term u more accurate than t has (at least) the same type (also satisfies $u \in \tau$). Therefore, τ is a $\sqsubseteq$-order filter, defined as follows.

Definition 3.32 *Given a cpo $(D, \sqsubseteq_D)$, a subset $S \subseteq D$ is said to be a* **filter** *if*

(i) *$S \neq D$ and*

(ii) *whenever $x \sqsubseteq_D y$ and $x \in I$ we have $y \in I$.*

A filter $S \subseteq D$ is said to be a **open filter** *if in addition*

(iii) *for every increasing sequence $\{x_i\}_{i\in\mathbb{N}}$ if $\bigsqcup_{i\in\mathbb{N}} x_i \in S$, then there exists a $i \in \mathbb{N}$ such that $x_i \in S$.*

[3]Another possibility would be to define both order relations separately. However, —we think– such option makes the theory unnecessarily complicate.

Notice that a set $S \subseteq D$ is a (open) filter if, and only if, $D \setminus S$ is a (closed) ideal.

Given a pair of filters $S, S' \subseteq D$, we define the computational order relation $S \sqsubseteq_{\mathcal{S}(D)} S'$ by $S \supseteq S'$. Therefore,

$$S \sqsubseteq_{\mathcal{S}(D)} S' \text{ iff } D \setminus S \sqsubseteq_{\mathcal{J}(D)} D \setminus S'$$

The set of filters of a cpo is also a cpo where

$$\begin{array}{rcl} \bot_{\mathcal{S}(D)} & = & D \setminus \{\bot\} \\ \bigsqcup_{i \in \mathbb{N}} S_i & = & \bigcup_{i \in \mathbb{N}} S_i \end{array}$$

The set of open filters of a domain is also a domain where

$$\begin{array}{rcl} \bot_{\mathcal{S}(D)} & = & D \setminus \{\bot\} \\ \bigsqcup_{i \in \mathbb{N}} S_i & = & \overline{\bigcup_{i \in \mathbb{N}} S_i^o} = \overline{\bigcup_{i \in \mathbb{N}} (D \setminus S_i)^o} \end{array}$$

This duality will allow to extend most of the properties of ideals to filters. In particular, it allows to define a filter functor in the category of domains.

Definition 3.33 *The function $\mathcal{S}$: **Dom** $\rightarrow$ **Dom** is defined by*

$$\begin{array}{rcl} \mathcal{S}(D) & \stackrel{def}{=} & \{S \mid S \textit{ is a open filter, with the order } \sqsubseteq_{\mathcal{S}(D)} \stackrel{def}{=} \supseteq\} \\ \mathcal{S}(f) & \stackrel{def}{=} & \lambda S : \mathcal{S}(D) \,.\, E \setminus \mathcal{J}(f)(D \setminus S) \end{array}$$

for any object D and any arrow $f : D \rightarrow E$ is a functor in the category of domains and continuous functions between domains.

Proof: The prove for the correctness of this definition is based on the following points.

(i) S is a ω-finite open filter iff $D \setminus S$ is a ω-finite closed ideal.

Let's prove the implication in one direction (in the opposite direction it is completely equivalent). Let S be a ω-finite open filter, we define $I \stackrel{def}{=} D \setminus S$. For any increasing sequence $I_1 \sqsubseteq_{\mathcal{J}(D)} I_2 \sqsubseteq_{\mathcal{J}(D)} \ldots I_i \ldots$ we can construct another increasing sequence $D \setminus I_1 \sqsubseteq_{\mathcal{S}(D)} D \setminus I_2 \sqsubseteq_{\mathcal{S}(D)} \ldots D \setminus I_i \ldots$, such that if $I \sqsubseteq_{\mathcal{J}(D)} \bigsqcup_{i \in \mathbb{N}} I_i$, then $S = D \setminus I \sqsubseteq_{\mathcal{S}(D)} \bigsqcup_{i \in \mathbb{N}} D \setminus I_i$. Notice that $\bigsqcup_{i \in \mathbb{N}} D \setminus I_i = D \setminus \bigsqcup_{i \in \mathbb{N}} I_i$, where the left hand is a lub in $\mathcal{S}(D)$ and the right one a lub in $\mathcal{J}(D)$. As far as S is ω-finite, there exists a $n \in \mathbb{N}$ such that $S \sqsubseteq_{\mathcal{S}(D)} D \setminus I_n$, and therefore, $I \sqsubseteq_{\mathcal{J}(D)} I_n$. We can conclude that I is a ω-finite closed ideal.

(ii) $(\mathcal{S}(D), \sqsubseteq_{\mathcal{S}(D)})$ is a domain.

There are denumerable many ω-finite closed ideals of a domain, therefore, there are also denumerable many ω-finite open filters.

Now we have to prove that for any open filter S there exists an increasing sequence $S_0 \sqsubseteq_{\mathcal{S}(D)} S_1 \sqsubseteq_{\mathcal{S}(D)} \ldots S_i \ldots$ of ω-finite open filters such that $S = \bigsqcup_{i \in \mathbb{N}} S_i$. Using the equivalent result for closed ideals, we know that there exists an increasing sequence $I_0 \sqsubseteq_{\mathcal{J}(D)} I_1 \sqsubseteq_{\mathcal{J}(D)} \ldots I_i \ldots$ of ω-finite closed ideals with $D \setminus S = \bigsqcup_{i \in \mathbb{N}} I_i$. The result can be extended to open filters if we take $S_i \stackrel{def}{=} D \setminus I_i$.

(iii) $\mathcal{S}(f) : \mathcal{S}(D) \to \mathcal{S}(E)$ is a continuous function for any continuous function $f : D \to E$.

Notice that for any increasing sequence of filters $S_0 \sqsubseteq_{\mathcal{S}(D)} S_1 \sqsubseteq_{\mathcal{S}(D)} \cdots S_i \cdots$, we have

$$\bigsqcup_{i \in \mathbf{N}}^{\mathcal{S}(D)} S_i = D \setminus \bigsqcup_{i \in \mathbf{N}}^{\mathcal{J}(D)} (D \setminus S_i)$$

This equality and the continuity of $\mathcal{J}(f)$ allow to prove easily the continuity of $\mathcal{S}(f)$.

(iv) The compositional properties of $\mathcal{S}$ are easily derived from the compositional properties of $\mathcal{J}$. ■

The local continuity of $\mathcal{S}$ also arises from the duality between $\mathcal{J}$ and $\mathcal{S}$.

Theorem 3.34 *The functor $\mathcal{S} : \mathbf{Dom} \to \mathbf{Dom}$ is locally continuous.*

Proof: The proof is based on the equality

$$D \setminus \bigsqcup_{n \in \mathbf{N}}^{\mathcal{J}(D)} I_n = \bigsqcup_{n \in \mathbf{N}}^{\mathcal{S}(D)} (D \setminus I_n) \qquad (3.2)$$

which relates the least upper bound of both domains $\mathcal{J}(D)$ and $\mathcal{S}(D)$. ■

Another way to introduce this functor, independently from $\mathcal{J}$, is defining:

$$\mathcal{S}(f)^R \stackrel{def}{=} \lambda S : \mathcal{S}(E) . \{x \in D \mid f(x) \in S\}$$

for any arrow $f : D \to E$. We know that each projection determines a unique embedding (and biceversa), which may be computed easily as

$$f(x) = \sqcap \{y \in E \mid f^R(y) = x\}$$

Let's prove such result.

Lemma 3.35 *Let D and E be cpos, and $f^R : E \to D$ be a projection function.*

(i) *There exists a unique function $f : D \to E$ satisfying $f^R \circ f = Id_D$ and $f \circ f^R \sqsubseteq Id_E$.*

(ii) *Moreover, such embedding is given by $f(x) = \sqcap\{y \in E \mid f^R(y) = x\}$.*

Proof:

(i) Let f_1 and f_2 be two functions satisfying such equations. Then $f_2 = f_2 \circ (f^R \circ f_1) = (f_2 \circ f^R) \circ f_1 \sqsubseteq f_1$. Conversely $f_1 \sqsubseteq f_2$.

(ii) Let $f : D \to E$ the projection corresponding to f^R. First, $f^R \circ f(x) = x$ therefore $f(x) \in \{y \in E \mid f^R(y) = x\}$. Second, for any $y \in E$ we have $f \circ f^R(y) \sqsubseteq_E y$ and if $y \in \{y \in E \mid f^R(y) = x\}$ then $f(x) = f(f^R(y)) \sqsubseteq_E y$. Concluding, if $f(x)$ belongs to such set and it is smaller than any element of the set then it is the greatest upper bound of the set. ■

Notice that previous lemma ensures that whenever f^R is a projection, that is, when its corresponding embedding exists, then it is $f(x) = \sqcap\{y \in E \mid f^R(y) = x\}$. It does not mean that $f(x)$ will be always an embedding for any continuous function f^R, even when $\sqcap S$ exists for any set S.

We can use previous lemma to find a more practical definition of $\mathcal{S}(f)$.

Theorem 3.36 *The $\mathcal{S}$ functor, defined in 3.33, satisfies*

$$\begin{array}{lcl} \mathcal{S}(f) & = & \lambda S : \mathcal{S}(D) . \overline{\{y \in E^o \mid \forall x \in D . y \sqsubseteq_B f(x) \Rightarrow x \in S\}} \\ \mathcal{S}(f)^R & = & \lambda S : \mathcal{S}(E) . \{x \in D \mid f(x) \in S\} \end{array}$$

for any embedding $f : D \to E$.

Proof: First, we prove that it satisfies the second equality.

$$\begin{array}{lcl} \mathcal{S}(f)^R & = & \mathcal{S}(f^R) = \lambda S : \mathcal{S}(E) . D \setminus \mathcal{J}(f^R)(E \setminus S) \\ & = & \lambda S : \mathcal{S}(E) . D \setminus \{x \in D \mid f(x) \in E \setminus S\} \\ & = & \lambda S : \mathcal{S}(E) . \{x \in D \mid f(x) \in S\} \end{array}$$

Therefore, the function defined by such expression is a projection and we can apply lemma 3.35.

$$\begin{array}{lcl} \mathcal{S}(f) & = & \lambda S : \mathcal{S}(D) . \sqcap \{Y \in \mathcal{S}(E) \mid \mathcal{S}(f)^R(Y) = X\} \\ & = & \lambda S : \mathcal{S}(D) . \bigcup\{Y \in \mathcal{S}(E) \mid \forall x \in D . x \in S \Leftrightarrow f(x) \in Y\} \end{array}$$

Such expression can be simplified. For that, we will compute the glb of such set of filters.

First, we prove that $Z \overset{def}{=} \{y \in E \mid \forall x \in D . y \sqsubseteq_B f(x) \Rightarrow x \in S\}$ satisfies $\forall x \in D . x \in S \Leftrightarrow f(x) \in Z$. We have to prove

$$\forall x \in D . x \in S \Leftrightarrow (\forall x' \in D . f(x) \sqsubseteq_B f(x') \Rightarrow x' \in S)$$

There are to cases:

$\Leftarrow$ It is trivial if we take $x' \overset{def}{=} x$.

$\Rightarrow$ We have $x \in S$ and $f(x) \sqsubseteq_B f(x')$. Now $x = f^R \circ f(x) \sqsubseteq_D f^R \circ f(x') = x'$, and therefore $x' \in S$ because S is a filter and $x \in S$.

Unfortunately, Z is a filter, but not an open filter. We have to take then $\overline{Z^o}$ which is the bigger open filter contained in Z. However, $\overline{Z^o}$ also satisfies $\forall x \in D . x \in S \Leftrightarrow f(x) \in \overline{Z^o}$. If $x \in D$ is ω-finite, it is trivial because Z and $\overline{Z^o}$ contains the same ω-finite elements and $f(x)$ is also ω-finite. If x is not ω-finite, then $x = \bigsqcup_{i \in \mathbb{N}} x_i$ where x_i are ω-finite. We can prove the result taking into account that $x \in S$ iff there exists an $i \in \mathbb{N}$ such that $x_i \in S$ (because S is an open filter) and $f(\bigsqcup_{i \in \mathbb{N}} x_i) = \bigsqcup_{i \in \mathbb{N}} f(x_i) \in \overline{Z^o}$ iff there exists an $i \in \mathbb{N}$ such that $f(x_i) \in \overline{Z^o}$ (because $\overline{Z^o}$ is also an open filter).

Second, we prove that any open filter Y satisfying $\forall x \in D . x \in S \Leftrightarrow f(x) \in Y$ satisfies also $Y \subseteq Z$, and consequently $Y \subseteq \overline{Z^o}$. Let be $y \in Y$. Then, for any $x \in E$ if $y \sqsubseteq_B f(x)$ then $f(x) \in Y$ (because Y is a filter) and $x \in S$ (by definition of Y). Therefore $y \in Z$ which proves the inclusion. ■

The domains $\mathcal{J}(D)$ and $\mathcal{S}(D)$ are in fact isomorphic. We can define an isomorphism $\mathcal{C}_D : \mathcal{J}(D) \to \mathcal{S}(D)$ by $\mathcal{C}_D(S) = D \setminus S$ for any domain D. We have $\mathcal{C}_D^{-1} = \mathcal{C}_D$, and the equation (3.2) proves the continuity of $\mathcal{C}_D$, therefore $\mathcal{C}_D$ is an embedding. The existence of such a isomorphism will be very helpful to prove some results in next section.

Example. Another possibility would be defining $\mathcal{S}(f)(S)$ for any arrow $f : D \to E$ and any open filter $S \in \mathcal{S}(D)$ as the minimum open filter containing $\{f(x) \mid x \in S\}$. This set is not uniquely defined. If $f(x)$ is not an ω-finite value of E then in order to be open, $\mathcal{S}(f)(S)$ has to contain an smaller ω-finite value, which is not determined. We can define then $\mathcal{S}(f)(S)$ as the minimum open filter containing $\{f(x) \mid x \in S \wedge f(x) \text{ is } \omega\text{-finite}\}$, that is:

$$\mathcal{S}'(f) = \lambda S : \mathcal{S}(D) \,.\, \overline{\{y \in E^o \mid \exists x \in S \,.\, f(x) \sqsubseteq_E y\}}$$

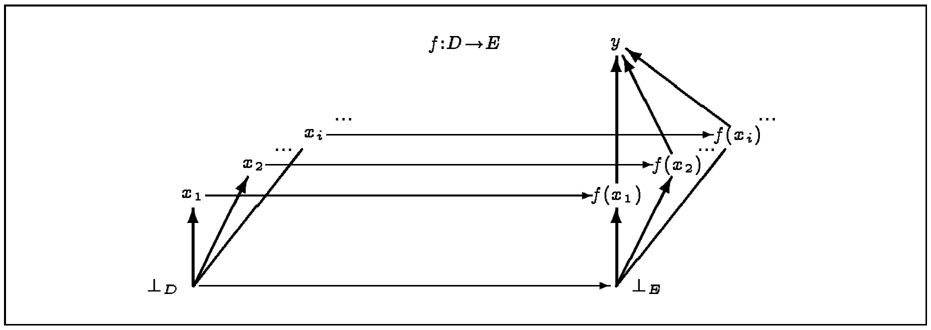


Figure 3.2: A counter-example for the alternative definition of $\mathcal{S}$.

However, it does not works because $\mathcal{S}(f)$ is not a continuous function. Let D, E and $f : D \to E$ be the two domains and the morphism shown in figure 3.2. If we take the increasing sequence of open filters $S_i \overset{def}{=} \{x_i, x_{i+1}, \ldots\}$, with the previous $\mathcal{S}'$ definition we have $\mathcal{S}'(f)(\bigsqcup_{i\in\mathbb{N}} S_i) = \emptyset$ whereas $\bigsqcup_{i\in\mathbb{N}} \mathcal{S}'(f)(S_i) = \{y\}$. Using the correct definition we obtain $\mathcal{S}(f)(\bigsqcup_{i\in\mathbb{N}} S_i) = \bigsqcup_{i\in\mathbb{N}} \mathcal{S}(f)(S_i) = \{y\}$.

3.2.7 Well-Founded Domains

In the previous section we have taken open filters over a domain as the semantic domain. These open filters are 1) closed for bigger elements (w.r.t. the computational order relation $\sqsubseteq$) and 2) open for increasing sequences (if $\bigsqcup_{i\in\mathbb{N}} x_i$ belongs to the set then there exists an $n \in \mathbb{N}$ such that x_n also belongs to the set). In the introduction we have justified the use of closed $\prec$-order ideals as semantic domain, i.e. sets which are 1) closed for smaller elements (w.r.t. the structural order relation $\prec$) and 2) closed for $\prec$-increasing sequences (if x_i belongs to the set for any $i \in \mathbb{N}$ then $\bigsqcup_{i\in\mathbb{N}} x_i$ also belongs to the set). We identify the structural order relation $\prec$ with the inverse of the computational order relation $\sqsubseteq$, therefore the first property of open $\sqsubseteq$-filters is equivalent to the first property of closed $\prec$-ideals. However, the second properties of both definitions are not equivalent. The second condition of open $\sqsubseteq$-filters is necessary in order to obtain a domain; notice that the set of (not necessarily open) filters of a domain is not, in general, a domain. Now, we introduce a new condition for domains that we will use to prove some of the following theorems, in particular lemma 3.46. This condition ensures that any element of an open filter has a minimal in the filter (filters are minimal complete), however this condition is still not enough for ensuring that any decreasing sequence in a filter has its glb in the filter, i.e. the second property of closed $\prec$-order ideals.

Definition 3.37 *A domain $(D, \sqsubseteq_D)$ is said to be well-founded if the relation $\sqsubseteq_D$ defines a well-founded order over the set of ω-finite elements of D; i.e. there does not exist any infinite strictly decreasing sequence $x_1 \sqsupset x_2 \sqsupset \cdots x_n \cdots$ of ω-finite elements.*

Lemma 3.38 *Every open filter over a well-founded domain is minimally complete.[4] Moreover, any minimal element is ω-finite.*

Proof: First, we proof for any element of the filter that either it is minimal or there exists another strictly smaller ω-finite element in the filter. If an element is not minimal then there exists another element strictly smaller than it in the filter. Let x be such element. Using lemma 3.17 we can ensure that there exists an increasing sequence $\{x_i\}_{i\in\mathbb{N}}$ of ω-finite elements with x as least upper bound. Now, if the filter is open then at least one of the ω-finite elements x_n belongs to the filter, and $x_n \sqsubseteq x$. Second, if such strictly smaller ω-finite element is not minimal in the filter, then we can repeat the same reasoning. This process can not be repeated indefinitely, otherwise we would construct an infinite strictly decreasing sequence of ω-finite elements. We conclude that any element of the filter has a minimal element below it. Finally, we prove that any minimal element x of the filter is ω-finite. Otherwise, there would be an increasing sequence $\{x_i\}_{i\in\mathbb{N}}$ of ω-finite elements with $\bigsqcup_{i\in\mathbb{N}} x_i = x$ and $x_j \neq x$ for any $j \in \mathbb{N}$ (notice that x_j is ω-finite and x is not). As far as the filter is open, there exists an $n \in \mathbb{N}$ such that x_n belongs to the filter and x will be not minimal in the filter. ■

Notice that in the previous proof we need filters to be open. Otherwise, we would need the finiteness of every strictly decreasing sequence, not only of every sequence of ω-finite elements.

From now on, we will work in the category of well-founded domains and continuous functions on them. This simplifies some proofs. For instance, to prove that a filter is included into another filter it is enough to prove that any minimum ω-finite element of the first one belongs to the second one. However, we have to prove that the functors used to construct the reflexive domain preserve the well-foundation property.

[4] See definition 3.21.

Lemma 3.39 *The functors $+$, $\times$, $\rightarrow$ $\mathcal{J}$ and $\mathcal{S}$ map well-founded domains to well-founded domains.*

Proof: For $+$ and $\times$ the proof is rather simple. For the functors $\mathcal{J}$ and $\mathcal{S}$ it is based on the well-foundation property of finite-multiset[5] orderings. Let $\leq_D$ be an order relation on D. We define an order relation $\leq_{\mathcal{P}(D)}$ on the finite subsets of D by $S_1 \leq_{\mathcal{P}(D)} S_2$ if $\forall x \in S_1 . \exists y \in S_2 . x \leq_D y$. Then, as a consequence of König's lemma (Dershowitz and Manna, 1979) if $\leq_D$ is well-founded then so it is $\leq_{\mathcal{P}(D)}$. Now, let I_1 and I_2 be two ω-finite ideals over a well-founded domain $(D, \sqsubseteq_D)$, and let M_1 and M_2 be their sets of maximal elements. As far as I_1 and I_2 are ω-finite, M_1 and M_2 are finite and only contain ω-finite elements. It is not difficult to prove that $I_1 \sqsubseteq_{\mathcal{J}(D)} I_2$ iff $M_1 \sqsubseteq_{\mathcal{P}(D)} M_2$, being $\sqsubseteq_{\mathcal{P}(D)}$ the finite-set ordering induced by $\sqsubseteq_D$. The well-foundation property for $\sqsubseteq_{\mathcal{P}(D)}$ proves then that $\mathcal{J}(D)$ is also a well-founded domain.

Any decreasing sequence $S_1 \sqsupseteq_{\mathcal{S}(D)} S_2 \sqsupseteq_{\mathcal{S}(D)} \cdots S_n \cdots$ in $\mathcal{S}(D)$ has associated a decreasing sequence $D \setminus S_1 \sqsupseteq_{\mathcal{J}(D)} D \setminus S_2 \sqsupseteq_{\mathcal{S}(D)} \cdots D \setminus S_n \cdots$ in $\mathcal{J}(D)$. It allows to conclude that if $\sqsubseteq_{\mathcal{J}(D)}$ is well-founded, then so it is $\sqsubseteq_{\mathcal{S}(D)}$. ■

Lemma 3.40 *Every ordered[6] set S of a maximal complete cpo has a greatest lower bound $\sqcap S$.*

Proof: We define the set $X \overset{def}{=} \{x \in D \mid \forall y \in S . x \sqsubseteq_D y\}$. This set is nonempty (it contains $\perp_D$) and bounded (any element of S is an upper bound), therefore, if D is maximal complete then X has a least upper bound. Such least upper bound of X is a greater lower bound of S. ■

3.3 Type Domain Construction

We have motivated in the introduction the adequacy of using a semantic domain (type domain) consisting on the set of $\prec$-order ideals of another set of values (value domain). Considerations of subsection 3.2.6 about the *computational* ($\sqsubseteq$) and the *structural* ($\prec$) orderings suggest the use of $\sqsubseteq$-order filters instead of order ideals, i.e. the identification of $\prec$ and $\sqsupseteq$. We define a value domain U as the initial fixed point of a recursive domain equation $F_\nu(D) \cong D$, where the endofunctor F_ν may be defined by $F_\nu(D) = C + D \times D + [D \rightarrow D]$. The existence and uniqueness (up to isomorphism) of such domain U is proved using standard techniques described in subsection 3.2.1. The isomorphism mapping $F_\nu(U)$ to U will be noted by $\alpha : F_\nu(U) \rightarrow U$. This is enough for giving semantics to values. For types, we will use the semantic domain $\mathcal{S}(U)$. If we use value constructors (cartesian product and functional space) and lattice constructors (union, intersection, top and bottom) as type constructors, then it seem natural to require $\mathcal{S}(U)$ to be also an F_ν-algebra. That is, to prove the existence of an

[5] The following result was proved for finite multisets. However, for our purposes it is enough to work with finite sets.

[6] A subset S of a cpo D is said to be ordered if for any pair of elements $x, y \in S$ we have either $x \sqsubseteq_D y$ or $y \sqsubseteq_D x$.

embedding $F_\nu(\mathcal{S}(U)) \lhd \mathcal{S}(U)$. Nevertheless, as we will see in this section, this is not reasonable (even not possible) to use the same endofunctor F for types and for values. Main reasons are:

1. The pairing constructor (for values) and the cartesian product (for types) are not equivalent. The product constructor only satisfies the inclusion $S \subseteq \mathit{proj}_1(S) \times \mathit{proj}_2(S)$, but not the inclusion in the opposite direction, which would be needed to prove the universal property for such construction (see subsection 3.3.1).

2. A value may be either a constant, a pair of values or a function, but not two of them simultaneously. This motivates the use of coalesced sum $+$ in the definition of F_ν. However, a type may be composed by more than one type of values, thus it has a component consisting on a set of constants, other consisting on a set of pairs of values and another of functions. It motivates the use of the $\times$ constructor, instead of $+$, in the definition of F_τ (see subsection 3.3.2).

The last reason motivates the following definition for the endofunctor F_τ.

$$F_\tau(D) \overset{def}{=} \mathcal{S}(K) \times [D \times D] \times [D \to D] \tag{3.3}$$

In the following we prove that $\mathcal{S}(U)$ is a F_τ-algebra, i.e. the existence of an embedding

$$\beta : F_\tau(\mathcal{S}(U)) \lhd \mathcal{S}(U)$$

The basic idea is defining an embedding $\mathsf{Code}_\mathcal{C} : \mathcal{C}(\mathcal{S}(D)) \to \mathcal{S}(\mathcal{C}(D))$ for each type constructor $\mathcal{C}$. It would allow to define an embedding $\mathsf{Code}_{F_\tau} : F_\tau(\mathcal{S}(D)) \to \mathcal{S}(F(D))$, which composed with $\mathcal{S}(\alpha) : \mathcal{S}(F(D)) \to \mathcal{S}(D)$ results on the desired embedding. As we will see such embedding is given by

$$\beta = \mathcal{S}(\alpha) \circ \mathsf{Code}_+ \circ (Id_{\mathcal{S}(K)} \times \mathsf{Code}_\times \times \mathsf{Code}_\to)$$

where $\alpha : F(U) \to U$ is the mediating morphism of $F(D) \cong D$, and $\mathsf{Code}_\times$, Code_+ and $\mathsf{Code}_\to$ are defined in the following subsections.

This technique may be compared with Scott's work (Scott, 1976; Gunter and Scott, 1990). Thus, $\mathcal{S}(U)$ may be considered as an universal domain, and $\mathsf{Code}_\mathcal{C}$ functions would be the embeddings used to prove representativeness of the operator $\mathcal{C}$.

3.3.1 The Embedding $\mathsf{Code}_\times : \mathcal{S}(D) \times \mathcal{S}(E) \lhd \mathcal{S}(D \times E)$

We will use a pairing constructor $\times$ for types, together with its corresponding projections proj_1 and proj_2. Such constructor is *represented* (in the sense of Scott) as a set of pairs of values. The *codification embedding* is the cartesian product defined as follows.

Definition-lemma 3.41 *The function* $\mathsf{Code}_\times : \mathcal{S}(D) \times \mathcal{S}(E) \to \mathcal{S}(D \times E)$ *defined by*

$$\mathsf{Code}_\times \langle S_1, S_2 \rangle \stackrel{def}{=} S_1 \times S_2$$

for any $S_1 \in \mathcal{S}(D)$, $S_2 \in \mathcal{S}(E)$ *is an embedding.*
Its corresponding projection satisfies

$$\mathsf{Code}_\times^R(S) = \langle \mathcal{S}(proj_1)(S), \mathcal{S}(proj_2)(S) \rangle$$

where $\mathcal{S}(proj_i)(S) = \{proj_i(x) \mid x \in S\}$, *for any* $S \in \mathcal{S}(D \times E)$.

Proof: It has to be proved that such equalities define an embedding between domains. Continuity of both functions is a straightforward exercise. Compositional properties are proved by

$$\begin{aligned}
\mathsf{Code}_\times^R \circ \mathsf{Code}_\times (\langle S_1, S_2 \rangle) &= \langle \{proj_1(x) \mid x \in S_1 \times S_2\}, \{proj_2(x) \mid x \in S_1 \times S_2\} \rangle \\
&= \langle S_1, S_2 \rangle \\
\mathsf{Code}_\times \circ \mathsf{Code}_\times^R (S) &= \{\langle x, y \rangle \mid \exists s \in S \,.\, x = proj_1(s) \wedge \exists s' \in S \,.\, y = proj_2(s')\} \\
&\supseteq S
\end{aligned}$$

■

Notice that such embedding satisfies $\mathcal{S}(proj_i) \circ \mathsf{Code}_\times \langle X_1, X_2 \rangle = X_i$ but only the inclusion $\mathsf{Code}_\times \langle \mathcal{S}(proj_1)(S), \mathcal{S}(proj_2)(S) \rangle \supseteq S$ holds in general. Therefore, $\mathsf{Code}_\times$ is not a proper pairing function because the universal property $\forall X \,.\, \langle proj_1(X), proj_2(X) \rangle = X$ does not hold.

The codification function $\mathsf{Code}_\times$ and the embedding $\beta : F_\tau(\mathcal{S}(U)) \lhd \mathcal{S}(U)$ allow to define three interpretation functions which will be used to give semantics to the pairing $\times$ and the projection $proj_i$ constructors.

Definition 3.42 *The* **interpretation functions** *for products and projections are defined as follows*

$$\begin{array}{llcl}
\mathsf{Inter}_\times : & \mathcal{S}(U) \times \mathcal{S}(U) & \to & \mathcal{S}(U) \\
& P & \mapsto & \beta(\langle \emptyset, P, \emptyset \rangle) = \mathcal{S}(\alpha \circ in_2) \circ \mathsf{Code}_\times(P) \\
\mathsf{Inter}_{proj_i} : & \mathcal{S}(U) & \to & \mathcal{S}(U) \\
& S & \mapsto & proj_i \circ proj_2 \circ \beta^R(S) = \mathcal{S}(proj_i) \circ \mathcal{S}(out_2 \circ \alpha)
\end{array}$$

They satisfy the following inequalities.

Lemma 3.43 *Functions* $\mathsf{Inter}_\times$ *and* Inter_{proj_i} *satisfy:*

$$\begin{array}{l}
\mathsf{Inter}_{proj_i}(\mathsf{Inter}_\times \langle S_1, S_2 \rangle) = S_i \\
\mathsf{Inter}_\times \langle \mathsf{Inter}_{proj_1}(S), \mathsf{Inter}_{proj_2}(S) \rangle \sqsubseteq_{\mathcal{S}(U)} S
\end{array}$$

3.3.2 The Isomorphism $\mathsf{Code}_+ : \mathcal{S}(D) \times \mathcal{S}(E) \cong \mathcal{S}(D+E)$

Definition of an embedding $\mathsf{Code}_+ : \mathcal{S}(D) + \mathcal{S}(E) \lhd \mathcal{S}(D+E)$ is not possible, as the following example shows.

Example. The function $\mathsf{Code}_+ : \mathcal{S}(D) + \mathcal{S}(E) \to \mathcal{S}(D+E)$ defined by

$$\mathsf{Code}_+(X) \overset{def}{=} \begin{cases} if \quad X = \bot_{\mathcal{S}(D)+\mathcal{S}(E)} & then \quad D+E \setminus \{\bot_{D+E}\} \\ if \quad is_1(X) & then \quad \{in_1(y) \mid y \in out_1(X)\} \\ if \quad is_2(X) & then \quad \{in_2(y) \mid y \in out_2(X)\} \end{cases}$$

for any $X \in \mathcal{S}(D)+\mathcal{S}(E)$, is continuous and injective, but it is not an embedding.

The only monotonic function $\mathsf{Code}_+^R : \mathcal{S}(D+E) \to \mathcal{S}(D) + \mathcal{S}(E)$ satisfying $\mathsf{Code}_+^R \circ \mathsf{Code}_+ = Id_{\mathcal{S}(D)+\mathcal{S}(E)}$ is defined by

$$\mathsf{Code}_+^R(S) = \begin{cases} if \quad \forall x \in S \,.\, is_1(x) & then \quad in_1(\{out_1(x) \mid x \in S\}) \\ if \quad \forall x \in S \,.\, is_2(x) & then \quad in_2(\{out_2(x) \mid x \in S\}) \\ otherwise & \qquad \bot_{\mathcal{S}(D)+\mathcal{S}(E)} \end{cases}$$

for any $S \in \mathcal{S}(D+E)$. However, such function is not continuous as the following example shows.

Consider an infinite sequence of values $d_i \in D$, a value $e \in E$ and the increasing sequence of filters $S_i \in \mathcal{S}(D+E)$ defined by $S_i = \{in_2(e), in_1(d_i), in_1(d_{i+1}), \ldots\}$ and satisfying $\sqcup S_i = \{in_2(e)\}$. It is easy to prove that $\mathsf{Code}_+^R(S_i) = \bot_{\mathcal{S}(D)+\mathcal{S}(E)}$ for any $i \in \mathbb{N}$, whereas $\mathsf{Code}_+^R(\bigsqcup_{i\in\mathbb{N}} S_i) = in_2(\{e\})$. Therefore $\mathsf{Code}_+^R(\bigsqcup_{i\in\mathbb{N}} S_i) \neq \bigsqcup_{i\in\mathbb{N}} \mathsf{Code}_+^R(S_i)$.

As we have mentioned, coalesced sum is not adequate to put together the different kinds of types. A value has a unique kind, whereas a type may be composed by different kinds of values. Thus, it is more sensible to decompose a type —an element of $\mathcal{S}(D+E)$— into its different components —an element of $\mathcal{S}(D) \times \mathcal{S}(E)$—, instead of classifying it into two kinds of types —an element of $\mathcal{S}(D) + \mathcal{S}(E)$—. We see then that a type is uniquely characterized by its components. It means that we can define an isomorphism between the universe of *mixed types* $\mathcal{S}(D+E)$ and the product of *pure types* $\mathcal{S}(D) \times \mathcal{S}(E)$.

Definition-lemma 3.44 *The continuous function* $\mathsf{Code}_+ : \mathcal{S}(D) \times \mathcal{S}(E) \to \mathcal{S}(D+E)$ *defined by*

$$\mathsf{Code}_+\langle S_1, S_2\rangle \overset{def}{=} \{in_1(y) \mid y \in S_1\} \cup \{in_2(y) \mid y \in S_2\}$$

is an isomorphism. Its inverse is defined by

$$\mathsf{Code}_+^{-1}(S) = \langle \{out_1(x) \mid x \in S \wedge is_1(x)\}, \{out_2(x) \mid x \in S \wedge is_2(x)\}\rangle$$

for any $S_1 \in \mathcal{S}(D)$, $S_2 \in \mathcal{S}(E)$ *and* $S \in \mathcal{S}(D+E)$.

3.3.3 The Embedding $\mathsf{Code}_{\rightarrow} : \mathcal{S}(D) \rightarrow \mathcal{S}(E) \lhd \mathcal{S}(D \rightarrow E)$

Contrary to previous cases, definition of an embedding from $\mathcal{S}(D) \rightarrow \mathcal{S}(E)$ to $\mathcal{S}(D \rightarrow E)$ is not so easy as it could seem. Thus, many simple definitions fail when we try to prove their continuity or the inclusion relations defining an embedding. There is no room here to present all unsuccessful attempts, so we have selected only a pair of them to show the complexity of the task. The definition of such codification function is one of the most important contributions of this thesis. The first one presents one of the more *aesthetic* solutions that can be proposed. The second one is based on the first one. The final solution is a slight modification of this second example.

Example. Our experience suggests us to define an embedding-projection pair proposing a projection candidate first, and using lemma 3.35 to find the corresponding embedding later. Suppose we have a singleton open filter $S = \{f\} \in \mathcal{S}(D \rightarrow E)$. If we have to choose a function $\mathsf{Code}^R_{\rightarrow}(\{f\}) : \mathcal{S}(D) \rightarrow \mathcal{S}(E)$ being codified by f, the more appropriate candidate would be $\mathcal{S}(f)$. Now, if S contains more than one function then each element belonging to S contributes to *"make* $\mathsf{Code}^R_{\rightarrow}(S)$ *more undefined"*. In other words, if $\mathsf{Code}^R_{\rightarrow}$ is a projection then it is monotonic and as much *bigger* S is (smaller w.r.t. $\sqsubseteq_{\mathcal{S}(D \rightarrow E)}$), more *undefined* $\mathsf{Code}^R_{\rightarrow}(S)$ is (smaller w.r.t. $\sqsubseteq_{\mathcal{S}(D) \rightarrow \mathcal{S}(E)}$). We could define $\mathsf{Code}^R_{\rightarrow}(S) \overset{def}{=} \mathcal{S}(\sqcap S)$. But this is not a good choice, because it is equivalent to trying to codify a function in $\mathcal{S}(D) \rightarrow \mathcal{S}(E)$ using a unique function in $D \rightarrow E$, which is clearly impossible. It is better to define:

$$\mathsf{Code}^R_{\rightarrow}(S) \overset{def}{=} \sqcap_{f \in S} \mathcal{S}(f)$$

If such function is a projection, then its corresponding embedding will be:

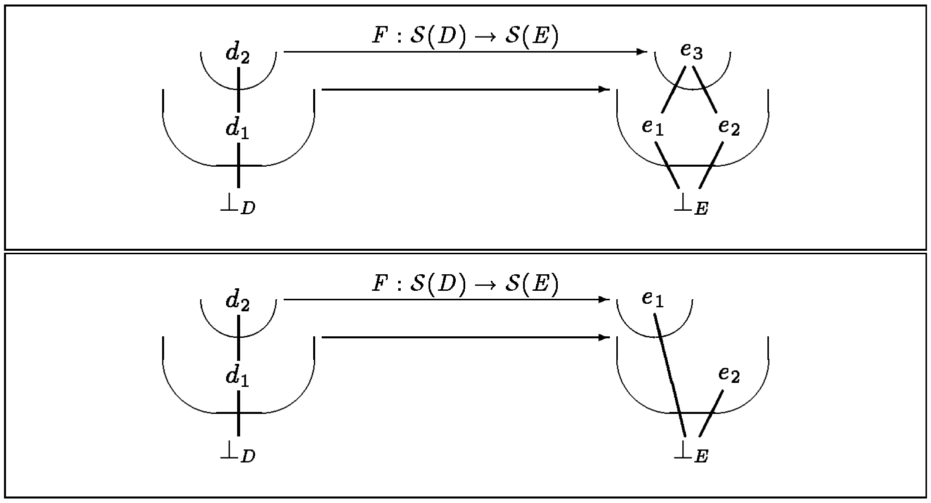
$$\mathsf{Code}_{\rightarrow}(F) = \overline{\{f \in (D \rightarrow E)^o \mid F \sqsubseteq_{\mathcal{S}(D) \rightarrow \mathcal{S}(E)} \mathcal{S}(f)\}}$$

Continuity of such functions may be easily proved, as well as the following inequalities:

$$\begin{aligned} \mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F) &= \sqcap_{F \sqsubseteq \mathcal{S}(f)} \mathcal{S}(f) \sqsupseteq F \\ \mathsf{Code}_{\rightarrow} \circ \mathsf{Code}^R_{\rightarrow}(S) &= \overline{\{f \in (D \rightarrow E)^o \mid \sqcap_{f' \in S} \mathcal{S}(f') \sqsubseteq \mathcal{S}(f)\}} \sqsubseteq S \end{aligned}$$

In fact, given a continuous function, like $\mathsf{Code}^R_{\rightarrow}$, it is always possible to find another function, like $\mathsf{Code}_{\rightarrow}$, satisfying such inequalities. Problems arise when we try to prove $\mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F) \sqsubseteq F$. This inequality only holds when for any $X \in \mathcal{S}(D)$ and any $y \in F(X)$ we can find a function $f : D \rightarrow E$ such that $F \sqsubseteq \mathcal{S}(f)$ and $y \in \mathcal{S}(f)(X)$. Unfortunately, first example of figure 3.3 shows that it is not possible to find such function for $X = \{d_2\}$ and $y = e_3$.

Example. In previous example we have chosen $\mathsf{Code}^R_{\rightarrow}(\{f\}) = \mathcal{S}(f)$. This is the smallest function satisfying $x \in X \Leftrightarrow f(x) \in \mathcal{S}(f)(X)$. We can take the biggest



$F : \mathcal{S}(D) \to \mathcal{S}(E)$
d_2
d_1
$\bot_D$
e_3
e_1
e_2
$\bot_E$
$F : \mathcal{S}(D) \to \mathcal{S}(E)$
d_2
d_1
$\bot_D$
e_1
e_2
$\bot_E$

Figure 3.3: Two counter-examples for two possible definitions of $\mathsf{Code}_{\rightarrow}$.

one of such functions. In this case, we obtain:

$$\begin{array}{rcl} \mathsf{Code}_{\rightarrow}(F) & = & \overline{\{f \in (D \rightarrow E)^o \mid \forall X \in \mathcal{S}(D) \,.\, \forall x \in X \,.\, f(x) \in F(X)\}} \\ \mathsf{Code}^R_{\rightarrow}(S) & = & \lambda X : \mathcal{S}(D) \,.\, \{f(x) \mid x \in X \ \wedge \ f \in S\} \end{array}$$

Then, inequality $\mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F) \sqsubseteq F$ only holds if for any $X \in \mathcal{S}(D)$ and any $y \in F(X)$ we can find an $f : D \rightarrow E$ such that $\forall X \in \mathcal{S}(D) \,.\, \forall x \in X \,.\, f(x) \in F(X)$ and $\exists x \in X \,.\, y = f(x)$. Second example of figure 3.3 also shows that it is not possible to find such function for $X = \{d_1, d_2\}$ and $y = e_1$.

A careful analysis shows that the function $\mathsf{Code}_{\rightarrow}$ only codifies correctly $\cup$-additive function. If $x \in X_1 \cup X_2$ and $\forall X \in \mathcal{S}(D) \,.\, x \in X \,.\, f(x) \in F(X)$ then $f(x) \in F(X_1)$ or $f(x) \in F(X_2)$. Therefore,

$$\mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)(X_1 \cup X_2) = \mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)(X_1) \cup \mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)(X_2)$$

Such problem always appears when we try to codify a function on sets using the set of functions mapping each element from a set of the domain to an element of the corresponding image set (see (Sannella et al., 1990)). We already mention such problem in (Levy et al., 1990), where we propose the definition of an encoding function from $\mathcal{S}(D) \rightarrow \mathcal{S}(E)$ to $\mathcal{S}(\mathcal{S}(D) \rightarrow E)$ (see subsection 3.3.4). The set of functions used to codify a function on sets is then larger and we get the desired embedding. However, now we know that it is possible to define a codification function being an embedding without enlarging the codification space.

The only way to avoid $\mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)$ be additive is using more functions to codify F. We can do that introducing in $\mathsf{Code}(F)$ functions satisfying $\forall x \in$

$X_1 \cup X_2 . f(x) \in F(X_1 \cup X_2)$ but neither $\forall x \in X_1 . f(x) \in F(X_1)$ nor $\forall x \in X_2 . f(x) \in F(X_2)$. We define then

$$\mathsf{Code}_{\rightarrow}(F) \overset{def}{=} \overline{\{f \in (D \rightarrow E)^o \mid \exists X \in \mathcal{S}(D)^o \setminus \{\emptyset\} . \forall x \in X . f(x) \in F(X)\}}$$

Then, $f(x) \in \mathsf{Code}^R_{\rightarrow}(S)(X)$ for any $x \in X$ and any $f \in S$ is no longer true and we have to restrict the set of functions f used to define $\mathsf{Code}^R_{\rightarrow}$. The corresponding projection is, *more or less*

$$\begin{aligned}\mathsf{Code}^R_{\rightarrow} = \lambda S : \mathcal{S}(D) . \lambda X : \mathcal{S}(D) . \{y \in D \mid \exists x \in X . \exists f \in S . \bot_E \neq f(x) \sqsubseteq_E y \\ \wedge f \text{ is minimal in } S \wedge (\forall x' \notin X . f(x') = \bot_E)\}\end{aligned}$$

Here, *more or less* means that this function is not continuous and we have to modify it slightly to obtain a continuous function. We define continuous extensions for such purpose.

Definition 3.45 *Let D and E be domains, and $f : D^o \rightarrow E$ be a monotonic function. Then, there exists a unique continuous function $f^{ext} : D \rightarrow E$, named* **continuous extension**, *satisfying $f^{ext}(x) = f(x)$ for any ω-finite element $x \in D^o$.*

Proof: Lemma 3.17 ensure that if D is a domain and $x \in D$, then either $x \in D^o$ or there exists an increasing sequence of ω-finite elements $\{x_i\}_{i \in \mathbb{N}}$ such that $x = \bigsqcup_{i \in \mathbb{N}} x_i$. Evidently, such sequence is not unique. We define f^{ext} by $f^{ext}(x) \overset{def}{=} f(x)$ if $x \in D^o$ and $f^{ext}(x) = \bigsqcup_{i \in \mathbb{N}} f(x_i)$ for one of such sequences if x is not ω-finite. However, we have to prove then the result is independent from the chosen sequence.

Let $\{x_i\}_{i \in \mathbb{N}}$ and $\{x'_j\}_{j \in \mathbb{N}}$ be two sequences satisfying $x = \bigsqcup_{i \in \mathbb{N}} x_i = \bigsqcup_{j \in \mathbb{N}} x'_i$. For any $i \in \mathbb{N}$ we have $x_i \sqsubseteq \bigsqcup_{j \in \mathbb{N}} x'_j$ and, as longer as x_i is ω-finite we have also $\forall i \in \mathbb{N} . \exists j \in \mathbb{N} . x_i \sqsubseteq x'_j$. If f is monotonic, then $\forall i \in \mathbb{N} . \exists j \in \mathbb{N} . f(x_i) \sqsubseteq f(x'_j)$ and we conclude $\bigsqcup_{i \in \mathbb{N}} f(x_i) \sqsubseteq \bigsqcup_{j \in \mathbb{N}} f(x'_j)$. Similarly we prove $\bigsqcup_{i \in \mathbb{N}} f(x_i) = \bigsqcup_{j \in \mathbb{N}} f(x'_j)$.

Now, we prove the continuity of the function defined in this way. (Notice that the function $f : D^o \rightarrow E$ is continuous if, and only if, it is monotonic). Let $\{x_i\}_{i \in \mathbb{N}}$ be an increasing sequence in D. For any $i \in \mathbb{N}$ we can find an increasing sequence $\{p^i_j\}_{j \in \mathbb{N}}$ such that $x_i = \bigsqcup_{j \in \mathbb{N}} p^i_j$. It is not difficult to prove that $\forall i \in \mathbb{N} . \exists k \in \mathbb{N} . x_i \sqsubseteq p^{i+1}_k$ using the ω-finiteness definition. Therefore, we can define a sequence of k_i such that $p^1_{k_1} \sqsubseteq p^2_{k_2} \sqsubseteq \dots$ is an increasing sequence satisfying $\bigsqcup_{i \in \mathbb{N}} x_i = \bigsqcup_{i \in \mathbb{N}} p^i_{k_i}$. Now, using the definition of f^{ext} we have $f^{ext}(\bigsqcup_{i \in \mathbb{N}} x_i) = f^{ext}(\bigsqcup_{i \in \mathbb{N}} p^i_{k_i}) = \bigsqcup_{i \in \mathbb{N}} f^{ext}(p^i_{k_i})$ and on the other hand $\bigsqcup_{i \in \mathbb{N}} f^{ext}(x_i) = \bigsqcup_{i \in \mathbb{N}} f^{ext}(\bigsqcup_{j \in \mathbb{N}} p^i_j) = \bigsqcup_{i \in \mathbb{N}} \bigsqcup_{j \in \mathbb{N}} f^{ext}(p^i_j)$. Clearly, $f^{ext}(\bigsqcup_{i \in \mathbb{N}} x_i) = \bigsqcup_{i \in \mathbb{N}} f^{ext}(p^i_{k_i}) \sqsubseteq \bigsqcup_{i \in \mathbb{N}} \bigsqcup_{j \in \mathbb{N}} f^{ext}(p^i_j) = \bigsqcup_{i \in \mathbb{N}} f^{ext}(x_i)$. The inclusion in the opposite direction is derived from the monotonicity of f^{ext}. This concludes the proof. ■

This lemma can be extended to functions with several arguments. Thus, if $f : D^o \rightarrow E^o \rightarrow F$ is monotonous, then its continuous extension is

$(\lambda x : D^o . (\lambda y : E^o . f(x,y))^{ext})^{ext}$, which is continuous in both arguments. We can conclude also that two continuous functions are equal ($f = g$) if, and only if, they are equal for any ω-finite argument ($\forall x \in D^o . f(x) = g(x)$).

From now on, we will to take care of defining monotonic functions on ω-finite arguments, it does not mind if they are also continuous or not. Then, we will use their continuous extensions to obtain a continuous function.

Definition-lemma 3.46 *The function* $\mathsf{Code}_{\rightarrow}$ *defined by*

$$\mathsf{Code}_{\rightarrow}(F) \stackrel{def}{=} \overline{\{f \in (D \rightarrow E)^o \mid \exists X \in \mathcal{S}(D)^o \setminus \{\emptyset\} . \forall x \in X . f(x) \in F(X)\}}$$

for any $F : \mathcal{S}(D) \rightarrow \mathcal{S}(E)$ *is an embedding. Its corresponding projection is*

$$\mathsf{Code}^R_{\rightarrow} = \Big(\lambda S : \mathcal{S}(D)^o . \lambda X : \mathcal{S}(D)^o . \big\{y \in D \mid \exists x \in X . \exists f \in S . \bot_B \neq f(x) \sqsubseteq_B y \\ \wedge f \textit{ is minimal in } S \wedge (\forall x' \notin X . f(x') = \bot_B)\big\}\Big)^{ext}$$

Proof: The proof of correctness of the previous definition is based on the following points.

(i) Function $\mathsf{Code}_{\rightarrow}$ is continuous.

We prove first that $f \in \big(\bigsqcup_{i\in\mathbb{N}} \mathsf{Code}_{\rightarrow}(F_i)\big)^o$ implies $f \in \big(\mathsf{Code}_{\rightarrow}(\bigsqcup_{i\in\mathbb{N}} F_i)\big)^o$. If $f \in \big(\bigsqcup_{i\in\mathbb{N}} \mathsf{Code}_{\rightarrow}(F_i)\big)^o$, then for any $i \in \mathbb{N}$ there exists an $X \in \mathcal{S}(D)^o \setminus \{\emptyset\}$ such that $\forall x \in X . f(x) \in F_i(X)$. Let X_i be the maximum[7] filter satisfying such proposition for each $i \in \mathbb{N}$, then it is easy to prove that the sequences $\{X_i\}_{i\in\mathbb{N}}$ and $\{F_i(X_i)\}_{i\in\mathbb{N}}$ are both increasing. Now, if f is ω-finite, then its set of images is a *finite* set of ω-finite elements.Therefore, if each filter $F_i(X_i)$ contains at least one of such images, then $\bigcup_{i\in\mathbb{N}} F_i(X_i)^o$ will contain also at least one of them. Let p be one of them. It is not difficult to prove that the set $X \stackrel{def}{=} \{x \in D \mid p \sqsubseteq f(x)\}$ is a nonempty open filter, and it satisfies $\forall x \in X . f(x) \in \bigcup_{i\in\mathbb{N}} F_i(X_i)^o$. As far as F_i are continuous functions we have $\bigcup_{i\in\mathbb{N}} (F_i(X_i))^o = \bigcup_{i\in\mathbb{N}} \bigcup_{j\in\mathbb{N}} (F_i(X_j))^o = \bigcup_{i\in\mathbb{N}} (F_i(\overline{\bigcup_{j\in\mathbb{N}} X_j^o}))^o = (\bigsqcup_{i\in\mathbb{N}} F_i(X))^o$. Therefore, we conclude that $f \in \big(\mathsf{Code}_{\rightarrow}(\bigsqcup_{i\in\mathbb{N}} F_i)\big)^o$. Implication in the other direction is ensured by the monotonicity of $\mathsf{Code}_{\rightarrow}$. We have then $\big(\bigsqcup_{i\in\mathbb{N}} \mathsf{Code}_{\rightarrow}(F_i)\big)^o = \big(\mathsf{Code}_{\rightarrow}(\bigsqcup_{i\in\mathbb{N}} F_i)\big)^o$ which proves the continuity of $\mathsf{Code}_{\rightarrow}$.

(ii) Continuity of $\mathsf{Code}^R_{\rightarrow}$ is ensured by lemma 3.45.

(iii) $\mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow} = \mathit{Id}$.

First, we prove that if $p \in \mathsf{Code}^R_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)(X)$ then $p \in F(X)$, where we can suppose without loose of generality[8] that $X \in (\mathcal{S}(D))^o$ and $p \in E^o$ are both

[7]Notice that the union of filters satisfying such proposition also satisfies it, therefore the maximum filter exists.

[8]Notice that two continuous functions are equal iff they are equal for any ω-finite element and that two closed ideals are equal iff they contains the same minimal ω-finite elements.

ω-finite and p is minimal. Therefore, there exists a function $f \in \mathsf{Code}_{\rightarrow}(F)$ and a $x \in X$ such that

$$p = f(x) \neq \perp_{\mathcal{B}} \tag{3.4}$$

$$\forall x' \notin X \,.\, f(x') = \perp_{\mathcal{B}} \tag{3.5}$$

$$f \text{ is minimal in } \mathsf{Code}_{\rightarrow}(F) \tag{3.6}$$

We can suppose without lose of generality that $f \in \mathsf{Code}_{\rightarrow}(F)$ is ω-finite (because it is minimal) therefore there exists a nonempty set $X' \in \mathcal{S}(D)^o$ such that

$$\forall x' \in X' \,.\, f(x') \in F(X') \tag{3.7}$$

It is easy to prove that the condition (3.6) ensures

$$\forall x' \notin X' \,.\, f(x') = \perp_{\mathcal{B}} \tag{3.8}$$

otherwise, the function f' defined by $f'(x) \overset{def}{=} f(x)$ if $x \in X'$ and by $f'(x) \overset{def}{=} \perp_{\mathcal{B}}$ otherwise, belonging to $\mathsf{Code}_{\rightarrow}(F)$, would be smaller than f. Now, as $f(x) = p \neq \perp_{\mathcal{B}}$ condition (3.8) ensures $x \in X'$ and, using (3.7), $p = f(x) \in F(X')$. On the other hand, for any $x' \notin X$ the condition (3.5) ensures $f(x') = \perp_{\mathcal{B}} \notin F(X')$ and, using (3.7), $x' \notin X'$. From that we conclude $X' \subseteq X$ and $F(X') \subseteq F(X)$. Those two fact prove $p = f(x) \in F(X') \subseteq F(X)$ which finishes the proof.

Second, we prove that if $p \in F(X)$ then $p \in \mathsf{Code}^{R}_{\rightarrow} \circ \mathsf{Code}_{\rightarrow}(F)(X)$, where we can also suppose without loose of generality that $X \in \mathcal{S}(D)^o$ and $p \in F(X)^o$ are both ω-finite and p is minimal. We define $X_{min} \subseteq X$ as being a filter such that $p \in F(X_{min})$ and and no other filter contained in X_{min} satisfies such property.[9] We use such filter to define the following function

$$f(x) \overset{def}{=} \begin{cases} p & \mathit{if}\ x \in X_{min} \\ \perp_{\mathcal{B}} & \mathit{otherwise} \end{cases}$$

This function satisfies evidently $f \in \mathsf{Code}_{\rightarrow}(F)$ and $f(x) = p$ for some $x \in X$. As $X_{min} \subseteq X$ we have also $\forall x' \notin X \,.\, f(x') = \perp_{\mathcal{B}}$. Finally the minimallity of f is ensured by the fact that there is not any filter X' such that $X' \subset X_{min}$ and $p \in F(X')$ and p is minimal in $F(X)$ and therefore in $F(X_{min})$.

(iv) $\mathsf{Code}_{\rightarrow} \circ \mathsf{Code}^{R}_{\rightarrow}(S) \supseteq S$.

We prove that if $f \in S$ then $f \in \mathsf{Code}_{\rightarrow} \circ \mathsf{Code}^{R}_{\rightarrow}(S)$, where we can suppose without loose of generality[10] that $f \in S^o$ and f is minimum in S. First, we will that the set

$$\mathcal{D}om(f) \overset{def}{=} \{x \in D \mid f(x) \neq \perp_{\mathcal{B}}\}$$

[9] The Kuratowski-Zorn theorem ensures the existence of at least one of such filters. We can construct a maximal ordered set (i.e. a set where any pair of elements are comparable and no other element can be added) of filters, containing the filter X and satisfying $p \in F(X)$. The least upper bound X_{min} of such maximal set exists and also satisfies $p \in F(X_{min})$ (because F is continuous for lub of ordered sets). As the ordered set is maximal, there will not be any set bigger than (contained in) X_{min} and satisfying $p \in F(X)$.

[10] Again, a filter is included into another one if the set of minimal elements of the first one is included into the second one.

is a nonempty open filter. Suppose that $x \in \mathcal{Dom}(f)$ and $x \sqsubseteq_D x'$, then $\perp_B \neq f(x) \sqsubseteq_B f(x')$ and therefore $x' \in \mathcal{Dom}(f)$. Suppose now that $\bigsqcup_{i\in\mathbb{N}} x_i \in \mathcal{Dom}(f)$, then $\bigsqcup_{i\in\mathbb{N}} f(x_i) = f(\bigsqcup_{i\in\mathbb{N}}) \neq \perp_B$ and therefore $f(x_n) \neq \perp_B$ for some $n \in \mathbb{N}$. We conclude that $\mathcal{Dom}(f)$ is an open filter. Moreover $\mathcal{Dom}(f) \neq \emptyset$, otherwise we would have $f = \perp_{D\to B}$ which contradicts $\perp_{D\to B} \notin S$ for any filter S. It can also been easily proved that if f is ω-finite function then $\mathcal{Dom}(f)$ is also an ω-finite open filter.

Then, $\forall x \notin \mathcal{Dom}(f) \,.\, f(x) = \perp_B$ and, as far as f is minimal in S, we will have $f(x) \in \mathsf{Code}^R_\to(S)(\mathcal{Dom}(f))$ for any $x \in \mathcal{Dom}(f)$. Now, as $\mathcal{Dom}(f)$ is a nonempty open filter we have $f \in \mathsf{Code}_\to \circ \mathsf{Code}^R_\to(S)$. ∎

In this case the interpretation functions are Fun and Graph. On such functions, relies the soundness of β and η-rules, as we have shown in chapter 2.

Definition 3.47 *The interpretation functions for λ-abstractions and applications are defined as follows*

$$\begin{array}{llcl}
\mathsf{Graph}: & \mathcal{S}(U)\to\mathcal{S}(U) & \to & \mathcal{S}(U) \\
 & F & \mapsto & \beta(\langle\emptyset,\emptyset,F\rangle) = \mathcal{S}(\alpha\circ in_3)\circ\mathsf{Code}_\to(F) \\
\mathsf{Fun}: & \mathcal{S}(U) & \to & \mathcal{S}(U)\to\mathcal{S}(U) \\
 & S & \mapsto & proj_3\circ\beta^R(S) = \mathsf{Code}^R_\to\circ\mathcal{S}(out_3\circ\alpha)
\end{array}$$

They satisfy the following inequalities.

Lemma 3.48 *Functions* Fun *and* Graph *satisfy:*

$$\mathsf{Fun}\circ\mathsf{Graph}(F) = F$$
$$\mathsf{Graph}\circ\mathsf{Fun}(S) \supseteq S$$

This lemma sets that our *ideal model* is a quasi-extensional COR-model.

3.3.4 The Embedding $\mathsf{Code}_\to : \mathcal{S}(D)\to\mathcal{S}(E) \lhd \mathcal{S}(\mathcal{S}(D)\to E)$

The problems to define an embedding between $\mathcal{S}(D)\to\mathcal{S}(E)$ and $\mathcal{S}(D\to E)$ can be avoided if we enlarge the codification space. The main problem in previous subsection was that the pointwise extension of a set of functions in $D\to E$ always results in a $\cup$ additive function on $\mathcal{S}(D)\to\mathcal{S}(E)$ (see second example in subsection 3.3.3). To avoid such problem we can enlarge the domain of the functions used to codify. One possible solution is defining an embedding between $\mathcal{S}(D)\to\mathcal{S}(E)$ and $\mathcal{S}(\mathcal{S}(D)\to E)$. For simplicity, we will do it in two steps. First, we will define an embedding between $\mathcal{J}(D)\to\mathcal{J}(E)$ and $\mathcal{J}(\mathcal{J}(D)\to E)$. Second, using the isomorphism $\mathcal{J}(D)\cong\mathcal{S}(D)$, we will define the embedding between $\mathcal{S}(D)\to\mathcal{S}(E)$ and $\mathcal{S}(\mathcal{S}(D)\to E)$.

Definition-lemma 3.49 *The pair of functions*

$$\begin{array}{lcccl}
\widehat{\mathsf{Code}}_\to & : & \mathcal{J}(D)\to\mathcal{J}(E) & \to & \mathcal{J}(\mathcal{J}(D)\to E) \\
\widehat{\mathsf{Code}}^R_\to & : & \mathcal{J}(\mathcal{J}(D)\to E) & \to & \mathcal{J}(D)\to\mathcal{J}(E)
\end{array}$$

defined by

$$\begin{array}{lcl}\widehat{\mathsf{Code}}_{\rightarrow}(F) & \overset{def}{=} & \{f : \mathcal{J}(D) \to E \mid \forall X \in \mathcal{J}(D) . f(X) \in F(X)\} \\ \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I) & \overset{def}{=} & \lambda X : \mathcal{J}(D) . \overline{\{f(X) \mid f \in I\}}\end{array}$$

form an embedding-projection pair.

Proof: This proof is based on the following points.

(i) $\widehat{\mathsf{Code}}_{\rightarrow}(F) \in \mathcal{J}(\mathcal{J}(D) \to E)$.

If $f \in \widehat{\mathsf{Code}}_{\rightarrow}(F)$ and $g \sqsubseteq f$, then for any ideal $X \in \mathcal{J}(U)$ we have $g(X) \sqsubseteq f(X) \in F(X)$. Now, if $F(X)$ is an ideal then $g(X) \in F(X)$, and therefore $g \in \widehat{\mathsf{Code}}_{\rightarrow}(F)$.

If for any $i \in \mathbb{N}$, $f_i \in \widehat{\mathsf{Code}}_{\rightarrow}(F)$, then for any $X \in \mathcal{J}(U)$ we have $f_i(X) \in F(X)$. Now, as $F(X)$ is closed for increasing sequences $[\sqcup_{i\in\mathbb{N}} f_i](X) = \sqcup_{i\in\mathbb{N}} f_i(X) \in F(X)$, and therefore $\sqcup_{i\in\mathbb{N}} f_i \in \widehat{\mathsf{Code}}_{\rightarrow}(F)$.

(ii) $\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X) \in \mathcal{J}(E)$.

Using the domain properties, it is enough to prove that $\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X)^{open} \overset{def}{=} \{f(X) \in E^o \mid f \in I\}$ is a (may be open) order ideal.

For any $p \in E$ we have to prove that if $p \sqsubseteq E f(X)$ and $f \in I$, then there exist a function $g : \mathcal{J}(D) \to E$ such that $g(X) = p$ and $g \in I$. Using again the domain properties, there exists a sequence $\{p_i\}_{i\in\mathbb{N}}$ of ω-algebraic values with $\sqcup_{i\in\mathbb{N}} p_i = p$. We define then

$$g \overset{def}{=} \lambda X : \mathcal{J}(D) . \bigsqcup\{p_i \mid p_i \sqsubseteq_E f(X)\}$$

which trivially satisfies $g \sqsubseteq f$ and $g(X) = p$. Consequently, $g \in I$.

The continuity of f, and the ω-finiteness of y_i prove the continuity of g:

$$\begin{array}{lcl} g(\sqcup_{j\in\mathbb{N}} X_j) & = & \bigsqcup\{y_i \mid y_i \sqsubseteq f(\sqcup_{j\in\mathbb{N}} X_j)\} = \bigsqcup\{y_i \mid y_i \sqsubseteq \sqcup_{j\in\mathbb{N}} f(X_j)\} \\ & = & \bigsqcup\{y_i \mid \exists j \in \mathbb{N} . y_i \sqsubseteq f(X_j)\} = \sqcup_{j\in\mathbb{N}} \bigsqcup\{y_i \mid y_i \sqsubseteq f(X_j)\} \\ & = & \sqcup_{j\in\mathbb{N}} g(X_j)\end{array}$$

(iii) $\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I) : \mathcal{J}(D) \to \mathcal{J}(E)$ is continuous.

$\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)$ is obviously monotonic, let's prove then $\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(\sqcup_{i\in\mathbb{N}} X_i) \subseteq \sqcup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)$.

Suppose $p \in \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(\sqcup_{i\in\mathbb{N}} X_i)^{open}$ then there exists $g \in I$ such that $p = g(\sqcup_{i\in\mathbb{N}} X_i)$. So, by the continuity of g we have $p = \sqcup_{i\in\mathbb{N}} g(X_i)$, where for any $j \in \mathbb{N}$ evidently $g(X_j) \in \mathsf{Fun}(I)(X_j) \subseteq \underline{\cup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)}$. Therefore, by the definition of closure, $p = \sqcup_{j\in\mathbb{N}} g(X_j) \in \overline{\cup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)} = \sqcup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)$. Concluding, $\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(\sqcup_{i\in\mathbb{N}} X_i) = \overline{\widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(\sqcup_{i\in\mathbb{N}} X_i)^{open}} = \overline{\sqcup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)} = \sqcup_{i\in\mathbb{N}} \widehat{\mathsf{Code}}^{R}_{\rightarrow}(I)(X_i)$.

(iv) $\widehat{\mathsf{Code}}^{R}_{\rightarrow} \circ \widehat{\mathsf{Code}}_{\rightarrow} = Id$.

The inclusion $\widehat{\mathsf{Code}}^{R}_{\rightarrow} \circ \widehat{\mathsf{Code}}_{\rightarrow}(F) \subseteq F$ is easy to prove, let's prove the inclusion $\widehat{\mathsf{Code}}^{R}_{\rightarrow} \circ \widehat{\mathsf{Code}}_{\rightarrow}(F) \supseteq F$.

We have to prove that for every $X \in \mathcal{J}(D)$ the inclusion

$$F(X) \subseteq \overline{\{f(X) \mid \exists f : \mathcal{J}(D) \to E \,.\, \forall Y \in \mathcal{J}(U) \,.\, f(Y) \in F(Y)\}}$$

holds. Let $p \in F(X)$ be any element of the first set. By lemma 3.17, there exists an increasing sequence of ω-finite elements such that $\sqcup_{i \in \mathbb{N}} p_i = p$. We define

$$f \stackrel{def}{=} \lambda X' : \mathcal{J}(D) \,.\, \bigsqcup \{p_i \mid p_i \in F(X')\}$$

Following the same reasoning than in point (ii), the continuity of F and the ω-finiteness of p_i prove the continuity of f. On the other hand, it is evident that for any $X' \in \mathcal{J}(U)$ we have $f(X') \in F(X')$. Therefore, $f \in \widehat{\mathsf{Code}}_{\to}(F)$. It is also evident that $f(X) = p$. We can conclude that $p \in \widehat{\mathsf{Code}}^R_{\to} \circ \widehat{\mathsf{Code}}_{\to}(F)(X)$.

(v) $\widehat{\mathsf{Code}}_{\to} \circ \widehat{\mathsf{Code}}^R_{\to} \sqsubseteq Id$ is easily proved. ■

Now, if $\mathcal{C}_D : \mathcal{J}(D) \to \mathcal{S}(D)$ is the isomorphism defined by $\mathcal{C}_D = \lambda I : \mathcal{J}(D) \,.\, D \setminus I$ for each domain D, we can define the embedding $\mathsf{Code}_{\to}$ as follows.

Definition-lemma 3.50 *The function*

$$\mathsf{Code}_{\to} \stackrel{def}{=} \mathcal{C}_{\mathcal{S}(D) \to E} \circ \mathcal{J}(\mathcal{C}_D^{-1} \to Id_E) \circ \widehat{\mathsf{Code}}_{\to} \circ (\mathcal{C}_D \to \mathcal{C}_E^{-1})$$

defines an embedding between $\mathcal{S}(D) \to \mathcal{S}(E)$ and $\mathcal{S}(\mathcal{S}(D) \to E)$.

Proof: Notice that $\mathcal{C}_D$ and $\mathcal{C}_D^{-1}$ are both embeddings for any domain D. Taking into account that Id_D is evidently an embedding, the application of a continuous functor (like $\to$ or $\mathcal{J}$) to an embedding is also an embedding, and the composition of embeddings is also an embedding, then $\mathsf{Code}_{\to}$ is an embedding, as far as we have yet proved (see lemma 3.49) that $\widehat{\mathsf{Code}}_{\to}$ is an embedding. ■

This alternative definition of the embedding $\mathsf{Code}_{\to}$ also leads to a functional ideal model provided that we define the value domain U as the initial solution of:

$$U \;\cong\; C + U \times U + \mathcal{S}(U) \to U$$

instead of using equation (3.1).

We can also mix both solutions and define U as the initial solution of:

$$U \;\cong\; C + U \times U + U \to U + \mathcal{S}(U) \to U$$

Then, two kinds of λ-abstraction may be defined, one interpreted in $\mathcal{S}(U \to U)$ using the first definition of $\mathsf{Code}_{\to}$, and the other interpreted in $\mathcal{S}(\mathcal{S}(U) \to U)$ using the second $\mathsf{Code}_{\to}$ definition.

3.4 An Ideal Model for COR

In this section we present the semantic rules used to map the language expressions over the semantic domain. As we have seen, we use $\mathcal{S}(U)$ as semantic domain. As usual, the semantic function is parametric in a valuation function. These valuation functions map identifiers to domain values: $\rho : \mathit{Ident} \longrightarrow \mathcal{S}(U)$.

Definition 3.51 *The semantic interpretation function*

$$\tau : \mathit{Expressions} \longrightarrow (\mathit{Ident} \longrightarrow \mathcal{S}(U)) \longrightarrow \mathcal{S}(U)$$

is defined inductively by:

$$\begin{array}{lcl}
\tau[\![\bot]\!]_\rho & = & \emptyset \\
\tau[\![\top]\!]_\rho & = & U \setminus \{\bot_U\} \\
\tau[\![x]\!]_\rho & = & \rho[\![x]\!] \\
\tau[\![t_1 \cup t_2]\!]_\rho & = & \tau[\![t_1]\!]_\rho \; \cup \; \tau[\![t_2]\!]_\rho \\
\tau[\![t_1 \cap t_2]\!]_\rho & = & \tau[\![t_1]\!]_\rho \; \cap \; \tau[\![t_2]\!]_\rho \\
\tau[\![t_1 \times t_2]\!]_\rho & = & \mathsf{Inter}_\times \langle \tau[\![t_1]\!]_\rho \; , \; \tau[\![t_2]\!]_\rho \rangle \\
\tau[\![\mathit{proj}_i \; (t)]\!]_\rho & = & \mathsf{Inter}_{\mathit{proj}_i} \; ([\![t]\!]_\rho) \\
\tau[\![\lambda x.t]\!]_\rho & = & \mathsf{Graph}(\lambda \epsilon \, . \, \tau[\![t]\!]_{\rho[\epsilon/x]}) \\
\tau[\![t_1(t_2)]\!]_\rho & = & \mathsf{Fun}(\tau[\![t_1]\!]_\rho)(\tau[\![t_2]\!]_\rho)
\end{array}$$

where $\rho[\epsilon/x]$ is the valuation ρ, except that it maps x to ϵ.

Lemmas 3.46 and 3.50 proves that $(\mathcal{S}(U), \mathsf{Fun}, \mathsf{Graph})$ is, in both cases, a COR-domain (see definition 2.4). The interpretation function $\tau[\![\cdot]\!]$ has been defined following the definition 2.6, therefore to prove that $(\mathcal{S}(U), \mathsf{Fun}, \mathsf{Graph}, \tau)$ is a COR-environment model we only need to prove the following lemma.

Lemma 3.52 *For any* COR*-term t the function $f \overset{\mathrm{def}}{=} \lambda S \in \mathcal{S}(U) \, . \, \tau[\![t]\!]_{\rho[S/x]}$ is continuous.*

Proof: The proof is done by induction on the structure of the term t.

(i) Suppose that $t = \lambda y \, . \, u$.
We have the following sequence of equalities

$$\begin{array}{rll}
\tau[\![t]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x]} & = \tau[\![\lambda y \, . \, u]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x]} & \\
& = \mathsf{Graph}(\lambda S \, . \, \tau[\![u]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x][S/y]}) & \text{definition of } \tau \\
& = \mathsf{Graph}(\lambda S \, . \, \bigsqcup_{i\in\mathbf{N}} \tau[\![u]\!]_{\rho[S_i/x][S/y]}) & \text{induction hypothesis} \\
& = \mathsf{Graph}(\bigsqcup_{i\in\mathbf{N}} \lambda S \, . \, \tau[\![u]\!]_{\rho[S_i/x][S/y]}) & \\
& = \bigsqcup_{i\in\mathbf{N}} \mathsf{Graph}(\lambda S \, . \, \tau[\![u]\!]_{\rho[S_i/x][S/y]}) & \text{continuity of } \mathsf{Graph} \\
& = \bigsqcup_{i\in\mathbf{N}} \tau[\![t]\!]_{\rho[S_i/x]}) &
\end{array}$$

which prove $f(\bigsqcup_{i\in\mathbf{N}} S_i) = \bigsqcup_{i\in\mathbf{N}} f(S_i)$.

(ii) Suppose that $t = u(v)$.
Then we have

$$\begin{array}{rll}
\tau[\![t]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x]} & = \tau[\![u(v)]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x]} & \\
& = \mathsf{Fun}(\tau[\![u]\!]_{\rho[\bigsqcup_{i\in\mathbf{N}} S_i/x]})(\tau[\![v]\!]_{\rho[\bigsqcup_{j\in\mathbf{N}} S_j/x]}) & \text{definition of } \tau \\
& = \mathsf{Fun}(\bigsqcup_{i\in\mathbf{N}} \tau[\![u]\!]_{\rho[S_i/x]})(\bigsqcup_{j\in\mathbf{N}} \tau[\![v]\!]_{\rho[S_j/x]}) & \text{induction hypothesis} \\
& = \bigsqcup_{i\in\mathbf{N}} \bigsqcup_{j\in\mathbf{N}} \mathsf{Fun}(\tau[\![u]\!]_{\rho[S_i/x]})(\tau[\![v]\!]_{\rho[S_j/x]}) & \text{continuity of } \mathsf{Fun} \\
& = \bigsqcup_{i\in\mathbf{N}} \mathsf{Fun}(\tau[\![u]\!]_{\rho[S_i/x]})(\tau[\![v]\!]_{\rho[S_i/x]}) & \\
& = \bigsqcup_{i\in\mathbf{N}} \tau[\![t]\!]_{\rho[S_i/x]}) &
\end{array}$$

For the rest of cases the proof is very similar and is also based on the continuity of the interpretation functions. ■

Theorem 3.53 *The tuple* $(\mathcal{S}(U), \mathsf{Fun}, \mathsf{Graph}, \tau)$ *is a* COR-*environment model.*

Proof: The proof is based on the previous lemma and the equality $\mathsf{Fun}(\mathcal{S}(U)) = \mathcal{S}(U) \to \mathcal{S}(U)$. The former means that the image of any filter for Fun is a continuous function, which has been already proved. ■

Our model, being a COR-model, is also a λ-model, and satisfies all provable equations of λ-calculus. Furthermore, our model allows to define a new type of formulas based on the refinement relation $\subseteq$.

Definition 3.54 *A term t_1 is a refinement of another term t_2 (noted $t_1 \subseteq t_2$) if for any valuation ρ the inclusion $\xi[\![t_1]\!]_\rho \subseteq \xi[\![t_2]\!]_\rho$ holds.*

3.5 Conclusions

We have proved that if we identify the *structural* order relation $\prec$ with the inverse of the *computational* ordering $\sqsubseteq$ in a functional domain U, then we can build over a COR *ideal model* $\mathcal{S}(U)$. It is also a λ-calculus model and allows to interpret λ-expressions as types. The definition of a functor $\mathcal{S}$ spreads out the family of λ-models. The study of such kind of models has not finished. We plan to prove some kind of initial property for them and to propose some other cases where both orderings would not be identified.

Chapter 4

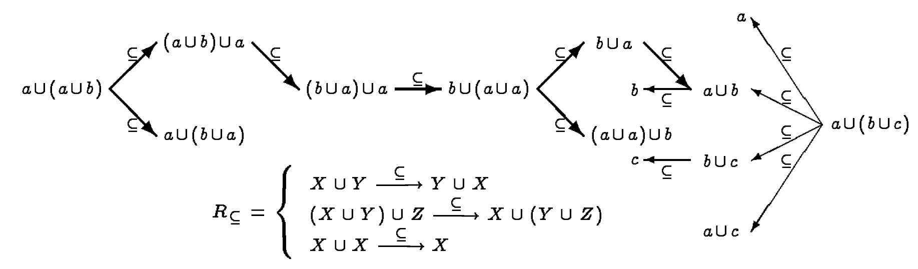
First-Order Bi-rewriting Systems

Abstract: In this chapter we propose an extension of term rewriting techniques to automate the deduction in monotone pre-order theories. To prove an inclusion $a \subseteq b$ from a given set I of them, we generate from I, using a completion procedure, a *bi-rewriting system* $\langle R_{\subseteq}, R_{\supseteq}\rangle$, that is, a pair of rewriting relations $\xrightarrow[R_{\subseteq}]{}$ and $\xrightarrow[R_{\supseteq}]{}$, and seek a common term c such that $a \xrightarrow[R_{\subseteq}]{*} c$ and $b \xrightarrow[R_{\supseteq}]{*} c$. Each component of the bi-rewriting system $\xrightarrow[R_{\subseteq}]{}$ and $\xrightarrow[R_{\supseteq}]{}$ is allowed to be a subset of the corresponding inclusion relation $\subseteq$ or $\supseteq$ defined by the theory of I. In order to assure the decidability and completeness of such proof procedure we study the termination and commutation of $\xrightarrow[R_{\subseteq}]{}$ and $\xrightarrow[R_{\supseteq}]{}$. The proof of the commutation property is based on a critical pairs lemma, using an *extended* definition of critical pair. We also extend the existing techniques of rewriting modulo equalities to bi-rewriting modulo a set of inclusions. Although we center our attention on the completion process à la Knuth-Bendix, the same notion of extended critical pair is suitable of being applied to the so called unfailing completion procedures. The completion process is illustrated by means of three examples corresponding to the theory of the union operator, non-distributive lattices and distributive lattices. We show that confluence of *extended* critical pairs may be ensured adding *rule schemes*. Such rule schemes contain variables denoting schemes of expressions, instead of expressions. We compare the results with the classical rewriting formulation.

4.1 Introduction

Rewriting systems are usually associated with rewriting on equivalence classes of terms, defined by a set of equations. However term rewriting techniques may be used to compute other relations than congruence. Particularly interesting are non-symmetric relations like pre-orders. In this chapter we will show the applicability of rewriting techniques to monotonic pre-order relations on terms, that is the deduction of inequalities —here we call them inclusions— from a given set of them.

The idea of applying rewriting techniques to the deduction of inclusions between terms, like $a \subseteq b$, is very simple. We compute by repeatedly replacing both 1) subterms of a by "bigger" terms using the axioms and 2) subterms of b by "smaller" terms using the same axioms until a path is found between a and b. Evidently there are many paths starting from a in the direction $\xrightarrow{\subseteq}$ and from b in the direction $\xrightarrow{\supseteq}$ (see figure 4.1). Many of them are blind alleys and others are not terminating. It is essential that the search procedure avoids infinite sequences of rewriting steps with infinitely many different terms (infinite paths due to cycles can be avoided if we control the introduction of repeated terms). Obviously infinitely many different rewriting steps would prevent the termination of the procedure. The solution to non-termination is, like in term rewriting systems, to orient the axioms using a well founded ordering on terms. Because the relation is non-symmetric, the orientation results in a pair of rewriting systems $\langle R_{\subseteq}, R_{\supseteq}\rangle$, i.e. we get what we call a bi-rewriting system. We introduce the definitions of Church-Rosser and quasi-terminating bi-rewriting system in order to assure the soundness, completeness and termination of the search procedure. That is, given a set of axioms, if we can orient and complete them obtaining a quasi-terminating and Church-Rosser bi-rewriting system, then we will have a decision algorithm to test $a \subseteq b$.



$$R_{\subseteq} = \begin{cases} X \cup Y \xrightarrow{\subseteq} Y \cup X \\ (X \cup Y) \cup Z \xrightarrow{\subseteq} X \cup (Y \cup Z) \\ X \cup X \xrightarrow{\subseteq} X \end{cases}$$

$$R_{\supseteq} = \begin{cases} X \cup Y \xrightarrow{\supseteq} X \\ X \cup Y \xrightarrow{\supseteq} Y \end{cases}$$

Figure 4.1: A graphical representation of the bi-rewriting algorithm

Most of the notions of rewriting developed for the equational case can be extended to bi-rewriting and the development of the chapter follows the same pattern as equational rewriting: the Church-Rosser property is proved by means of a critical pairs lemma, and we use a completion process to ensure the confluence of the critical pairs (Knuth and Bendix, 1970; Huet, 1980; Klop, 1987; Dershowitz and Jouannaud, 1990). However there are also some differences. Equational rewriting is in essence a theory of normal forms, while bi-rewriting disregards this notion. Bi-rewriting can also be seen as a generalization of equational rewriting: equations can be translated to pairs of inclusions and then we can reproduce the equational case. One of the costs of this generalization is that bi-rewriting is based on a search procedure, which is avoided in canonical rewriting systems thanks to the existence of unique normal forms. Another cost is that now critical pairs must be computed considering variable overlapping, producing possibly infinitely many of them, which are represented as critical pair schemes.

This chapter proceeds as follows.

In section 4.2 we present a version of the critical pairs lemma for bi-rewriting systems using an *extended* definition of critical pairs. We also give a counter-example that invalidates this lemma stated only in terms of *standard* critical pairs.

In section 4.3 we generalize the results of section 4.2 to bi-rewriting systems modulo a set of (non-orientable) inclusions. We have divided this section in two subsections, the first devoted to abstract bi-rewriting properties and the second to term dependent properties.

In section 4.4 we present three examples of canonical bi-rewriting systems for the theories of union, non-distributive lattices and distributive lattices. We show that although in general extended critical pairs could be intractable, there exist for this theory, and possibly for others, practical ways to handle them.

We also show in section 4.5 some of the disadvantages of using equations to model inclusions in lattice theories.

4.2 Inclusions and Bi-rewriting Systems

If nothing is said, we follow the notation and the standard definitions used in (Huet, 1980; Klop, 1987; Dershowitz and Jouannaud, 1990). We are concerned with *first-order terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$ over a nonempty *signature* $\mathcal{F} = \bigcup_{n\in\mathbb{N}}\mathcal{F}_n$ of function symbols, and a denumerable set $\mathcal{X}$ of *variables.*[1] A *position* p is a sequence of positive integers. Given two positions, $p_1 \cdot p_2$ denotes their concatenation. We write $p_1 \prec p_2$ when p_1 is a prefix of p_2 and $p_1|p_2$ when they are disjoint.[2] The *occurrence*[3] of a subexpression at a position p of a term t is denoted by $t|_p$. The expression $t[u]_p$ denotes the result of replacing in t the occurrence of $t|_p$ by u.[4] A *context* $F[\cdot]_p$ is an expression with a *hole* $[\cdot]$ at a distinguished position p. The *set of (free) variables* of a term t is denoted by $\mathcal{FV}(t)$. A *substitution*

[1] As we will see later, in most cases we also require the finiteness of $\mathcal{F}$. We suppose that $\mathcal{F}_n$ are disjoint sets. The set $\mathcal{T}(\mathcal{F},\mathcal{X})$ is defined as the smallest set containing $\mathcal{X}$ such that if $f \in \mathcal{F}_n$ and $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ for $i = 1,\dots,n$ then $f(t_1,\dots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$.

[2] We write $p_1 \prec p_2$ when there exists a sequence q such that $p_2 = p_1 \cdot q$, and $p_1|p_2$ when $p_1 \not\prec p_2$ and $p_2 \not\prec p_1$.

[3] If p is an empty sequence then $t|_p$ is defined by $t|_{<>} \overset{def}{=} t$ otherwise it is defined inductively by $f(t_1,\dots,t_n)|_{<i_1,i_2,\dots,i_r>} \overset{def}{=} t_{i_1}|_{<i_2,\dots,i_r>}$.

[4] If p is the empty sequence then $t[u]_{<>} \overset{def}{=} u$, otherwise $f(t_1,\dots,t_n)[u]_{<i_1,\dots,i_m>} \overset{def}{=} f(t_1,\dots,t_{i_1}[u]_{<i_2,\dots,i_m>},\dots,t_n)$.

$\sigma = [X_1 \mapsto t_1, \ldots, X_n \mapsto t_n]$ is a mapping from a finite set $\{X_1, \ldots, X_n\} \subseteq \mathcal{X}$ of variables to $\mathcal{T}(\mathcal{F}, \mathcal{X})$, extended as a morphism[5] to $\mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$. The set $\mathcal{D}om(\sigma) \overset{def}{=} \{X_1, \ldots, X_n\}$ is called the *domain* of the substitution. We use the *relational logic*[6] notation (deKogel, 1992; Bäumer, 1992) to present the abstract bi-rewriting properties. The inverse of the relation R is denoted by R^{-1}, its reflexive-transitive closure by R^*, the transitive composition by $R_1 \circ R_2$, and the union by $R_1 \cup R_2$. Notation R^+ is a shorthand for $R \circ R^*$. A relation R is said to be *terminating* if R^+ is a well-founded ordering, *quasi-terminating* if the set $\{u \mid t \; R^* \; u\}$ is finite for any value t; and *finitely branching* if $\{u \mid t \; R \; u\}$ is finite for any t. A binary relation R on terms is said to be *closed under substitutions* if $t \; R \; u$ implies $\sigma(t) \; R \; \sigma(u)$, for any substitution σ and pair of terms t and u; *monotonic* if $t \; R \; u$ implies $F[t]_p \; R \; F[u]_p$, for any context $F[\cdot]_p$; and a *rewrite relation* if it is closed under substitutions and monotonic. We denote by $\xrightarrow[R]{}$ the rewrite relation defined by the set of rules R.[7] Notation $\xleftarrow[R]{}$ is a shorthand for $(\xrightarrow[R]{})^{-1}$.

An **inclusion** is a pair of terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ written $s \subseteq t$. Given a finite set of inclusions Ax and a pair of terms s and t, we say that $s \subseteq_{Ax} t$ iff $Ax \vdash_{POL} s \subseteq t$, where *POL* stands for *Partial Order Logic* and $\vdash_{POL}$ is the entailment relation defined by the following inference rules

$$\frac{}{\Delta, s \subseteq t \vdash_{POL} s \subseteq t} \qquad \frac{}{\Delta \vdash_{POL} s \subseteq s} \qquad \frac{\Delta \vdash_{POL} s \subseteq t \quad \Delta \vdash_{POL} t \subseteq u}{\Delta \vdash_{POL} s \subseteq u}$$

$$\frac{\Delta \vdash_{POL} s \subseteq t}{\Delta \vdash_{POL} \sigma(s) \subseteq \sigma(t)} \qquad \frac{\Delta \vdash_{POL} s \subseteq t}{\Delta \vdash_{POL} u[s]_p \subseteq u[t]_p}$$

where σ is a substitution, p a position in u, i.e. $u[\cdot]_p$ is a context, and Δ is a finite set of inclusions.

Meseguer (Meseguer, 1990; Meseguer, 1992) has studied widely the logic of conditional inequalities, which he names *rewriting logic*, and its models.

The set of inclusions $s \subseteq t$ that can be inferred from Ax using $\vdash_{POL}$ forms an inclusion theory, noted by $Th(Ax)$. Notice that, in first-order logic, $Th(Ax)$ is a denumerable set and the deduction problem $Ax \vdash_{POL} s \subseteq t$ is semi-decidable. In the following we will propose sufficient conditions to have a decision algorithm for $Ax \vdash_{POL} s \subseteq t$ based on rewriting techniques.

Given an inclusion $s \subseteq t$ of Ax, we can orient it obtaining a term rewriting rule $s \xrightarrow{\subseteq} t$ or a rule $t \xrightarrow{\supseteq} s$. Thus, the orientation, for rewriting purposes, of a finite set of inclusions Ax results in two sets of rewriting rules, $R_\subseteq$ with rules like $s \xrightarrow{\subseteq} t$ and $R_\supseteq$ with rules like $s \xrightarrow{\supseteq} t$. The pair $\langle R_\subseteq, R_\supseteq \rangle$ is called a bi-rewriting system.

[5]That is, $\sigma(f(t_1, \ldots, t_n)) \overset{def}{=} f(\sigma(t_1), \ldots, \sigma(t_n))$.

[6]This is an extension of the Kleen algebra for regular expressions, i.e. those built up using the constructors $\cup$, $\circ$ and $*$.

[7]The minimal rewriting relation satisfying $s \xrightarrow[R]{} t$ for any rule $s \to t \in R$.

Definition 4.1 *A* **(term) bi-rewriting system** *is a pair $\langle R_\subseteq, R_\supseteq\rangle$ of finite sets of (term) rewriting rules $R_\subseteq = \{s_1 \xrightarrow{\subseteq} t_1, \ldots, s_n \xrightarrow{\subseteq} t_n\}$ and $R_\supseteq = \{u_1 \xrightarrow{\supseteq} v_1, \ldots, u_m \xrightarrow{\supseteq} v_m\}$.*
Given a bi-rewriting system $\langle R_\subseteq, R_\supseteq\rangle$, its **corresponding inclusion theory** *is defined by the set of axioms $Ax = \{s \subseteq t \mid s \xrightarrow{\subseteq} t \in R_\subseteq \ \vee\ t \xrightarrow{\supseteq} s \in R_\supseteq\}$.*

The orientation criteria is based, like in rewriting systems, on a well-founded ordering on terms (noted as $\succ$) (Dershowitz, 1987). In this section we suppose that each inclusion $s \subseteq t$ in Ax may be oriented, putting $s \xrightarrow{\subseteq} t$ in $R_\subseteq$ if $s \succ t$, or $t \xrightarrow{\supseteq} s$ in $R_\supseteq$ if $t \succ s$. In the next section we will consider the case of inclusions which can not be oriented because $s \not\succ t$ and $t \not\succ s$. For example, inclusions defining the inclusion theory of the union may be oriented using a simplification ordering as it is shown in figure 4.2.

$$Ax = \left\{ \begin{array}{l} X \cup X \subseteq X \\ X \subseteq X \cup Y \\ Y \subseteq X \cup Y \end{array} \right. \qquad \begin{array}{l} R_\subseteq = \{\ r_1 : \quad X \cup X \xrightarrow{\subseteq} X \\ \\ R_\supseteq = \left\{ \begin{array}{ll} r_2 : & X \cup Y \xrightarrow{\supseteq} X \\ r_3 : & X \cup Y \xrightarrow{\supseteq} Y \end{array} \right. \end{array}$$

Figure 4.2: Orientation of the inclusion theory of the union.

Given a bi-rewriting system $\langle R_\subseteq, R_\supseteq\rangle$ the monotonic and substitution closure of each one of its components $R_\subseteq$ and $R_\supseteq$ results in a rewriting relation, noted by $\xrightarrow[R_\subseteq]{}$ and $\xrightarrow[R_\supseteq]{}$ respectively, defined as follows.

Definition 4.2 *We say that s* **R-rewrites to** *t, written $s \xrightarrow[R]{} t$, if there exist a rule $l \longrightarrow r \in R$, a position p in s, and a substitution σ, such that $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$.*

If $s|_p = \sigma(l)$ then we say that $s|_p$ and l match. Notice that if $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$ then the substitution σ in the previous definition, with its domain restricted to $\mathcal{Dom}(\sigma) \subseteq \mathcal{FV}(l)$, is unique.

A variant of the theorem of Birkhoff (Birkhoff, 1935) allows to prove the following lemma.

Lemma 4.3 *Given a bi-rewriting system $\langle R_\subseteq, R_\supseteq\rangle$ and its corresponding inclusion theory Ax, for any pair of terms s, t we have $s\ (\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*\ t$ if, and only if, $Ax \vdash_{POL} s \subseteq t$.*

However, the relation $(\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*$ is in general not computable, i.e. given two terms s and t there does not exist a decision algorithm for $s\ (\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*\ t$. We are interested in reducing the previous relation into the subrelation $\xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*}$, which we will show is computable.

Based on the bi-rewriting system $\langle R_\subseteq, R_\supseteq\rangle$ a deduction procedure for its corresponding inclusion theory $Th(Ax)$ can be easily defined (see figure 4.1). To prove $Ax \vdash_{POL} s \subseteq t$ the procedure enumerates recursively the nodes of two trees T_1 and T_2, defined by $\mathsf{root}_{T_1} = s$, $\mathsf{root}_{T_2} = t$, $\mathsf{branch}_{T_1}(s_1) = \{s_2 \mid s_1 \xrightarrow[R_\subseteq]{} s_2\}$ and $\mathsf{branch}_{T_2}(t_1) = \{t_2 \mid t_1 \xrightarrow[R_\supseteq]{} t_2\}$, avoiding repeated nodes. If the procedure finds a common node in both trees then it stops and answers *true*, otherwise if both sets of nodes are finite then it stops and answers *false* or else it does not stop.

Notice that the nodes of both trees are always recursively enumerable, although the trees may be infinitely branching. We say that a tree is infinitely branching if it contains a node with infinitely many branches.

The following definition states sufficient conditions for the soundness and completeness, and for the termination of this procedure. Notice that the soundness and completeness properties are based on the equivalence of the relation $\xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*}$ computed by the algorithm and the relation $(\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*$ implementing the inclusion relation defined by the theory. The termination property is based on the finiteness of both search trees.

Definition 4.4 *A bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ is said to be*

(i) **terminating** *iff* $(\xrightarrow[R_\subseteq]{} \cup \xrightarrow[R_\supseteq]{})^*$ *is a well-founded ordering;*[8]

(ii) **quasi-terminating** *or* **globally finite** *iff the sets* $\{u \mid t \xrightarrow[R_\subseteq]{*} u\}$ *and* $\{v \mid t \xrightarrow[R_\supseteq]{*} v\}$ *are both finite for any term t; and*

(iii) **Church-Rosser** *iff* $(\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* \subseteq \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*}$.

We can prove the following results for the decision procedure based on a bi-rewriting system, and the $Ax \vdash_{POL} t \subseteq u$ deduction problem of its corresponding inclusion theory.

Lemma 4.5 *If the bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ is Church-Rosser then the decision procedure based on it is sound and complete, i.e. $Ax \vdash_{POL} t \subseteq u$ holds if, and only if, the procedure terminates and answers* true.
If the bi-rewriting system is Church-Rosser and quasi-terminating then the decision procedure is sound, complete and terminates, therefore the satisfiability problem is decidable.

We only need to require the quasi-termination property of the bi-rewriting system —which is (strictly) weaker than the termination property— in order to prove the termination of the procedure; whereas in the equational case, the termination property of the rewriting system is needed to prove the termination of a procedure based on the computation of the normal form.

Lemma 4.6 *Any terminating term bi-rewriting system is quasi-terminating.*

[8] In a previous version of this work (Levy and Agustí, 1993c), a bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ is said to be terminating iff both $\xrightarrow[R_\subseteq]{*}$ and $\xrightarrow[R_\supseteq]{*}$ are well-founded orderings. This is a weaker condition and it is not enough to prove the equivalence between the Church-Rosser and the local bi-confluence properties. This error was communicated to the authors by Professor Harald Ganzinger.

Proof: If $(\xrightarrow[R_\subseteq]{} \cup \xrightarrow[R_\supseteq]{})^*$ is terminating then both $\xrightarrow[R_\subseteq]{*}$ and $\xrightarrow[R_\supseteq]{*}$ are terminating, and the problem is reduced to prove that any terminating term rewriting system is quasi-terminating.

First we prove that any terminating *term* rewriting relation is finitely branching. If $\xrightarrow[R]{}$ is terminating then any rewriting rule $l \longrightarrow r$ in R satisfies $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$ (otherwise it would be easy to construct a non terminating sequence of terms taking a convenient instantiation of one the variables of $\mathcal{FV}(r) \backslash \mathcal{FV}(l)$). Now, to rewrite a term we have finitely many ways to choose a rule $l \longrightarrow r$ and a subterm $t|_p$. Once we have fixed them, if it exists, there is a unique substitution satisfying $\mathcal{Dom}(\sigma) \subseteq \mathcal{FV}(l)$ and $t|_p = \sigma(l)$. Finally, if $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$, such substitution determines the result of the rewriting step.

Second we prove that any finitely branching and terminating relation is quasi-terminating. This is a straightforward application of the Koenig's lemma, although it can also be proved directly by noetherian induction. Let P be the predicate $P(s) \stackrel{def}{=}$ "the set $\{t \mid s \xrightarrow[R]{*} t\}$ is finite". Evidently, it holds for the base case, i.e. for the $\xrightarrow[R]{}$-normal terms. For the induction case we have $\{x \mid t \xrightarrow[R]{*} x\} = \{t\} \cup \bigcup_{u \in \{u \mid t \xrightarrow[R]{} u\}} \{y \mid u \xrightarrow[R]{*} y\}$. Now, as far as each one of the sets $\{y \mid u \xrightarrow[R]{*} y\}$ is finite by induction hypothesis, and there are finitely many of this sets, because $\xrightarrow[R]{}$ is finitely branching, we have that $\{x \mid t \xrightarrow[R]{*} x\}$ is also finite. By induction we conclude that $P(s)$ holds for any term s and $\xrightarrow[R]{}$ is quasi-terminating. ■

In order to test automatically the Church-Rosser property we extend the standard procedure used in term rewriting to bi-rewriting. So we reduce the Church-Rosser property to three simpler properties, namely *bi-confluence* (or commutativity), *local bi-confluence* and *critical pairs bi-confluence.*

Definition 4.7 *A bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ is said to be*

(i) **bi-confluent** *iff* $\xleftarrow[R_\supseteq]{*} \circ \xrightarrow[R_\subseteq]{*} \subseteq \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*}$ *; and*

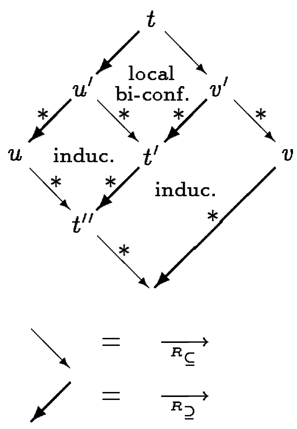
(ii) **locally bi-confluent** *iff* $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[R_\subseteq]{} \subseteq \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*}$.

A pair of terms $\langle s, t \rangle$ is said to be **bi-confluent** *iff* $s \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*} t$.

A variant of the Newman's lemma (Newman, 1942; Huet, 1980) proves the following result for bi-rewriting systems.

Lemma 4.8 *A terminating bi-rewriting system is Church-Rosser iff it is locally bi-confluent.*

Proof: The only if implication is trivially proved since $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[R_\subseteq]{} \subseteq (\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*$. The proof for the if implication is done by noetherian induction and is quite similar to the standard proof for the confluence of terminating and locally confluent term rewriting systems (Huet, 1980). If fact the statement is implied by lemma 1.2 in (Bachmair and Dershowitz, 1986a).



We prove that property

$$P(t) \stackrel{def}{=} \forall u, v \,.\, u \xleftarrow[R_\supseteq]{*} t \xrightarrow[R_\subseteq]{*} v \;\Rightarrow\; u \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*} v$$

holds for any term t by noetherian induction. The base case $t = u$ or $t = v$ is trivially satisfied. The induction case follows directly from the induction hypothesis $P(u')$ and $P(v')$ using the diagram on the left.

■

Notice that in the previous lemma we require the union of both rewriting relations to be well-founded, and it is not sufficient if both relations are well-founded separately. For instance, the bi-rewriting system defined by $R_\subseteq = \{b \xrightarrow{\subseteq} c, c \xrightarrow{\subseteq} d\}$ and $R_\supseteq = \{c \xrightarrow{\supseteq} b, b \xrightarrow{\supseteq} a\}$ is locally bi-confluent and both rewriting relations $\xrightarrow[R_\subseteq]{}$ and $\xrightarrow[R_\supseteq]{}$ are terminating, not their union. However, the bi-rewriting system is not Church-Rosser.

A simple adaptation of the standard critical pairs definition (Knuth and Bendix, 1970) can be given for bi-rewriting systems. However, as we will see, it is not sufficient to prove the critical pairs lemma. This simple definition of critical pair arises from the most general non-variable overlap between the left hand side of a rule in $R_\subseteq$ and a sub-expression of the left hand side of a rule in $R_\supseteq$, (or viceversa). Given a pair of rules $l \xrightarrow{\subseteq} r$ and $s \xrightarrow{\supseteq} t$, a position p of a non-variable subterm of s, and the most general unifier σ of l and $s|_p$, the pair $\sigma(t) \subseteq \sigma(s[r]_p)$ is a (standard) critical pair between $R_\subseteq$ and $R_\supseteq$; and similarly for critical pairs between $R_\supseteq$ and $R_\subseteq$.

Unfortunately, in the presence of non-left-linear rules,[9] the critical pair lemma stated in terms of such standard critical pairs can not be proved because the confluence of variable overlaps is no longer possible. The same fact has already been discussed in (Bachmair, 1991). Here is a simple counter-example to the validity of this lemma.

Counter-example. The following bi-rewriting system

$$R_\subseteq = \{f(X,X) \xrightarrow{\subseteq} X\} \qquad R_\supseteq = \{a \xrightarrow{\supseteq} b\}$$

is terminating and has no standard critical pairs, however the divergence $f(a,b) \xleftarrow[R_\supseteq]{} f(a,a) \xrightarrow[R_\subseteq]{} a$ does not satisfy the Church-Rosser property (the pair $f(a,b) \subseteq a$ is not bi-confluent). This problem would be avoided if $a \xrightarrow{\subseteq} b \in R_\subseteq$, but then the inclusion theory corresponding to the bi-rewriting system would be different.

[9] A rule $l \longrightarrow r$ is left- (right-) linear iff any variable in l (in r) occurs at most once in l (in r).

Non-left-linear rules also invalidate the bi-rewrite parallel of Toyama's theorem (Toyama, 1987) as the following counter-example shows.

Counter-example. The following bi-rewriting system

$$R_{\subseteq} = \begin{cases} X \cup X \xrightarrow{\subseteq} X \\ X \cup Y \xrightarrow{\subseteq} Y \cup X \\ X \cup (Y \cup Z) \xrightarrow{\subseteq} (X \cup Y) \cup Z \end{cases} \qquad R_{\supseteq} = \begin{cases} X \cup Y \xrightarrow{\supseteq} X \\ X \cup Y \xrightarrow{\supseteq} Y \end{cases}$$

is Church-Rosser and quasi-terminating, if we consider a signature containing uniquely constants and the binary union operator, i.e. $\mathcal{F}_2 = \{\cup\}$ and $\mathcal{F}_i = \emptyset$ for $i \notin \{0, 2\}$. However, if we introduce a new 1-ary symbol in the signature $f \in \mathcal{F}_1$ then we have the following divergence which is not bi-confluent.

$$f(X) \cup f(Y) \xleftarrow[R_{\supseteq}]{} f(X) \cup f(X \cup Y) \xleftarrow[R_{\supseteq}]{} f(X \cup Y) \cup f(X \cup Y) \xrightarrow[R_{\subseteq}]{} f(X \cup Y)$$

This means that many properties of bi-rewriting systems depend not only on the axioms of the theory but also on the signature.

Using the standard definition of critical pairs, the critical pairs lemma is only true for left-linear systems: a terminating and left-linear bi-rewriting system is Church-Rosser iff all standard critical pairs are bi-confluent. In order to keep this lemma for non-left-linear bi-rewriting systems, we have to enlarge the set of critical pairs to be considered as follows.

Definition 4.9 *If $l \xrightarrow{\subseteq} r \in R_{\subseteq}$ and $s \xrightarrow{\supseteq} t \in R_{\supseteq}$ are two rewriting rules (with variables distinct) and p a position in s, then*

(i) *if $s|_p$ is a non-variable subterm and σ is the most general unifier of $s|_p$ and l then*

$$\sigma(t) \subseteq \sigma(s[r]_p) \in ECP(R_{\subseteq}, R_{\supseteq})$$

is a **(standard) critical pair,**

(ii) *if $s|_p = x$ is a repeated variable in s, F is a term containing only fresh variables, q is an occurrence in F, and $l \xrightarrow[R_{\supseteq}]{*} r$ does not hold,*[10] *then*

$$\sigma(t) \subseteq \sigma(s[F[r]_q]_p) \in ECP(R_{\subseteq}, R_{\supseteq})$$

is an **(extended) critical pair** *where $\sigma = [x \mapsto F[l]_q]$.*

Similarly for critical pairs between $R_{\supseteq}$ and $R_{\subseteq}$, $ECP(R_{\supseteq}, R_{\subseteq})$.

The set of (extended) critical pairs of the previous definition is in general infinite, $\sigma(t) \subseteq \sigma(s[F[r]_q]_p)$ is really a *critical pair scheme* because we do not impose any restriction on the *context* $F[\cdot]_q$. In section 4.4 we will see an example where we use such kind of schemes. So the critical pairs lemma even if true with this definition of critical pairs, will be of little practical help to test bi-confluence. Then the conditions of bi-confluence have to be studied in each case taking into account the particular shape of the non-left-linear rules. In section 5.2 we face the problem of testing bi-confluence automatically by codifying extended critical pairs using the linear second-order typed λ-calculus.

[10] If this condition is satisfied then we can make the pair resulting from the variable overlapping confluent like in the equational case.

Theorem 4.10 extended critical pair lemma. *A terminating bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ is Church-Rosser iff any (standard or extended) critical pair $s \subseteq t$ in $ECP(R_\subseteq, R_\supseteq)$ or $s \supseteq t$ in $ECP(R_\supseteq, R_\subseteq)$ is bi-confluent, i.e. $s \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*} t$.*

Proof: For the if part, see the proof of theorem 4.19, which states a more general result, taking $I_\subseteq = \emptyset$. For the only if part, extended critical pairs are sound deductions, therefore if $s \subseteq t$ is an extended critical pair, then $s(\xrightarrow[R_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* t$ holds. Now, if the bi-rewriting system is Church-Rosser, then $s \xrightarrow[R_\subseteq]{*} \circ \xleftarrow[R_\supseteq]{*} t$. ∎

This theorem, lemma 4.17 and theorem 4.19 may be considered as instances of the *general critical pair theorem* proved by Geser in his thesis (Geser, 1990). Nevertheless, we have decided to include a sketch of their proof for completeness.

The extended critical pair theorem generalizes the critical pairs lemma (Knuth and Bendix, 1970) for bi-rewriting systems. However, we require the bi-confluence of not only the standard critical pairs, but also of the extended critical pairs. Nevertheless, if all rules come from the translation of an equational theory E, then any equation $a = b$ with $a \succ b$ results in two bi-rewriting rules $a \xrightarrow{\subseteq} b$ in $R_\subseteq$ and $a \xrightarrow{\supseteq} b$ in $R_\supseteq$ and both bi-rewriting relations $\xrightarrow[R_\subseteq]{} = \xrightarrow[R_\supseteq]{}$ are equal. Then we only obtain standard critical pairs because the condition $l \xrightarrow[R_\supseteq]{*} r$ in the definition 4.9 of extended critical pair is always satisfied. So we recover the old results for the equational case.

4.3 Bi-rewriting Modulo a Set of Inclusions

Like in equational rewriting, in bi-rewriting it is not always possible to orient all inclusions of a theory presentation in two terminating rewrite relations, as was assumed in the previous section. Frequently enough, we must handle three rewrite relations, the terminating relations $\xrightarrow[R_\subseteq]{}$ and $\xrightarrow[R_\supseteq]{}$ resulting from the inclusions $R_\subseteq$ and $R_\supseteq$ oriented to the right and to the left respectively, and the non-terminating relation $\xleftrightarrow[I_\subseteq]{}$ resulting from the non-oriented inclusions I. Then we say to have a $\langle R_\subseteq, R_\supseteq \rangle$ bi-rewriting system modulo I.[11] Figure 4.3 in section 4.4 shows an example of these bi-rewriting systems. The inverse of the relation $\xleftrightarrow[I_\subseteq]{}$ is noted $\xleftrightarrow[I_\supseteq]{}$. The Birkhoff theorem is stated then as $Ax \vdash_{POL} t \subseteq u$ iff $t\ (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*\ u$.

4.3.1 From Church-Rosser to Local Bi-Confluence

The simplest way to have a complete and decidable proof procedure based on the $\langle R_\subseteq, R_\supseteq \rangle$ bi-rewriting system modulo I is reducing it to the bi-rewriting system $\langle R_\subseteq \cup I, R_\supseteq \cup I \rangle$ and, using the results of the previous section, requiring of it the following properties:

[11] Although we use the word "modulo", it does not mean that $\xleftrightarrow[I_\subseteq]{*}$ is a congruence, be aware it is a non-symmetric relation (monotonic pre-order).

1. The relations $\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{}$ and $\xrightarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\supseteq]{}$ are both quasi-terminating, and

2. they satisfy the *(weak) Church-Rosser* property

$$(\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* \subseteq (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^* \circ (\xleftarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^*$$

However, as we have seen in the previous section the quasi-termination of $\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{}$ and $\xrightarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\supseteq]{}$ is not enough to reduce the (weak) Church-Rosser property to the local bi-confluence property $(\xleftarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^* \circ (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^* \subseteq (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^* \circ (\xleftarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^*$ using lemma 4.8. To do this we would need the termination of $\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xrightarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\supseteq]{}$, which, of course, never holds.[12] The solution to this problem comes from requiring the termination of $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$. Using this termination property, the weak Church-Rosser property can be reduced to a local bi-confluence property.

Lemma 4.11 *If the relation* $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$ *is terminating, then the following properties*

$$(\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* \subseteq (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^* \circ (\xleftarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\subseteq]{})^*$$

(weak) Church-Rosser

$$\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \subseteq (\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{})^* \circ \xleftrightarrow[I_\subseteq]{*} \circ (\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*})^*$$

(weak) local bi-confluence

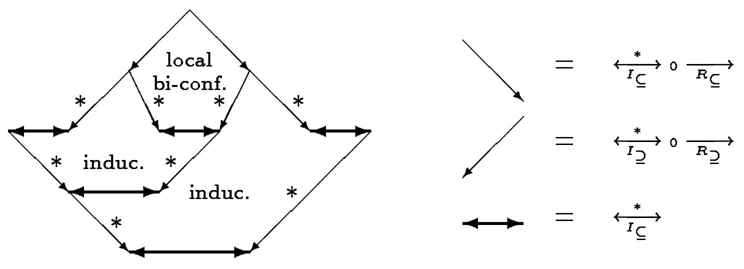
are equivalent.

Proof: Using the equalities $(A \cup B)^* = (A^* \circ B)^* \circ A^* = A^* \circ (B \circ A^*)^*$ we prove that right hand sides of both inclusions are equal. Now $\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \subseteq (\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^*$ shows that local bi-confluence implies Church-Rosser.

For the converse we use $(A \cup B)^* \subseteq A^* \circ B^* \Leftrightarrow B^* \circ A^* \subseteq A^* \circ B^*$ to prove the equivalence between the Church-Rosser property and the following one.

$$\xleftrightarrow[I_\subseteq]{*} \circ (\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*})^* \circ (\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{})^* \circ \xleftrightarrow[I_\subseteq]{*} \subseteq (\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{})^* \circ \xleftrightarrow[I_\subseteq]{*} \circ (\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*})^*$$

Now, if $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$ is terminating we can prove by noetherian induction that this property is equivalent to the local bi-confluence property. The following diagram shows the scheme of the proof.


local
bi-conf.
induc.
induc.
*
= $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{}$
= $\xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$
= $\xleftrightarrow[I_\subseteq]{*}$

[12]The relation $\xleftrightarrow[I_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{}$ is never terminating.

Where the proposition proved by noetherian induction is the following one.

$$P(t) = \forall u, v \,.\, u \xleftrightarrow[I_\subseteq]{*} \circ (\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*})^* t \;\wedge\; t(\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{})^* \circ \xleftrightarrow[I_\subseteq]{*} v \Rightarrow$$
$$\Rightarrow u(\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{})^* \circ \xleftrightarrow[I_\subseteq]{*} \circ (\xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*})^* v$$

■

If $\xleftrightarrow[I_\subseteq]{}$ is symmetric ($\xleftrightarrow[I_\subseteq]{} = \xleftrightarrow[I_\supseteq]{}$) the above termination property becomes similar to the termination property required in rewriting modulo a set of equations (Bachmair and Dershowitz, 1989a). That is, $\xleftrightarrow[I_\subseteq]{}$ symmetric means we can define equivalence classes ($[s]_I \xrightarrow[R]{} [t]_I$ iff $s \xleftrightarrow[I]{*} \circ \xrightarrow[R]{} \circ \xleftrightarrow[I]{*} t$) and, the termination of $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$ is ensured by the existence of a well-founded I-compatible quasi-ordering[13] $\preceq$ satisfying $\xrightarrow[R_\subseteq]{} \subseteq \succ$, $\xrightarrow[R_\supseteq]{} \subseteq \succ$ and $\xleftrightarrow[I_\subseteq]{} \subseteq \approx$, where the equivalence relation $\approx$ is the intersection of $\succeq$ and $\preceq$ and the strict ordering $\succ$ is the difference of $\succeq$ and $\approx$. The quasi-termination property of $\xleftrightarrow[I_\subseteq]{}$ is equivalent then to the finiteness of the equivalence classes.

However, like in the equational case, rewriting by $\xleftrightarrow[I]{*} \circ \xrightarrow[R]{}$ is inefficient, and the local commutativity of $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{}$ and $\xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$ can not be reduced to the bi-confluence of a selected set of critical pairs. Therefore we will approximate them by two weaker, but more practical rewriting relations, named $I \backslash R_\subseteq$ and $I \backslash R_\supseteq$ respectively by similarity to the corresponding equational definitions. In the following, we prove the abstract properties of these relations. We will suppose that they satisfy:[14]

$$\xrightarrow[R_\subseteq]{} \subseteq \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{}$$
$$\xrightarrow[R_\supseteq]{} \subseteq \xrightarrow[I\backslash R_\supseteq]{} \subseteq \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$$

leaving their definition for the next subsection.

We require these new rewriting relations to satisfy what we call a *strong Church-Rosser modulo I* property, defined as follows.

Definition 4.12 *The bi-rewriting system $\langle R_\subseteq, R_\supseteq \rangle$ modulo I is* **(strong) Church-Rosser** *iff*

$$(\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*}$$

The following lemma states sufficient conditions to define a search decision procedure for $Ax \vdash_{POL} t \subseteq u$ based on the relations $I \backslash R_\subseteq$ and $I \backslash R_\supseteq$.

Lemma 4.13 *If the relations $\xrightarrow[I\backslash R_\subseteq]{}$ and $\xrightarrow[I\backslash R_\supseteq]{}$ are both computable*[15] *and quasi-terminating, the relation $\xleftrightarrow[I_\subseteq]{*}$ is decidable, and $\langle R_\subseteq, R_\supseteq \rangle$ is strong Church-Rosser modulo I, then there exists a decision procedure for the inclusion relation defined by these relations.*

[13] A well-founded quasi-ordering is a well-founded, reflexive and transitive relation.

[14] Notice that although we use the notation $\xrightarrow[I\backslash R]{}$, it does not means that this relation is the monotonic and substitution closure of a set of rules, $I \backslash R$ is just the name of the relation.

[15] We say that a relation $\xrightarrow[R]{}$ is computable iff the set $\{u \mid t \xrightarrow[R]{} u\}$ is finite and computable for any given term t.

Proof: Like in the simpler case of the previous section, given two terms s and t, the algorithm generates the sets $\{s' \mid s \xrightarrow[I\backslash R_{\subseteq}]{*} s'\}$ and $\{t' \mid t \xrightarrow[I\backslash R_{\supseteq}]{*} t'\}$ and seek for a term s' from the first set and a term t' from the second one such that $s' \xleftrightarrow[I_{\subseteq}]{*} t'$. If relations $\xrightarrow[I\backslash R_{\subseteq}]{*}$ and $\xrightarrow[I\backslash R_{\supseteq}]{*}$ satisfy the above inclusions and the Church-Rosser property then $(\xrightarrow[R_{\subseteq}]{} \cup \xleftrightarrow[I_{\subseteq}]{} \cup \xleftarrow[R_{\supseteq}]{})^* = \xrightarrow[I\backslash R_{\subseteq}]{*} \circ \xleftrightarrow[I_{\subseteq}]{*} \circ \xleftarrow[I\backslash R_{\supseteq}]{*}$. Now, it is easy to prove that the algorithm is a decision procedure for the relation $\xrightarrow[I\backslash R_{\subseteq}]{*} \circ \xleftrightarrow[I_{\subseteq}]{*} \circ \xleftarrow[I\backslash R_{\supseteq}]{*}$ and $Ax \vdash_{POL} s \subseteq t$ is equivalent to $s(\xrightarrow[R_{\subseteq}]{} \cup \xleftrightarrow[I_{\subseteq}]{} \cup \xleftarrow[R_{\supseteq}]{})^* t$. ■

The solution we propose of reducing the strong Church-Rosser property to a local bi-confluence property is inspired mainly by the two solutions known for the equational case. In the following we consider how they can be adapted to bi-rewriting.

Huet (Huet, 1980; Kirchner, 1985a; Jouannaud and Kirchner, 1986) prove that given a set of rules R and equations E such that $\xleftrightarrow[E]{*} \circ \xrightarrow[R]{}$ is terminating, R is strong Church-Rosser modulo E iff all *peaks* and *cliffs* are confluent: $\xleftarrow[R]{} \circ \xrightarrow[R]{} \subseteq \xrightarrow[R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[R]{*}$ and $\xleftrightarrow[E]{} \circ \xrightarrow[R]{} \subseteq \xrightarrow[R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[R]{*}$. Notice that these are sufficient and, what is also important, necessary conditions. Besides, the finiteness of the E-equivalence classes is not required. However, these confluence properties are too strong and can not be reduced to the confluence of critical pairs unless the rules are left-linear.

To overcome this limitation for non-left-linear systems Peterson and Stickel (Peterson and Stickel, 1981) propose the use of a new rewriting relation $E\backslash R$ satisfying $\xrightarrow[R]{} \subseteq \xrightarrow[E\backslash R]{} \subseteq \xleftrightarrow[E]{*} \circ \xrightarrow[R]{}$. They prove that when this relation is E-compatible, that is when $\xleftrightarrow[E]{*} \circ \xrightarrow[R]{} \subseteq \xrightarrow[E\backslash R]{} \circ \xleftrightarrow[E]{*} \circ (\xleftarrow[R]{} \circ \xleftrightarrow[E]{*})^*$, and terminating, then the Church-Rosser property becomes equivalent to the confluence of peaks of the form $\xleftarrow[E\backslash R]{} \circ \xrightarrow[E\backslash R]{} \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$. They also study how a rewriting relation R can be extended to obtain a E-compatible rewriting relation $E\backslash R$ when E is an associative and commutative theory. However, in this case the problem is that the set of critical pairs of the form $t \xleftarrow[E\backslash R]{} u \xrightarrow[E\backslash R]{} v$ is in general infinite.

Jouannaud and Kirchner (Jouannaud and Kirchner, 1986) (see also theorem 4, chapter 2 of (Kirchner, 1985b)) prove that when $\xleftrightarrow[E]{*} \circ \xrightarrow[R]{} \circ \xleftrightarrow[E]{*}$ is terminating then the following three conditions are equivalent

1. Church-Rosser modulo E

$$(\xrightarrow[R]{} \cup \xleftrightarrow[E]{} \cup \xleftarrow[R]{})^* \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$$

2. confluence of (global) peaks and (global) cliffs:

$$\xleftarrow[E\backslash R]{} \circ \xrightarrow[E\backslash R]{} \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$$
$$\xleftrightarrow[E]{} \circ \xrightarrow[E\backslash R]{} \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$$

3. confluence of local peaks and local cliffs:

$$\xleftarrow[R]{} \circ \xrightarrow[E\backslash R]{} \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$$
$$\xleftrightarrow[E]{*} \circ \xrightarrow[E\backslash R]{} \subseteq \xrightarrow[E\backslash R]{*} \circ \xleftrightarrow[E]{*} \circ \xleftarrow[E\backslash R]{*}$$

Then these local confluences can be reduced to critical pairs confluence and to extended rules respectively.

Jouannaud and Kirchner also notice that their theorem is false if we require termination of $\xrightarrow[E\backslash R]{}$ instead of that for $\xleftrightarrow[E]{*} \circ \xrightarrow[R]{} \circ \xleftrightarrow[E]{*}$. As a counter-example we can take the rewriting system $R = E\backslash R = \{b \longrightarrow a, a \longrightarrow d\}$ with $E = \{a = b, b = c\}$. It satisfies local confluence properties and termination of $\xrightarrow[E\backslash R]{}$, but it is not Church-Rosser. However, termination of $\xrightarrow[E\backslash R]{}$ is enough to prove the equivalence between Church-Rosser property and "global" confluence properties. (Evidently, it is not enough to prove equivalence between local and global confluence properties).

As we will see in next subsection, proving confluence of local or of global peaks and cliffs makes no difference, therefore we have chosen this second option because imposes a weaker termination condition. If we would adopt the first option, then we will need a well-founded and E-compatible ordering on terms, i.e. a well-founded ordering on E-equivalence classes of terms. In our case, this E-compatible ordering on terms would be equivalent to requiring termination of

$$\left(\xleftrightarrow[I_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{} \right)^* \circ \left(\xrightarrow[I\backslash R_\subseteq]{} \cup \xrightarrow[I\backslash R_\supseteq]{} \right)$$

As we will see in the following subsection, after proving lemma 4.17, there is no gain choosing this first option.

The following lemma adapts to bi-rewriting this second version of the results of Jouannaud and Kirchner.

Lemma 4.14 *Let* $\xrightarrow[I\backslash R_\subseteq]{}$ *and* $\xrightarrow[I\backslash R_\subseteq]{}$ *be two rewriting relations satisfying* $\xrightarrow[R_\subseteq]{} \subseteq \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[R_\subseteq]{}$ *and* $\xrightarrow[R_\supseteq]{} \subseteq \xrightarrow[I\backslash R_\supseteq]{} \subseteq \xleftrightarrow[I_\supseteq]{*} \circ \xrightarrow[R_\supseteq]{}$. *If their union* $\xrightarrow[I\backslash R_\subseteq]{} \cup \xrightarrow[I\backslash R_\subseteq]{}$ *is terminating then the following three conditions*

$$\left.\begin{array}{l} \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*} \\ \xleftarrow[I\backslash R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*} \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*} \end{array}\right\} \quad \textit{cliffs}$$

$$\xleftarrow[I\backslash R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*} \qquad \textit{peaks}$$

and the strong Church-Rosser property

$$\left(\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{} \right)^* \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*}$$

are equivalent.

Proof: It is evident that the Church-Rosser property implies the three local bi-confluence properties, so we will prove the opposite implication. Such proof is based on the ideas of proof transformation and proof ordering proposed by Bachmair in his thesis (Bachmair, 1991) and in (Bachmair et al., 1986b).

Given a sequence of terms $\langle v_1, \ldots v_n \rangle$, we say that *it is a proof of* $s \subseteq t$ iff $v_1 = s$, $v_n = t$, and for any $i \in [1..n-1]$ we have $v_i \xrightarrow[I\backslash R_\subseteq]{} v_{i+1}$ or $v_i \xleftarrow[I\backslash R_\supseteq]{} v_{i+1}$

or $v_i \xleftrightarrow[I_\subseteq]{+} v_{i+1}$. Notice that we allow to concentrate one or more $\xleftrightarrow[I_\subseteq]{}$ rewriting steps in a single proof step. Evidently, $t \subseteq u$ has a proof iff $t(\xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \cup \xleftarrow[R_\supseteq]{})^* u$.

In the following we define a set of transformations on the proofs of an inclusion. Given a proof transformation rule $\langle s, \overline{t}, u\rangle \Rightarrow \langle s, \overline{v}, u\rangle$, we can use it to transform $\langle \overline{w_1}, s, \overline{t}, u, \overline{w_2}\rangle \Rightarrow \langle \overline{w_1}, s, \overline{v}, u, \overline{w_2}\rangle$. To prove the termination of such transformation relation we associate a multiset $S(\langle v_1, \dots, v_n\rangle)$ of terms to each proof $\langle v_1, \dots, v_n\rangle$ defined as follows.

$$S(\langle v\rangle) = \emptyset$$

$$S(\langle v_1, \dots, v_n\rangle) = S(\langle v_1, \dots, v_{n-1}\rangle) \cup \begin{cases} \{v_{n-1}, v_n\} & \text{if } v_{n-1} \xrightarrow[I\backslash R_\subseteq]{} v_n \\ & \text{or } v_n \xrightarrow[I\backslash R_\supseteq]{} v_{n-1} \\ \{v_{n-1}^2, v_n^2\} & \text{if } v_{n-1} \xleftrightarrow[I_\subseteq]{+} v_n \end{cases}$$

where $\cup$ denotes the multiset union operator and superscripts denote the number of occurrences of an element in a multiset. We define a well-founded ordering $\succ$ on these term multisets as the multiset extension of the order relation $\xrightarrow[I\backslash R_\subseteq]{+} \cup \xrightarrow[I\backslash R_\supseteq]{+}$ which we have supposed terminating. This ordering on associated multisets defines a well-founded ordering on proofs. Notice that this ordering is monotonic, i.e. if $S(\langle s, \overline{t}, u\rangle) \succ S(\langle s, \overline{v}, u\rangle)$, then $S(\langle \overline{w_1}, s, \overline{t}, u, \overline{w_2}\rangle) \succ S(\langle \overline{w_1}, s, \overline{v}, u, \overline{w_2}\rangle)$. This is a key point to prove that if any proof transformation rule $\langle s, \overline{t}, u\rangle \Rightarrow \langle s, \overline{v}, u\rangle$ satisfies $S(\langle s, \overline{t}, u\rangle) \succ S(\langle s, \overline{v}, u\rangle)$ then the proof transformation relation is terminating.

If cliffs are bi-confluent, then for any cliff $s \xleftrightarrow[I_\subseteq]{+} t \xrightarrow[I\backslash R_\subseteq]{} u$ we have

$$s \xrightarrow[I\backslash R_\subseteq]{} v_1 \cdots v_{p-1} \xrightarrow[I\backslash R_\subseteq]{} v_p \xleftrightarrow[I_\subseteq]{*} w_q \xleftarrow[I\backslash R_\supseteq]{} w_{q-1} \cdots w_1 \xleftarrow[I\backslash R_\supseteq]{} u$$

and we can apply one of the following proof transformations rules to eliminate it

$$\begin{array}{ll} \langle s, t, u\rangle \Rightarrow \langle s, v_1, \dots, v_p, w_q, \dots, w_1, u\rangle & \text{if } v_p \xleftrightarrow[I_\subseteq]{+} w_q \\ \langle s, t, u\rangle \Rightarrow \langle s, v_1, \dots, v_{p-1}, w_q, \dots, w_1, u\rangle & \text{if } s \xrightarrow[I\backslash R_\subseteq]{+} v_p = w_q \\ \langle s, t, u\rangle \Rightarrow \langle s, w_{q-1}, \dots, w_1, u\rangle & \text{if } s = v_p = w_q \xleftarrow[I\backslash R_\supseteq]{+} u \\ \langle s, t, u\rangle \Rightarrow \langle s\rangle & \text{if } s = v_p = w_q = u \end{array}$$

where $p, q \geq 0$, except in the second rule where $p \geq 1$, and the third rule where $q \geq 1$. Now, taking into account that $s \succ v_1 \succ \cdots \succ v_p$ and $t \succ u \succ w_1 \succ \cdots \succ w_q$, we can prove that the multiset associated to the left part of the rules $S(\langle s, t, u\rangle) = \{s^2, t^3, u\}$ is strictly greater than the multisets associated to the right part of the rules, which are respectively:

$$\begin{array}{l} S(\langle s, v_1, \dots, v_p, w_q, \dots, w_1, u\rangle) = \{s, v_1^2, \dots, v_p^2, w_q^2, w_2^2, \dots, w_1^2, u\} \cup {}^{16}\{v_p, w_q\} \\ S(\langle s, v_1, \dots, v_{p-1}, w_q, \dots, w_1, u\rangle) = \{s, v_1^2, \dots, v_{p-1}^2, w_q^2, \dots, w_1^2, u\} \\ S(\langle s, w_{q-1}, \dots, w_1, u\rangle) = \{s, w_{q-1}^2, \dots, w_1^2, u\} \\ S(\langle s\rangle) = \emptyset \end{array}$$

Similarly, if peaks are bi-confluent, then we can also apply the same proof transformations rule to any peak $s \xleftarrow[I\setminus R_\supseteq]{} t \xrightarrow[I\setminus R_\subseteq]{} u$. And, taking into account that now $t \succ s \succ v_1 \succ \cdots \succ v_p$ and $t \succ u \succ w_1 \succ \cdots \succ w_q$, we can also prove that the multiset associated to the left part of the rule, now $S(\langle s,t,u\rangle) = \{s, t^2, u\}$ is also strictly greater than the multisets associated to the corresponding right parts of the rules.

Evidently, if we iterate this process, the resulting canonical (normal) proof will not contain any cliffs nor peaks. Therefore it will be of the form $\xrightarrow[I\setminus R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\setminus R_\supseteq]{*}$. The process can not be applied infinitely, because the transformation relation is terminating. We conclude that if $s \subseteq t$ has a proof, then it has a canonical proof of the form $s \xrightarrow[I\setminus R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\setminus R_\supseteq]{*} t$. Therefore, the Church-Rosser property holds for these rewriting relations. ■

Now, the logical process would be to reduce the bi-confluence of peaks of the form $\xleftarrow[I\setminus R_\supseteq]{} \circ \xrightarrow[I\setminus R_\subseteq]{}$ to the bi-confluence of peaks of the form $\xleftarrow[I\setminus R_\supseteq]{} \circ \xrightarrow[R_\subseteq]{}$ or $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[I\setminus R_\subseteq]{}$, as Jouannaud and Kirchner suggested for the equational case. However, as the following counter-example shows, not any definition of $\xrightarrow[I\setminus R]{}$ satisfying $\xrightarrow[R]{} \subseteq \xrightarrow[I\setminus R]{} \subseteq \xleftrightarrow[I]{*} \circ \xrightarrow[R]{}$ permits such reduction, unless we require termination of $(\xleftrightarrow[I_\subseteq]{} \cup \xleftrightarrow[I_\supseteq]{})^* \circ (\xrightarrow[I\setminus R_\subseteq]{} \cup \xrightarrow[I\setminus R_\supseteq]{})$.

Counter-example. Consider the rewriting relations defined by the following sets of rules.

$$
\begin{aligned}
I_\subseteq &= \{a_1 \xleftrightarrow{\subseteq} b, b \xleftrightarrow{\subseteq} a_2\}\\
R_\subseteq &= \{a_1 \xrightarrow{\subseteq} b, a_2 \xrightarrow{\subseteq} c_2\}\\
R_\supseteq &= \{a_2 \xrightarrow{\subseteq} b, a_1 \xrightarrow{\subseteq} c_1\}
\end{aligned}
$$

$$\begin{tikzcd}
c_1 & a_1 \arrow[l, "R_\supseteq"'] \arrow[r, leftrightarrow, "I_\subseteq"] \arrow[r, bend right, "R_\subseteq"'] & b & a_2 \arrow[l, leftrightarrow, "I_\subseteq"'] \arrow[l, bend left, "R_\supseteq"] \arrow[r, "R_\subseteq"] & c_2
\end{tikzcd}$$

If we define $\xrightarrow[I\setminus R_\subseteq]{} \overset{def}{=} \xrightarrow[R_\subseteq]{} \cup \xleftrightarrow[I_\subseteq]{} \circ \xrightarrow[R_\subseteq]{}$ and $\xrightarrow[I\setminus R_\supseteq]{} \overset{def}{=} \xrightarrow[R_\supseteq]{} \cup \xleftrightarrow[I_\supseteq]{} \circ \xrightarrow[R_\supseteq]{}$, we will obtain two rewriting relations such that $\xrightarrow[I\setminus R_\subseteq]{} \cup \xrightarrow[I\setminus R_\supseteq]{}$ is terminating and the properties $\xrightarrow[R]{} \subseteq \xrightarrow[I\setminus R]{} \subseteq \xleftrightarrow[I]{*} \circ \xrightarrow[R]{}$ hold, however, although any peak of the form $\xleftarrow[I\setminus R_\supseteq]{} \circ \xrightarrow[R_\subseteq]{}$ or $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[I\setminus R_\subseteq]{}$ and any cliff is bi-confluent, there is a peak $c_1 \xleftarrow[I\setminus R_\supseteq]{} b \xrightarrow[I\setminus R_\subseteq]{} c_2$ which is not bi-confluent.

Fortunately, the method of ***rule extensions*** and the concrete definition of the relation $\xrightarrow[I\setminus R]{}$ ensures that, if inclusions in I are linear, then $\xleftrightarrow[I]{*}$ and $\xrightarrow[I\setminus R]{}$ commute, i.e. $\xleftrightarrow[I]{*} \circ \xrightarrow[I\setminus R]{} \subseteq \xrightarrow[I\setminus R]{} \circ \xleftrightarrow[I]{*}$. This property is stronger than the confluence of cliffs, and permits the desired reduction. However, such method takes into account the structure of terms, so we will describe it in the next subsection.

[16]Notice that in this case we can have $s = v_p$, $u = w_q$ or both together. With such union we capture four cases.

4.3.2 From Local Bi-Confluence to (Extended) Critical Pairs

Till now, we have studied Church-Rosser, termination and bi-confluence properties in the framework of relational algebra (Bäumer, 1992). All proofs are done without any reference to the structure of terms. In the following, we will consider the term structure in order to reduce the bi-confluence properties to the bi-confluence of critical pairs and rule extensions.

We begin defining the rewrite relations $I\backslash R_{\subseteq}$ and $I\backslash R_{\supseteq}$ that were only axiomatically characterized by $\xrightarrow[R]{} \subseteq \xrightarrow[I\backslash R]{} \subseteq \xleftrightarrow[I]{*} \circ \xrightarrow[R]{}$ in the previous subsection.

Definition 4.15 *We say that s $R_{\subseteq}$*-**rewrites** *to t* **modulo** *$I_{\subseteq}$, written $s \xrightarrow[I\backslash R_{\subseteq}]{} t$, iff there exists a rule $l \longrightarrow r$ in $R_{\subseteq}$, an occurrence p in s, and a substitution σ such that $s|_p \xleftrightarrow[I_{\subseteq}]{*} \sigma(l)$ and $t = s[\sigma(r)]_p$.*
Similarly for s $R_{\supseteq}$-rewrites to t modulo $I_{\supseteq}$, written $s \xrightarrow[I\backslash R_{\supseteq}]{} t$.

With this definition $\xrightarrow[I\backslash R_{\subseteq}]{}$ really verifies $\xrightarrow[R_{\subseteq}]{} \subseteq \xrightarrow[I\backslash R_{\subseteq}]{} \subseteq \xleftrightarrow[I_{\subseteq}]{*} \circ \xrightarrow[R_{\subseteq}]{}$ although in general $\xleftrightarrow[I_{\subseteq}]{*} \circ \xrightarrow[R_{\subseteq}]{} \not\subseteq \xrightarrow[I\backslash R_{\subseteq}]{}$. The choice of such definition is motivated, as in the equational case, by the fact that local bi-confluence of peaks $\xleftarrow[R_{\supseteq}]{} \circ \xrightarrow[I\backslash R_{\subseteq}]{}$ and $\xleftarrow[I\backslash R_{\supseteq}]{} \circ \xrightarrow[R_{\subseteq}]{}$ can be reduce to the bi-confluence of a selected set of critical pairs.

We will use the notions of E-matching and E-unification from (Peterson and Stickel, 1981) but adapted to bi-rewriting. Given two terms s and t, we say that s I-matches t iff there exists a substitution σ such that $s \xleftrightarrow[I_{\subseteq}]{*} \sigma(t)$, and s I^{-1}-matches t iff there exists a substitution σ such that $s \xleftrightarrow[I_{\supseteq}]{*} \sigma(t)$. We say that s I-unifies with t iff there exists a substitution σ such that $\sigma(s) \xleftrightarrow[I_{\subseteq}]{*} \sigma(t)$. This substitution is said to be a minimum unifier if for any other unifier σ', if $\sigma = \rho\circ\sigma'$, then ρ is a renaming of variables. Notice that, since $\xleftrightarrow[I_{\subseteq}]{}$ is not necessarily symmetric, I-matching and I^{-1}-matching are in general different non-symmetric relations, and I-unification is neither a symmetric relation. We will suppose in the following that I-unification and I and I^{-1}-matching are decidable.

As in the equational case (to prove bi-confluence of cliffs or E-compatibility), we will prove the commutativity properties by means of the rule extension and the extensionally closed property defined as follows.

Definition 4.16 *Given an inclusion $l \subseteq r$ in I, and a rule $s \xrightarrow{\subseteq} t$ in $R_{\subseteq}$, if $r|_p$ I-unify with s, being σ a minimum unifier, and $r|_p$ is neither a variable nor equal to r, then we say that $\sigma(l) \xrightarrow{\subseteq} \sigma(r[t]_p)$ is a* **right-I-extended** *rule of $R_{\subseteq}$.*
Given a set of rules $R_{\subseteq}$ and inclusions I, $R_{\subseteq}$ is said to be **right-I-extensionally closed** *iff any right-I-extended rule $l \xrightarrow{\subseteq} r$ of $R_{\subseteq}$ satisfies $l \xrightarrow[I\backslash R_{\subseteq}]{} \circ \xleftrightarrow[I_{\subseteq}]{*} r$.*
We define left-I-extended rule and left-I-extensionally closed similarly changing $\subseteq$ by $\supseteq$ and "$r|_p$ I-unify with s" by "s I-unify with $r|_p$".

Notice that in the previous definition, to consider a rewriting system extensionally closed, we require any rule extension $l \xrightarrow{\subseteq} r$ to satisfy $l \xrightarrow[I\backslash R_{\subseteq}]{} \circ \xleftrightarrow[I_{\subseteq}]{*} r$, it is not enough to require the pair $l \subseteq r$ to be bi-confluent.

Since $\xleftrightarrow[I]{}$ may be non-symmetric, we have had to distinguish between right- and left-extensionality in the previous definition. We will use a completion procedure to ensure that the final bi-rewriting system satisfies that $R_{\subseteq}$ is right-I-extensionally closed, and that $R_{\supseteq}$ is left-I-extensionally closed.

The following lemma states that, if all inclusions in I are linear, then the extensionally closed property ensures the commutativity of $\xleftrightarrow[I]{*}$ and $\xrightarrow[I\backslash R]{}$. Notice that this property is stronger than the bi-confluence of cliffs required in the previous subsection.

Lemma 4.17 Critical cliff lemma. *If all inclusions in I are linear, and $R_{\subseteq}$ is right-I-extensionally closed, then $\xleftrightarrow[I_{\subseteq}]{*}$ and $\xrightarrow[I\backslash R_{\subseteq}]{}$ commute, i.e. $\xleftrightarrow[I_{\subseteq}]{*} \circ \xrightarrow[I\backslash R_{\subseteq}]{} \subseteq \xrightarrow[I\backslash R_{\subseteq}]{} \circ \xleftrightarrow[I_{\subseteq}]{*}$.*
Similarly for $\xleftrightarrow[I_{\supseteq}]{*}$ and $\xrightarrow[I\backslash R_{\supseteq}]{}$ if the later is left-I-extensionally closed.

Proof: The conclusion of the lemma is equivalent to $\xleftrightarrow[I]{} \circ \xrightarrow[I\backslash R]{} \subseteq \xrightarrow[I\backslash R]{} \circ \xleftrightarrow[I]{*}$. Suppose $a \xleftrightarrow[I_{\subseteq}]{} b \xrightarrow[I\backslash R]{} c$, then there exists a position p_1 in a, a position p_2 in b, two substitutions σ_1 and σ_2, an inclusion $s \xleftrightarrow[I_{\subseteq}]{} t$ in I and a rule $l \xrightarrow[R]{} r$ in R such that:

$$\begin{array}{ll} a|_{p_1} = \sigma_1(s) & b|_{p_2} \xleftrightarrow[I]{*} \sigma_2(l) \\ b = a[\sigma_1(t)]_{p_1} & c = b[\sigma_2(r)]_{p_2} \end{array}$$

We have to consider the following three cases.

case $p_1|p_2$ Applying the definition of rewriting modulo we prove that $a \xrightarrow[I\backslash R]{} a[\sigma_2(r)]_{p_2}$. Now, if both redex do not overlap then $a[\sigma_2(t)]_{p_2} = b[\sigma_1(l)]_{p_1}[\sigma_2(t)]_{p_2} = b[\sigma_2(t)]_{p_2}[\sigma_1(l)]_{p_1} = c[\sigma_1(l)]_{p_1}$. Finally, we have $a \xrightarrow[I\backslash R]{} a[\sigma_2(r)]_{p_2} = c[\sigma_1(l)]_{p_1} \xleftrightarrow[I]{} c$.

case $p_1 \prec p_2$ Let v satisfy $p_2 = p_1 \cdot v$. We have $\sigma_1(t)|_v \xleftrightarrow[I]{*} \sigma_2(l)$. There are two possibilities:

- If v is a position in t, and $t|_v$ is not a variable, we are in the conditions of definition 4.16, i.e. $t|_v$ I-unify with l being σ a minimum unifier (smaller than $\sigma_1 \circ \sigma_2$, so $\sigma_1 \circ \sigma_2 = \rho \circ \sigma$ for some substitution ρ), and we can generate an extensional rule with $s \subseteq t$ and $l \longrightarrow r$. Now, if R is I-extensionally closed, then this rule, or a generalization of this rule, will belong to R and we can rewrite a into c using it at position p_1 with the substitution ρ.
- Otherwise, there exist two occurrences v_1 and v_2 satisfying $p_1 \cdot v_1 \cdot v_2 = p_2$ and being $t|_{v_1} = x$ a variable. If all inclusions in I are right-linear then $t|_{v_1}$ is the only occurrence of x in t, moreover if all inclusions are left-linear then x occurs once in s. Let v_1' be this occurrence of x in

s. We have $a|_{p_1\cdot v_1'\cdot v_2} \xleftrightarrow[I]{*} \sigma_2(l)$ and therefore $a \xrightarrow[I\backslash R]{} a[\sigma_2(r)]_{p_1\cdot v_1'\cdot v_2}$. Finally, we prove that $a[\sigma_2(r)]_{p_1\cdot v_1'\cdot v_2} \xleftrightarrow[I]{} c$ using the same equation $s \subseteq t$, at the same position p_1, but with a substitution σ_1' defined as $\sigma_1'(y) = \sigma_1(y)$ for any $y \neq x$, and $\sigma_1'(x) = \sigma_1(x)[\sigma_2(r)]_{v_2}$. Notice that is in this case, with variable overlapping, when we have to require both left- and right-linearity of $s \subseteq t$.

case $p_1 \succeq p_2$ Let v be the occurrence such that $p_2\cdot v = p_1$. We prove $a|_{p_2} \xleftrightarrow[I]{} b|_{p_2}$ using the equation $s \subseteq t$ at position v with the substitution σ_1. Now, as far as $a|_{p_2} \xleftrightarrow[I]{} b|_{p_2} \xleftrightarrow[I]{*} \sigma_2(l)$, we have $a \xrightarrow[I\backslash R]{} c$ using the rule $l \longrightarrow r$ and σ_2 at position p_2.

■

Notice that the conclusion of the previous lemma, not only ensures the bi-confluence of cliffs, but also allows to reduce the bi-confluence of peaks of the form $\xleftarrow[I\backslash R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{}$ to the confluence of the peaks of the form $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{}$ or $\xleftarrow[I\backslash R_\supseteq]{} \circ \xrightarrow[R_\subseteq]{}$ using the following sequence of inclusions

$$\begin{array}{ll}
\xleftarrow[I\backslash R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xleftarrow[R_\supseteq]{} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xleftarrow[R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{} \circ \xleftrightarrow[I_\subseteq]{*} \subseteq \cdots & \\
\xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \subseteq \cdots & \text{if peaks are bi-confluent} \\
\xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*} \circ \xleftarrow[I\backslash R_\supseteq]{*} & \text{if cliffs commute}
\end{array}$$

Notice also that like in (Peterson and Stickel, 1981), and differently from (Jouannaud and Kirchner, 1986), the inclusions in I are required to be (both left- and right-) linear.

There is a way to avoid left-linearity in the previous lemma. The reader may check that to prove local commutativity $\xleftrightarrow[I_\subseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{} \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*}$ (where several $\xrightarrow[I\backslash R_\subseteq]{}$ rewriting steps are allowed to eliminate the variable overlapping cliff) we only would need right-linearity. If we use a well-founded and $(\xleftrightarrow[I_\subseteq]{*} \cup \xleftrightarrow[I_\supseteq]{*})$ compatible ordering to prove the termination of the relation $\xrightarrow[I\backslash R_\subseteq]{} \cup \xrightarrow[I\backslash R_\supseteq]{}$ (like it is done in the equationa case), we can reduce the (global) commutativity property $\xleftrightarrow[I_\subseteq]{*} \circ \xrightarrow[I\backslash R_\subseteq]{*} \subseteq \xrightarrow[I\backslash R_\subseteq]{*} \circ \xleftrightarrow[I_\subseteq]{*}$ to the previous local commutativity. Unfortunately, it seems that we can not avoid right-linearity unless we use some kind of "extended critical cliff". However, even not being necessary in this case, left-linearity will be necessary to prove commutativity of $\xleftrightarrow[I_\supseteq]{*}$ and $\xrightarrow[I\backslash R_\supseteq]{*}$. Therefore, there is no gain in choosing this alternative, because linearity is always necessary unless we use some kind of extended critical cliffs.

For the bi-confluence of peaks we use a definition of (extended) critical pairs similar to the one introduced in the previous section.

Definition 4.18 *If $l \xrightarrow{\subseteq} r \in R_\subseteq$ and $s \xrightarrow{\supseteq} t \in R_\supseteq$ are two rewriting rules normalized apart, and p is a position in s, then*

(i) *if $s|_p$ is not a variable and σ is a minimum I-unifier of $s|_p$ and l, then*

$$\sigma(t) \subseteq \sigma(s[r]_p) \ \in \ ECP(I\backslash R_{\subseteq}, R_{\supseteq})$$

is a **(standard) critical pair,**

(ii) *if $s|_p = x$ is a repeated variable in s, F is a term containing only fresh variables, q is a position in F, and $l \xrightarrow[I\backslash R_{\supseteq}]{*} \circ \xleftrightarrow[I_{\supseteq}]{*} r$ does not hold, then*

$$\sigma(t) \subseteq \sigma(s[F[r]_q]_p) \ \in \ ECP(I\backslash R_{\subseteq}, R_{\supseteq})$$

is an **(extended) critical pair** *where the domain of σ is $\{x\}$ and $\sigma(x) = F[l]_q$.*

The set $ECP(R_{\subseteq}, I\backslash R_{\supseteq})$ can be defined similarly.

Again we have had to introduce critical pair schemes which may generate infinitely many critical pairs. Using this extended definition of critical pairs and the definition of extensionally closed bi-rewriting system we can prove the following theorem which characterizes the strong Church-Rosser property of a $\langle R_{\subseteq}, R_{\supseteq}\rangle$ bi-rewriting system modulo I.

Theorem 4.19 Critical pair lemma. *Given two sets of rules $R_{\subseteq}$ and $R_{\supseteq}$ and a set of inclusions I, if $\xrightarrow[I\backslash R_{\subseteq}]{} \cup \xrightarrow[I\backslash R_{\supseteq}]{}$ is terminating, $\xrightarrow[I\backslash R_{\subseteq}]{}$ is right-I-extensionally closed, $\xrightarrow[I\backslash R_{\supseteq}]{}$ is left-I-extensionally closed, all inclusions in I are linear, and all standard and extended critical pairs $ECP(I\backslash R_{\subseteq}, R_{\supseteq})$ and $ECP(R_{\subseteq}, I\backslash R_{\supseteq})$ are bi-confluent, then $\langle I\backslash R_{\subseteq}, I\backslash R_{\supseteq}\rangle$ is (strongly) Church-Rosser modulo I.*

Proof: We use lemma 4.14 to prove the Church-Rosser property. We are in the conditions of lemma 4.17, therefore we can ensure the bi-confluence of cliffs. Furthermore, as we have already commented, we can reduce the confluence of peaks to the confluence of peaks of the form $\xleftarrow[R_{\supseteq}]{} \circ \xrightarrow[I\backslash R_{\subseteq}]{}$. Let's study then this condition.

Suppose we have $a \xleftarrow[R_{\supseteq}]{} b \xrightarrow[I\backslash R_{\subseteq}]{} c$ where reductions take place at two positions p_1 and p_2 of b, using two rules $s \xrightarrow{\supseteq} t$ and $l \xrightarrow{\subseteq} r$ respectively:

$$\begin{array}{ll} b|_{p_1} = \sigma_1(s) \qquad & b|_{p_2} \xleftrightarrow[I_{\subseteq}]{*} \sigma_2(l) \\ a = b[\sigma_1(t)]_{p_1} & c = b[\sigma_2(r)]_{p_2} \end{array}$$

Three cases must be considered:

case $p_1|p_2$ As in the commutativity case, we prove that both reductions can be permuted and we reduce $a = b[\sigma_1(t)]_{p_1} \xrightarrow[I\backslash R_{\subseteq}]{} c[\sigma_1(t)]_{p_1} = b[\sigma_2(r)]_{p_2}[\sigma_1(t)]_{p_1}$ and $c = b[\sigma_2(r)]_{p_2} \xrightarrow[R_{\supseteq}]{} a[\sigma_2(r)]_{p_2} = b[\sigma_1(t)]_{p_1}[\sigma_2(r)]_{p_2}$ to the same term.

case $p_1 \prec p_2$ Let v be the occurrence such that $p_2 = p_1 \cdot v$. We have $\sigma_1(s)|_v \xleftrightarrow[I_{\subseteq}]{*} \sigma_2(l)$. There are three possibilities:

- Position v is a non-variable occurrence of s.

  Then the divergence $\sigma_1(t) \xleftarrow[R_\supseteq]{} \sigma_1(s) \xrightarrow[I\backslash R_\subseteq]{} \sigma_1(s)[\sigma_2(r)]_v$ we are considering is an instance of the standard critical pair $\sigma(t) \subseteq \sigma(s)[\sigma(r)]_v$ generated by I-unifying $s|_v$ and l, and therefore it is bi-confluent as far as any standard critical pair in $ECP(I\backslash R_\subseteq, R_\supseteq)$ is bi-confluent.

- Subterm $s|_v$ is a repeated variable x of s, or the occurrence of v in $\sigma_1(s)|_v$ is bellow a repeated variable x of s, i.e. there exists a pair of positions $v_1 \cdot v_2 = v$ such that $s|_{v_1}$ is a repeated variable x of s.

  In this case the divergence being studied is an instance of the extended critical pair $t[x \mapsto F[l]_{v_2}] \subseteq (s[x \mapsto F[l]_{v_2}])[F[r]_{v_2}]_{v_1}$, and therefore it will be bi-confluent if any extended critical pair of $ECP(I\backslash R_\subseteq, R_\supseteq)$ is bi-confluent.

- Subterm $s|_v$ is a non-repeated variable or $\sigma_1(s)|_v$ is bellow a non-repeated variable x of s, i.e. there exist two positions such that $v = v_1 \cdot v_2$ and $s|_{v_1} = x$ is a non-repeated variable of s.

  In this case we can rewrite a and c into a common term in the following way. We apply the rewriting step $\sigma_1(x) = \sigma_1(s)|_{v_1} \xrightarrow[I\backslash R_\subseteq]{} \sigma_1(x)[\sigma_2(r)]_{v_1}$ to any occurrence of x in t, i.e. of $\sigma_1(x)$ in $a = b[\sigma_1(t)]_{p_1}$. On the other hand, we apply the rule $s \xrightarrow{\supseteq} t$ to the position p_1 of c, but using the substitution σ_1' defined as $\sigma_1'(y) = \sigma_1(y)$ for any $y \neq x$ and $\sigma_1'(x) = \sigma_1(x)[\sigma_2(r)]_{v_2}$ instead of σ_1. In both cases we obtain the same result.

case $p_1 \succeq p_2$ Here we can suppose that the $\xleftrightarrow[I_\subseteq]{}$ rewriting steps, between the $\xleftarrow[R_\supseteq]{}$ and the $\xrightarrow[R_\subseteq]{}$ rewriting steps, occur bellow p_1 and p_2, like it is also argued in the equational case: if the divergence $a \subseteq c$ being studied is generated as $a \xleftarrow[I\backslash R_\supseteq]{} b \xleftrightarrow[I_\subseteq]{*} b[\sigma_2(l)]_{p_2} \xrightarrow[R_\subseteq]{} c$, we can use lemma 4.17 to commute the step $\xleftarrow[I\backslash R_\supseteq]{}$ and the steps $\xleftrightarrow[I_\subseteq]{*}$ obtaining $a \xleftrightarrow[I_\subseteq]{*} d \xleftarrow[I\backslash R_\supseteq]{} b[\sigma_2(l)]_{p_2} \xrightarrow[R_\subseteq]{} c$ for some term d. Now, we only need to prove the bi-confluence of the peak $d \xleftarrow[I\backslash R_\supseteq]{} b[\sigma_2(l)]_{p_2} \xrightarrow[R_\subseteq]{} c$. However, this situation is completely equivalent to the previous second case $p_1 \prec p_2$ if we change $\xrightarrow[R_\supseteq]{}$ by $\xrightarrow[R_\subseteq]{}$ and $\xrightarrow[I\backslash R_\subseteq]{}$ by $\xrightarrow[I\backslash R_\supseteq]{}$, therefore all divergences of this kind will be bi-confluent if all critical pairs in $ECP(R_\subseteq, I\backslash R_\supseteq)$ are bi-confluent.

Notice that at this point it becomes clear that if the commutativity of $\xrightarrow[I\backslash R]{}$ and $\xleftrightarrow[I]{*}$ did not hold and we only had bi-confluence of cliffs, then we could not suppose that the $\xleftrightarrow[I]{*}$ rewriting steps in the peak $\xleftarrow[R_\supseteq]{} \circ \xrightarrow[I\backslash R_\subseteq]{}$ always occur bellow the innermost of the redexes, and we would have to change the definition of critical pair.

■

4.4 Three Examples: Towards a Completion Procedure

As we said in the previous sections, bi-rewriting compared with equational rewriting, faces the extra difficulty of a possible infinite set of critical pairs. Non-left-linear rules may generate what we called critical pair schemes (see definitions 4.9 and 4.18). In this section instead of giving the completion procedure we sketch out the possibilities of completion à la Knuth-Bendix of three examples of bi-rewriting systems by means of rule schemes. Other completion methods, like unfailing completion (Bachmair et al., 1989b) are also suitable of being applied to automate the deduction in theories with monotonic order relations, using the same notion of extended critical pair.

4.4.1 Inclusion Theory of the Union Operator

The inclusions defining the theory of the union operator can be oriented following a simplification ordering as follows:

$$\begin{array}{ll} r_1 & X \cup X \xrightarrow{\subseteq} X \\ r_2 & X \cup Y \xrightarrow{\supseteq} X \\ r_3 & X \cup Y \xrightarrow{\supseteq} Y \end{array}$$

Although the standard critical pairs (scp) of this system are bi-confluent, the presence of the non-left-linear rule $X \cup X \xrightarrow{\subseteq} X$ also makes necessary the consideration of the extended critical pairs (ecp). We will do this in two steps. First, we consider scp and the finite subset of ecp of the particular form $\langle \sigma(t), \sigma(s[r]_p) \rangle$ where $s|_p = x$ is a repeated variable in the non-left-linear rule $\langle s \xrightarrow{\subseteq} t \rangle \in R_{\subseteq}$, $\langle l \xrightarrow{\supseteq} r \rangle \in R_{\supseteq}$ being the other rule, and σ substitutes x by l. It corresponds to the general extended critical pair definition where the context $F[\cdot]_p$ is a hole $[\cdot]$ itself. Using the standard Knuth-Bendix completion procedure and a reduction ordering, we generate, among others, the following rules:

$$\begin{array}{lll} r_4 & Y \cup (X \cup Y) \xrightarrow{\subseteq} X \cup Y & \text{ecp from } r_1 \text{ and } r_3 \\ r_5 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y & \text{scp from } r_2 \text{ and } r_4 \\ \\ r_6 & (X \cup Y) \cup Y \xrightarrow{\subseteq} X \cup Y & \text{ecp from } r_1 \text{ and } r_3 \\ r_7 & (X \cup Y) \cup (Y \cup Z) \xrightarrow{\subseteq} X \cup (Y \cup Z) & \text{ecp from } r_2 \text{ and } r_6 \\ r_8 & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) & \text{scp from } r_3 \text{ and } r_7 \end{array}$$

Rules r_5 and r_8, corresponding to the commutativity and associativity (AC) properties of the union, make redundant any other rule generated by the subset of ecp we are considering. It is well known that these rules can not be oriented in a reduction ordering. This fact makes necessary the use of $\langle \{r_1\}, \{r_2\} \rangle$ bi-rewriting modulo $I = \{r_5, r_8\}$. Notice that in this case the relation defined by non-orientable rules is symmetric, i.e. $\xleftrightarrow[I_{\subseteq}]{*} = \xleftrightarrow[I_{\supseteq}]{*}$, thus we can use the standard algorithms of AC-matching and AC-unification, as well as the flat notation for the infix operator $\cup$.

Let's consider now the general form of ecp, i.e. $\langle \sigma(t), \sigma(s[F[r]_q]_p)\rangle$ where $F[\cdot]_p$ is a context and σ substitutes $s|_p = x$ by $F[l]_q$. Using them we generate an extended critical pair which is made bi-confluent adding the following rule scheme:

$$r_9 \quad F[X] \cup F[X \cup Y] \xrightarrow{\subseteq} F[X \cup Y] \quad \text{ecp from } r_1 \text{ and } r_2$$

The orientation of this rule does not depend on the instance we take of the critical pair scheme, and it will be the same for any simplification ordering. This rule scheme generates the following scp:

$$F[X] \cup F[Y] \subseteq F[X \cup Y] \quad \text{scp from } r_2 \text{ and } r_9$$

Now, the orientation of this critical pair depends on the reduction ordering being used. If we use a lexicographic path ordering where $\cup$ is greater than any other symbol of the signature, then it will be oriented as follows for any instance of the critical pair.

$$r_{10} \quad F[X] \cup F[Y] \xrightarrow{\subseteq} F[X \cup Y] \quad \text{from } r_2 \text{ and } r_9$$

Now r_9 is subsumed by r_1 and r_{10}.

Notice that we are dealing with rule schemes instead of ordinary rules, thus we can not continue the completion process unless we have a critical pair definition for rule schemes.

The repeated context $F[\cdot]$ in the left hand side of the rule originates a problem similar to the one caused by non-left-linear rules. We can consider the following particular form of r_{10}, where we suppose that $F[\cdot]$ is a context containing $X' \cup Y'$ as a subexpression, i.e. $F[\cdot] \overset{def}{=} G[X' \cup Y', \cdot]$.

$$r_{11} \quad G[X' \cup Y', X] \cup G[X' \cup Y', Y] \xrightarrow{\subseteq} G[X' \cup Y', X \cup Y]$$

This instantiation of the rule scheme r_{10} generates new non-confluent critical pairs with r_1, which introduces the following rule schemes:

$$G[X', X] \cup G[X' \cup Y', Y] \xrightarrow{\subseteq} G[X' \cup Y', X \cup Y]$$
$$G[X', X] \cup G[Y', Y] \xrightarrow{\subseteq} G[X' \cup Y', X \cup Y]$$

It can be induced then that the completion process would introduce infinitely many rule schemes with the form:

$$r_{12} \quad G[X_1, \ldots, X_n] \cup G[Y_1, \ldots, Y_n] \xrightarrow{\subseteq} G[X_1 \cup Y_1, \ldots, X_n \cup Y_n]$$

for any $n > 0$.

If we are interested in an unfailing completion procedure, the fact that this set of rules would be infinite is not relevant, but we can not obtain a canonical bi-rewriting system (in the sense of Knuth-Bendix completion) in this way. However, in this case, if the signature $\mathcal{F}$ is finite, these (infinite) set of rule schemes will be subsumed by the following (finite) set of rules:

$$r_{13}^{(f)} \quad f(X_1, \ldots, X_n) \cup f(X'_1, \ldots, X'_n) \xrightarrow{\subseteq} f(X_1 \cup X'_1, \ldots, X_n \cup X'_n)$$

for any $n > 0$ and any $f \in \mathcal{F}_n$.

To prove this result we decompose an application of the rule scheme r_{12} into simple applications of the rules r_{13} using the following compositional property:

$$\begin{array}{rcl} F[G[X_1 \ldots X_n]] \cup F[G[Y_1 \ldots Y_n]] & \xrightarrow{\subseteq} & F[G[X_1 \ldots X_n] \cup G[Y_1 \ldots Y_n]] \\ & \xrightarrow{\subseteq} & F[G[X_1 \cup Y_1 \ldots X_n \cup Y_n]] \end{array}$$

Finally, using this *"manual"* completion process we obtain the canonical $\langle R_\subseteq, R_\supseteq \rangle$ bi-rewriting modulo I system shown in figure 4.3. Rules r_1^{ext} and r_{13}^{ext} are the I-extensions of the rules r_1 and r_{13}.

$$R_\subseteq = \left\{ \begin{array}{ll} r_1 & X \cup X \xrightarrow{\subseteq} X \\ r_1^{ext} & X \cup X \cup Y \xrightarrow{\subseteq} X \cup Y \\ \forall f \in \mathcal{F}^n & \\ r_{13} & f(X_1 \ldots X_n) \cup f(Y_1 \ldots Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1 \ldots X_n \cup Y_n) \\ r_{13}^{ext} & f(X_1 \ldots X_n) \cup f(Y_1 \ldots Y_n) \cup Z \xrightarrow{\subseteq} \\ & \qquad \xrightarrow{\subseteq} f(X_1 \cup Y_1 \ldots X_n \cup Y_n) \cup Z \end{array} \right.$$

$$R_\supseteq = \left\{ \begin{array}{ll} r_2 & X \cup Y \xrightarrow{\supseteq} X \end{array} \right.$$

$$I = \left\{ \begin{array}{ll} r_5 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y \\ r_8 & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) \end{array} \right.$$

Figure 4.3: A canonical bi-rewriting system for the inclusion theory of the union.

4.4.2 Inclusion Theory of Non-Distributive Lattices

The presentation of non-distributive lattices theory may be given by the following set of inclusions:

$$\begin{array}{ll} X \cup X \subseteq X & X \subseteq X \cap X \\ X \subseteq X \cup Y & X \cap Y \subseteq X \\ Y \subseteq X \cup Y & X \cap Y \subseteq Y \end{array}$$

Applying to them the completion process of the previous subsection we get the canonical $\langle R_\subseteq, R_\supseteq \rangle$ bi-rewriting modulo I system shown in figure 4.4. This is basically a duplication of the bi-rewriting system of figure 4.3. Notice that rule r_4 for $f = \cap$: $(X_1 \cap X_2) \cup (Y_1 \cap Y_2) \xrightarrow{\subseteq} (X_1 \cup Y_1) \cap (X_2 \cup Y_2)$ is subsumed by r_3 , and r_8 for $f = \cup$ is subsumed by r_7.

We don't know of any canonical rewriting system for non-distributive lattices, although it exists for distributive lattices (Hullot, 1980) and for boolean rings (Hsiang and Dershowitz, 1983). So its modeling by a bi-rewriting system represents a contribution to rewriting techniques. The lack of disjunctive and conjunctive normal forms in non-distributive lattices is the cause of non-existence of a canonical rewriting system for them. On the contrary, the proposed bi-rewriting system has two normalizing rules. Rules r_3 and r_7 acting in opposite directions allow to get a disjunctive normal form the first, and the

$$R_{\subseteq} = \left\{ \begin{array}{ll}
r_1 & X \cup X \xrightarrow{\subseteq} X \\
r_1^{ext} & X \cup X \cup Y \xrightarrow{\subseteq} X \cup Y \\
r_2 & X \cap Y \xrightarrow{\subseteq} X \\
r_3 & X \cup (Y \cap Z) \xrightarrow{\subseteq} (X \cup Y) \cap (X \cup Z) \\
r_3^{ext} & X \cup (Y \cap Z) \cup T \xrightarrow{\subseteq} \Big((X \cup Y) \cap (X \cup Z)\Big) \cup T \\
\multicolumn{2}{l}{\forall f \in \mathcal{F}^n} \\
r_4 & f(X_1 \ldots X_n) \cup f(Y_1 \ldots Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1 \ldots X_n \cup Y_n) \\
r_4^{ext} & f(X_1 \ldots X_n) \cup f(Y_1 \ldots Y_n) \cup Z \xrightarrow{\subseteq} \\
 & \qquad\qquad \xrightarrow{\subseteq} f(X_1 \cup Y_1 \ldots X_n \cup Y_n) \cup Z
\end{array} \right.$$

$$R_{\supseteq} = \left\{ \begin{array}{ll}
r_5 & X \cap X \xrightarrow{\supseteq} X \\
r_5^{ext} & X \cap X \cap Y \xrightarrow{\supseteq} X \cap Y \\
r_6 & X \cup Y \xrightarrow{\supseteq} X \\
r_7 & X \cap (Y \cup Z) \xrightarrow{\supseteq} (X \cap Y) \cup (X \cap Z) \\
r_7^{ext} & X \cap (Y \cup Z) \cap T \xrightarrow{\supseteq} \Big((X \cap Y) \cup (X \cap Z)\Big) \cap T \\
\multicolumn{2}{l}{\forall f \in \mathcal{F}^n} \\
r_8 & f(X_1 \ldots X_n) \cap f(Y_1 \ldots Y_n) \xrightarrow{\supseteq} f(X_1 \cap Y_1 \ldots X_n \cap Y_n) \\
r_8^{ext} & f(X_1 \ldots X_n) \cap f(Y_1 \ldots Y_n) \cap Z \xrightarrow{\supseteq} \\
 & \qquad\qquad \xrightarrow{\supseteq} f(X_1 \cap Y_1 \ldots X_n \cap Y_n) \cap Z
\end{array} \right.$$

$$I = \left\{ \begin{array}{ll}
r_9 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y \\
r_{11} & Y \cap X \xleftrightarrow{\subseteq} X \cap Y \\
r_{10} & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) \\
r_{12} & (X \cap Y) \cap Z \xleftrightarrow{\subseteq} X \cap (Y \cap Z)
\end{array} \right.$$

Figure 4.4: A canonical bi-rewriting system for the inclusion theory of non-distributive lattices.

other a conjunctive normal form. In a non-distributive lattice these rules are strict inclusions therefore they can not be used as equational rewrite rules. Furthermore, if they are put together in a unique rewriting system then we lose the termination property.

4.4.3 Inclusion Theory of Distributive Lattices

The example we present now is the inequality specification of distributive lattices. This specification is the base for many other specifications or specification languages like the *Unified Algebras* (Mosses, 1989b; Mosses, 1989c; Mosses, 1989a). The presentation of the distributive lattice theory may be given by the following set of inclusions:

$$\begin{array}{ll} X \cup X \subseteq X & X \cap X \supseteq X \\ X \cup Y \supseteq X & X \cap Y \subseteq X \\ X \cup Y \supseteq Y & X \cap Y \subseteq Y \\ \multicolumn{2}{c}{X \cap (Y \cup Z) \subseteq (X \cap Y) \cup (X \cap Z)} \end{array}$$

As we have seen in the previous examples, the orientation of all these inclusions to the right results in a terminating bi-rewriting system where all ***standard*** critical pairs are confluent. However, the presence of the two non-left-linear rules $X \cup X \xrightarrow{\subseteq} X$ and $X \cap X \xrightarrow{\supseteq} X$ makes necessary the consideration of the *extended* critical pairs. If we only take into account, in a first step, all those extended critical pairs of the form $\langle \sigma(\alpha_1[\beta_2]_p), \sigma(\beta_1) \rangle$, which correspond to the particular case where the position q in F is the most external one $q = \lambda$, then we can generate the following sequence of new rules:

$$\begin{array}{ll} q_1 & Y \cup (X \cup Y) \xrightarrow{\subseteq} X \cup Y \\ q_2 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y \\ & \\ q_3 & (X \cup Y) \cup Y \xrightarrow{\subseteq} X \cup Y \\ q_4 & (X \cup Y) \cup (Y \cup Z) \xrightarrow{\subseteq} X \cup (Y \cup Z) \\ q_5 & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) \end{array}$$

and the equivalent ones for $\cap$. The rules q_2 and q_5 are non-orientable and subsume the rest of rules. They make necessary the use of the bi-rewriting modulo a set of inclusions technique. These rules are symmetric –they are really equations–, therefore we can apply the standard commutative-associative closure definition (Peterson and Stickel, 1981). We obtain then the following set of rules.

$$R_{\subseteq} = \left\{ \begin{array}{ll} r_1 & X \cup X \xrightarrow{\subseteq} X \\ r_1^{ext} & X \cup X \cup Y \xrightarrow{\subseteq} X \cup Y \\ r_2 & X \cap Y \xrightarrow{\subseteq} X \\ r_3 & X \cap (Y \cup Z) \xrightarrow{\subseteq} (X \cap Y) \cup (X \cap Z) \\ r_3^{ext} & X \cap (Y \cup Z) \cap T \xrightarrow{\subseteq} \big((X \cap Y) \cup (X \cap Z)\big) \cap T \end{array} \right.$$

$$R_{\supseteq} = \left\{ \begin{array}{ll} r_4 & X \cap X \xrightarrow{\supseteq} X \\ r_4^{ext} & X \cap X \cap Y \xrightarrow{\supseteq} X \cap Y \\ r_5 & X \cup Y \xrightarrow{\supseteq} X \end{array} \right.$$

$$I = \left\{ \begin{array}{ll} r_6 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y \\ r_7 & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) \\ r_8 & Y \cap X \xleftrightarrow{\subseteq} X \cap Y \\ r_9 & (X \cap Y) \cap Z \xleftrightarrow{\subseteq} X \cap (Y \cap Z) \end{array} \right.$$

In a second step we have to consider also those rules needed to make confluent the rest of extended critical pairs. They are the following ones:

$$\begin{array}{c} F[X] \cup F[X \cup Y] \subseteq F[X \cup Y] \\ F[X \cap Y] \subseteq F[X] \cap F[X \cap Y] \end{array}$$

First, we will study the second extended critical pair. If we orient it to the left, we obtain the rule scheme $F[X] \cap F[X \cap Y] \xrightarrow{\supseteq} F[X \cap Y]$. This rule scheme generates a standard critical pair with the rule $X \cap Y \xrightarrow{\subseteq} Y$, which is made confluent adding the rule scheme $F[X] \cap F[Y] \xrightarrow{\supseteq} F[X \cap Y]$. The overlapping of the context $F[\_]$ of this rule scheme with the left part of the rule $X \cap Y \xrightarrow{\subseteq} Y$ generates infinitely many rule schemes $F[X_1, \ldots, X_n] \cap F[Y_1, \ldots, Y_n] \xrightarrow{\supseteq} F[X_1 \cap Y_1, \ldots, X_n \cap Y_n]$ for $n \geq 1$. The following (normal) rules subsume these rule schemes.

$$\begin{array}{ll} r_{10} & X \cap (Y \cup Z) \xrightarrow{\supseteq} (X \cap Y) \cup (X \cap Z) \\ \forall f \in \Sigma^n . & \\ r_{11}^{(f)} & f(X_1, \ldots, X_n) \cap f(Y_1, \ldots, Y_n) \xrightarrow{\supseteq} f(X_1 \cap Y_1, \ldots, X_n \cap Y_n) \end{array}$$

The rule r_{10} subsumes the instantiation of $r_{11}^{(f)}$ for the symbol $\cup \in \Sigma^2$. The dual solution is not applicable to $F[X] \cup F[X \cup Y] \subseteq F[X \cup Y]$ because $X \cup (Y \cap Z) \xrightarrow{\subseteq} (X \cup Y) \cap (X \cup Z)$ and the distributive rule r_3 would lead to the non-termination of the system. This problem can be avoided using the alternative set of rules:

$$\begin{array}{ll} r_{12}^{(f)} & f(X_1, \ldots, X_n) \cup f(Y_1, \ldots, Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1, \ldots, X_n \cup Y_n) \\ r_{13}^{(f)} & (X \cap f(Y_1, \ldots, Y_n)) \cup (X \cap f(Z_1, \ldots, Z_n)) \xrightarrow{\subseteq} \\ & \qquad \xrightarrow{\subseteq} X \cap f(Y_1 \cup Z_1, \ldots, Y_n \cup Z_n) \end{array}$$

They do not subsume $F[X] \cup F[Y] \xrightarrow{\subseteq} F[X \cup Y]$, but are particular instances of this rule schema. The last rule r_{13} is non-left-linear and generates a new extended critical pair which becomes confluent if we add the following rule.

$$\begin{array}{ll} r_{14}^{(f)} & (X \cap f(Y_1, \ldots, Y_n)) \cup (Z \cap f(V_1, \ldots, V_n)) \xrightarrow{\subseteq} \\ & \qquad \xrightarrow{\subseteq} (X \cup Z) \cap f(Y_1 \cup V_1, \ldots, Y_n \cup V_n) \end{array}$$

Rules $r_{14}^{(f)}$ and r_1 subsume $r_{13}^{(f)}$.

Let us prove now that rules r_{12} and r_{14} makes confluent the extended critical pair $F[X] \cup F[X \cup Y] \subseteq F[X \cup Y]$. Rules $r_{12}^{(f)}$ and $r_{14}^{(f)}$ subsume $F[X] \cup F[Y] \xrightarrow{\subseteq} F[X \cup Y]$ when the schema[17] $F[\cdot]$ can be expressed as a composition $F[\cdot] = F_1[\ldots F_n[\cdot] \ldots]$ of schemas, where each one of this schemes satisfies

[17]As usual, an schema is an expression with a *hole* in it, a selected position, denoted by an dot "$\cdot$". The schema composition $F[\cdot] \circ G[\cdot]$ is defined by the substitution of this selected position by the other schema, noted $F[G[\cdot]]$.

$F_i[\cdot] = f(\ldots,\cdot,\ldots)$, or $F_i[\cdot] = E_1 \cap f(\ldots,\cdot,\ldots) \cap E_2$ for any symbol f different from $\cap$, and any expressions E_1, E_2. It can be proved that any scheme $F[\cdot]$ can be expressed as $F[\cdot] = G[E_1 \cap \cdot \cap E_2]$ where the schema $G[\cdot]$ satisfies the previous condition and E_1, E_2 are two common expressions. This property allows to translate the inclusion schema (the extended critical pair) $F[X] \cup F[X \cup Y] \subseteq F[X \cup Y]$ into

$$G[X \cap H] \cup G[(X \cup Y) \cap H] \subseteq G[(X \cup Y) \cap H]$$

where $G[\cdot]$ can be rewritten using $F[X] \cup F[Y] \xrightarrow{\subseteq} F[X \cup Y]$. We prove then that this extended critical pair is bi-confluent using the following proof.

$$\begin{array}{rcl} G[X \cap H] \cup G[(X \cup Y) \cap H] & \xrightarrow{\subseteq} & G\big[(X \cap H) \cup \big((X \cup Y) \cap H\big)\big] \\ & \xrightarrow{\subseteq} & G[(X \cap H) \cup (X \cap H) \cup (Y \cap H)] \\ & \xrightarrow{\subseteq} & G[(X \cap H) \cup (Y \cap H)] \\ & \xleftarrow{\subseteq} & G[(X \cup Y) \cap H] \end{array}$$

A commutative and terminating bi-rewriting system for the distributive lattice theory is given by rules $r_1 \ldots r_{12}$, r_{14} and their corresponding $\cup$ and $\cap$ associative-commutative extensions, as shown in figure 4.5.

4.5 Why Inclusions and not Equations

In section 4.4 we have seen the possibility of modeling the deduction in a non-distributive free lattice by a canonical bi-rewriting system. This represents an advantage of the inclusion theory over the equational theory of lattices because there is not a canonical rewrite system for the equational theory of lattices (Freese et al., 1993). In general inclusions express weaker constraints between terms than equations. Even in the case of lattices where inclusions may be modeled by equations —the inclusion $a \subseteq b$ is modeled by $a \cup b = b$ or by $a \cap b = a$— inclusions are more natural and have some advantages. The transitivity and monotonicity of inclusions which are captured implicitly by bi-rewriting systems, must be "implemented" explicitly by equational rewrite rules. Let's consider an example. The inclusions $a \subseteq b$ and $b \subseteq c$ can be oriented like $a \xrightarrow{\subseteq} b$ and $b \xrightarrow{\subseteq} c$ and we can prove $a \subseteq c$ rewriting a into b and b into c. However, their translation into equations results in two rules $a \cup b \longrightarrow b$ and $b \cup c \longrightarrow c$. These rules generate non-confluent critical pairs with the other rules $X \cap (X \cup Y) \longrightarrow X$ and $X \cup (X \cap Y) \longrightarrow X$ defining the union and intersection, and the completion process leads to add the following rules $a \cap b \longrightarrow a$ and $b \cap c \longrightarrow b$. And, what is worse, it introduces the rules $a \cup c \longrightarrow c$ and $a \cap c \longrightarrow a$. It means that in general the completion of a theory where the sequence $a_1 \subseteq \ldots \subseteq a_n$ can be proved leads to add rules $a_i \cup a_j \longrightarrow a_j$ and $a_i \cap a_j \longrightarrow a_i$ for any $i < j$, during the completion process.

The transitivity of inclusions is not captured by the transitivity of the equality relation or by the transitivity of the rewriting relation $\xrightarrow{*}$, weakening in this way the power of rewriting systems, and loosing in most cases the possibility of having a canonical rewriting system for a theory.

$$
R_{\subseteq} = \left\{
\begin{array}{ll}
r_1 & X \cup X \xrightarrow{\subseteq} X \\
r_1^{ext} & X \cup X \cup Y \xrightarrow{\subseteq} X \cup Y \\
r_2 & X \cap Y \xrightarrow{\subseteq} X \\
r_3 & X \cap (Y \cup Z) \xrightarrow{\subseteq} (X \cap Y) \cup (X \cap Z) \\
r_3^{ext} & X \cap (Y \cup Z) \cap T \xrightarrow{\supseteq} \Big((X \cap Y) \cup (X \cap Z)\Big) \cap T \\
\multicolumn{2}{l}{\forall f \in \mathcal{F}^n} \\
r_{12} & f(X_1, \ldots, X_n) \cup f(Y_1, \ldots, Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1, \ldots, X_n \cup Y_n) \\
r_{12}^{ext} & f(X_1, \ldots, X_n) \cup f(Y_1, \ldots, Y_n) \cup Z \xrightarrow{\subseteq} \\
 & \qquad\qquad \xrightarrow{\subseteq} f(X_1 \cup Y_1, \ldots, X_n \cup Y_n) \cup Z \\
r_{14} & (X \cap f(Y_1, \ldots, Y_n)) \cup (Z \cap f(V_1, \ldots, V_n)) \xrightarrow{\subseteq} \\
 & \qquad\qquad \xrightarrow{\subseteq} (X \cup Z) \cap f(Y_1 \cup V_1, \ldots, Y_n \cup V_n) \\
r_{14}^{ext} & (X \cap f(Y_1, \ldots, Y_n)) \cup (Z \cap f(V_1, \ldots, V_n)) \cup W \xrightarrow{\subseteq} \\
 & \qquad\qquad \xrightarrow{\subseteq} ((X \cup Z) \cap f(Y_1 \cup V_1, \ldots, Y_n \cup V_n)) \cup W
\end{array}
\right.
$$

$$
R_{\supseteq} = \left\{
\begin{array}{ll}
r_4 & X \cap X \xrightarrow{\supseteq} X \\
r_4^{ext} & X \cap X \cap Y \xrightarrow{\supseteq} X \cap Y \\
r_5 & X \cup Y \xrightarrow{\supseteq} X \\
r_{10} & X \cap (Y \cup Z) \xrightarrow{\supseteq} (X \cap Y) \cup (X \cap Z) \\
r_{10}^{ext} & X \cap (Y \cup Z) \cap T \xrightarrow{\supseteq} \Big((X \cap Y) \cup (X \cap Z)\Big) \cap T \\
\multicolumn{2}{l}{\forall f \in \mathcal{F}^n} \\
r_{11} & f(X_1 \ldots X_n) \cap f(Y_1 \ldots Y_n) \xrightarrow{\supseteq} f(X_1 \cap Y_1 \ldots X_n \cap Y_n) \\
r_{11}^{ext} & f(X_1 \ldots X_n) \cap f(Y_1 \ldots Y_n) \cap Z \xrightarrow{\supseteq} \\
 & \qquad\qquad \xrightarrow{\supseteq} f(X_1 \cap Y_1 \ldots X_n \cap Y_n) \cap Z
\end{array}
\right.
$$

$$
I = \left\{
\begin{array}{ll}
r_6 & Y \cup X \xleftrightarrow{\subseteq} X \cup Y \\
r_7 & Y \cap X \xleftrightarrow{\subseteq} X \cap Y \\
r_8 & (X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z) \\
r_9 & (X \cap Y) \cap Z \xleftrightarrow{\subseteq} X \cap (Y \cap Z)
\end{array}
\right.
$$

Figure 4.5: A canonical bi-rewriting system for the inclusion theory of distributive lattices.

Moreover, the stability (closure for congruence) of the rewriting relation captures the congruence property for $=$, but not the monotonicity property for $\subseteq$. This would make necessary to consider the inclusion $f(X) \subseteq f(X \cup Y)$ and the corresponding rule $f(X) \cup f(X \cup Y) \longrightarrow f(X \cup Y)$ for each symbol f in the signature if we use the implementation described below.

4.6 Related Work

In the context of automated theorem proving, resolution is not very effective in dealing with transitive relations. Special techniques have been devised for such relations, specially for equivalence relations which have attracted most of the attention. Slagle (Slagle, 1972) was the first to encode resolution with the transitivity axiom in a chaining system with paramodulation (Robinson and Wos, 1969) for theories with equality, orders and sets. Chaining into variables, which is needed for completeness, is too prolific, like our extended critical pairs or like variable instance pairs in (Bachmair et al., 1986b). For special order theories this problem can be avoided. For dense total orderings without endpoints, Bledsoe and Hines (Bledsoe and Hines, 1980) proposed techniques for eliminating certain occurrences of variables from formulas. Bledsoe, Kunen and Shostak (Bledsoe et al., 1985) and Hines (Hines, 1992) gave completeness results for these restricted chaining systems. Monotonicity or anti-monotonicity of functions with respect to special (transitive) relations led Manna and Waldinger (Manna and Waldinger, 1986) to propose subterm chaining methods for general clauses but the proposed calculus was shown to be incomplete (Manna and Waldinger, 1992). In (Levy and Agustí, 1993c) we were the first to apply rewrite techniques to non-symmetric and monotonic relations by means of bi-rewriting systems. Bachmair and Ganzinger (Bachmair and Ganzinger, 1993c) used the idea of bi-rewriting to give a refutationally complete inference system of ordered chaining for general clauses and general transitive relations. They studied the particular case of dense total orderings using this technique in (Bachmair and Ganzinger, 1993a).

4.7 Conclusions

We have shown the adequacy of using a pair of rewriting systems and a bi-directional search procedure to automate the deduction with monotonic inclusions. Like in the equational case, a soundness and completeness theorem can be stated. However, in this case, they are based on an *extended* definition of critical pair which include *schemes* of critical pairs. It means that, if we want to use a kind of Knuth-Bendix completion algorithm, then we have to face the problem of working with schemes of rules. In chapter 5 we undertake this problem by means of second-order rules.

Chapter 5

Second-Order Bi-Rewriting Systems

Abstract: In the previous chapter we proved a critical pairs lemma, based on an extended definition of critical pair. This lemma is used to prove the completeness of bi-rewriting systems as deduction methods. However, the orientation of divergent extended critical pairs may give rise to *rule schemes* which disallow to automate the Knuth-Bendix completion process. In this chapter we propose the use of the *linear second-order λ-calculus* to *codify* these schemes. We provide a unification algorithm for such language and we prove a new critical pairs lemma for second-order bi-rewriting systems. Like in the previous chapter, the completion process is described by means of an example. Linear second-order λ-calculus can also be seen as another approach to the definition of Higher-Order Rewriting Systems besides the one based on *patterns* (Nipkow, 1991).

5.1 Introduction

Term Rewriting Systems (Dershowitz and Jouannaud, 1990) have been usually associated with the implementation of equational theories. Term Bi-rewriting Systems introduced in the previous chapter play the same role for inclusion theories. The orientation of a set of inclusions I (axioms with the form $a \subseteq b$) may result then in two sets of rewriting rules $R_{\subseteq}$ and $R_{\supseteq}$ and, therefore, two rewriting relations $\xrightarrow[R_{\subseteq}]{}$ and $\xrightarrow[R_{\supseteq}]{}$. A *bi-rewriting system* $\langle R_{\subseteq}, R_{\supseteq} \rangle$ is said to be 1) *quasi-terminating* (or globally terminating) if the sets $\{u \mid t \xrightarrow[R_{\subseteq}]{*} u\}$ and $\{u \mid t \xrightarrow[R_{\supseteq}]{*} u\}$ are both finite for any term t, 2) *Church-Rosser* if the property $(\xrightarrow[R_{\subseteq}]{} \cup \xleftarrow[R_{\supseteq}]{})^* \subseteq \xrightarrow[R_{\subseteq}]{*} \circ \xleftarrow[R_{\supseteq}]{*}$ holds and 3) *canonical* if both conditions are satisfied. As we have shown, these conditions are sufficient to prove the existence of a terminating and complete procedure for deriving inclusions $a \subseteq b$ based on the bi-directional search of a common reduct of a and b (an expression c such that $a \xrightarrow[R_\subseteq]{*} c$ and $b \xrightarrow[R_\supseteq]{*} c$).

If a rewriting relation is *finitely branching*,[1] as it is the case of first-order rewriting when any rewriting rule $l \longrightarrow r$ satisfies $\mathcal{V}(r) \subseteq \mathcal{V}(l)$, and terminating, then it is also quasi-terminating. This result is used to prove the first condition.

The second condition is proved by means of a *critical pairs lemma* (theorems 4.10 and 4.19). However, the bi-rewriting version of this lemma is based on an *extended* definition of critical pair (definition 4.9). This set of critical pairs is in general infinite (we are completely free to choose the context $F[\cdot]_q$ appearing in the definition of extended critical pairs). Although there exists canonical bi-rewriting systems for many inclusion theories (see the examples in section 4.4), the standard Knuth-Bendix completion procedure is of little practical help to automatically complete a bi-rewriting system. In this chapter we present an approach to this problem by means of second-order bi-rewriting systems.

In section 5.2 we show how these infinitely many extended critical pairs can be made confluent introducing rule schemes. These rule schemes can be *implemented* using second-order rules. However, the use of the *full* simple typed second-order λ-calculus for rewriting purposes introduces some problems, stated in section 5.3. Because of that, we define a restricted second-order language called *linear second-order λ-calculus*, which is described in section 5.4. Section 5.5 defines an unification procedure for this language. The new critical pairs lemma for second-order bi-rewriting systems is proved in section 5.6. Finally, we illustrate how the Knuth-Bendix completion procedure could be implemented throughout an example in section 5.7.

5.2 Codifying Rule Schemes by means of Second-Order Rules

From now on we will be concerned with the simply typed second-order λ-calculus. Thus, we will deal with a set of types $\mathcal{T} = \bigcup_{n\in\mathbb{N}} \mathcal{T}^n$ built up over a set $\mathcal{T}^1$ of base (first-order) types; where, as usual, $\mathcal{T}^n$ is the set of n-ordered types defined inductively as the minimum set containing $\mathcal{T}^{n-1}$ and such that if $\tau \in \mathcal{T}^{n-1}$ and $\tau' \in \mathcal{T}^n$ then $\tau \rightarrow \tau' \in \mathcal{T}^n$. Terms of the simply typed second-order λ-calculus $\mathcal{T}(\mathcal{F},\mathcal{X})$ are defined over a signature of third-order typed constants $\mathcal{F} = \bigcup_{\tau\in\mathcal{T}^2} \mathcal{F}_\tau$ and second-order typed variables $\mathcal{X} = \bigcup_{\tau\in\mathcal{T}^1} \mathcal{X}_\tau$. The typing relation $t : \tau$ is defined by the following set of inference rules

$$\frac{\{c \in \mathcal{F}_\tau\}}{c : \tau} \qquad \frac{\{x \in \mathcal{X}_\tau\}}{x : \tau} \qquad \frac{x : \tau \qquad t : \tau'}{(\lambda x : \tau \,.\, t) : \tau \rightarrow \tau'} \qquad \frac{t : \tau \rightarrow \tau' \qquad t' : \tau}{t(t') : \tau'}$$

The term t is said to be a well-formed n-order typed term, noted $t \in \mathcal{T}^n(\mathcal{F},\mathcal{X})$, if $t : \tau$ can be inferred from the set of rules below and $\tau \in \mathcal{T}^n$. The set of *free variables* of a term (noted $\mathcal{FV}(t)$), replacement (noted $t[X \mapsto u]$), and other concepts commonly used in λ-calculus are defined as usual (Barendregt, 1981; Hindley and Seldin, 1986). We will note free variables with capital letters (by

[1] A relation $\longrightarrow$ is finitely branching if the set $\{u \mid t \longrightarrow u\}$ is finite for any term t.

$X, Y, Z, \ldots$ when they are first-order typed and by $F, G, H, I, \ldots$ when they are second-order typed), bound variables and constants are noted using lower case letters.

Definition 5.1 *A (second-order typed)* **substitution** $\sigma = [X_1 \mapsto t_1, \cdots, X_n \mapsto t_n]$ *is a mapping from a finite set of variables* $\mathcal{Dom}(\sigma) = \{X_1, \ldots, X_n\} \subseteq \mathcal{X}$ *to* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ *such that* X_i *and* t_i *have the same type. This mapping is extended as a type-preserving mapping* $\sigma : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ *defined by*[2]

$$\sigma(u) \stackrel{def}{=} (\lambda X_1 \ldots X_n \,.\, u)(t_1, \ldots, t_n) = \big(u[X_1 \mapsto t_1] \ldots\big)[X_n \mapsto t_n]$$

The set of **free variables of a substitution** σ *is defined as follows.*

$$\mathcal{FV}(\sigma) = \bigcup_{X \in \mathcal{Dom}(\sigma)} \mathcal{FV}(\sigma(X))$$

Composition of two substitutions σ *and* τ *is a substitution, noted* $\tau\circ\sigma$, *such that* $\mathcal{Dom}(\tau\circ\sigma) \stackrel{def}{=} \mathcal{Dom}(\sigma)$ *and* $\tau\circ\sigma(X) \stackrel{def}{=} \tau(\sigma(X))$ *for any* $X \in \mathcal{Dom}(\sigma)$.[3]
A **partial order between substitutions** *can be defined as usual, i.e. we say that* $\rho \prec \sigma$ *if there exists a substitution* π *such that* $\sigma = \pi\circ\rho$.

The inclusion theory of the union operator is used throughout to motivate the definition of second-order bi-rewriting systems. In subsection 4.4.1 we proved the existence of a canonical first-order bi-rewriting system for such a theory (shown in figure 4.2). The same example is completed in section 5.7 for the second-order case. Our intention is to replace the set of rules of such example

$$f(X_1, \ldots, X_n) \cup f(Y_1, \ldots, Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1, \ldots, X_n \cup Y_n)$$

by a second-order rule. If we take up again the completion process described in subsection 4.4.1, we have that this set of rules is generated by the rule schema $F[X] \cup F[Y] \xrightarrow{\subseteq} F[X \cup Y]$, which results of making bi-confluent an extended critical pair, and where $F[\cdot]$ denotes a context. We will see now that we can *translate* this rule scheme into the second-order rule $G(X) \cup G(Y) \xrightarrow{\subseteq} G(X \cup Y)$, where now G denotes a second-order typed variable. Then, it is easy to see that this second-order rule subsumes the previous rule schema because the function variable G can be instantiated by $\lambda x \,.\, F[x]$. However, it does not subsume other rules like $f(X_1, \ldots, X_n) \cup f(Y_1, \ldots, Y_n) \xrightarrow{\subseteq} f(X_1 \cup Y_1, \ldots, X_n \cup Y_n)$ for $n \geq 2$. To obtain second-order rules subsuming them we must complete the system by generating all the critical pairs between $G(X) \cup G(Y) \xrightarrow{\subseteq} G(X \cup Y)$ and other rules.

The simply typed second-order λ-calculus is enough to *model* an untyped first-order language with contexts variables, like the one described by Comon in (Comon, 1993). In such a model, we can suppose that there exists an unique

[2]Notice that $\big(u[X_1 \mapsto t_1]\big)[X_2 \mapsto t_2] = u[X_1 \mapsto t_1, X_2 \mapsto t_2]$, but in general, $u[X_1 \mapsto t_1, X_2 \mapsto t_2] \neq u[X_2 \mapsto t_2, X_1 \mapsto t_1]$.

[3]Notice that $\tau\circ\sigma(t) \neq \tau(\sigma(t))$ unless we have $\mathcal{Dom}(\tau) \subseteq \mathcal{FV}(\sigma)$.

first-order type $Term \in \mathcal{T}^1$. Any n-ary symbol f of the signature is interpreted as a unary second-order typed constant $f : Term \overset{n}{\to\cdots\to} Term \to Term$, any variable X as a first-order typed variable $X : Term$ and any context variable $F[\cdot]$ as a second-order typed variable $F : Term \to Term$.

5.3 Some Problems of Second-Order Rewriting Systems

The use of *full* simple typed second-order typed λ-calculus in rewriting systems is not free from problems. If we unify a term (pattern) with a ground term (a term without free variables), the resulting unifier(s) do not necessarily instantiate all the free variables of the pattern. For instance, if we unify the pattern $F(X)$ with the ground expression $f(a)$, a minimum unifier ρ may assign $\rho(F) = \lambda x \,.\, f(a)$ and leave X non instantiated. It means that, although all variables appearing in the right part of a rule would also appear in its left part, not all the instantiations of such rule will satisfy that property. Therefore, the use of this rule can introduce new free variables during the rewriting process. For instance, the rule $F(X) \longrightarrow X$ satisfies $\mathcal{FV}(X) = \{X\} \subseteq \{X, F\} = \mathcal{FV}(F(X))$, even so it introduces a fresh variable X when is used to rewrite a into X using the substitution $\rho = [F \mapsto \lambda x.a]$. That problem prevents the orientation of the rules to obtain a terminating rewriting system. In the previous example, we can rewrite $a \longrightarrow a \longrightarrow a \longrightarrow \cdots$ using the rule $F(X) \longrightarrow X$ and the substitution $\rho = [F \mapsto \lambda x \,.\, a][X \mapsto a]$. The first-order matching problem satisfies the following property: given a pair of terms t and u there exists finitely many substitutions ρ such that $\mathcal{Dom}(\rho) \subseteq \mathcal{FV}(t)$ and $\rho(t) = u$. This result does not hold in general for second-order languages. It means that a second-order rewriting relation can be infinitely branching and many properties of term rewriting system do not hold. In particular, a second-order terminating rewriting system is not necessarily quasi-terminating.

In next section we define the *linear second-order typed λ-calculus* which avoids these problems (see lemma 5.4). The same kind of problems are studied by Nipkow (Nipkow, 1991; Nipkow, 1992) to justify his definition of ***higher-order rewriting systems*** based on ***patterns***. A term t in $\beta\eta$-normal form is said to be a ***pattern*** if every occurrence of a free variable F is in a subterm $F(\overline{u_n})$ such that $\overline{u_n}$ is a list of distinct bound variables (Nipkow, 1991; definition 3.1). Our approach can be seen as a new kind of higher-order rewriting systems based on the linear second-order typed λ-calculus.

5.4 Linear Second-Order Typed λ-Calculus

In this section we present the linear second-order typed λ-calculus used to *implement* expression schemes. The main idea is to define a second-order calculus where λ-abstractions always bound one and only one occurrence of a variable.

This language is more expressive than a language based on context variables, as the one described by Comon (Comon, 1993), and can be more easily formalized.

The inference rules for defining well-typed linear second-order (LSO) typed terms $t : \tau$ are the following ones.

$$\frac{x \in \mathcal{X}_\tau}{x : \tau} \qquad \frac{c \in \mathcal{F}_\tau}{c : \tau} \qquad \frac{\begin{array}{cc} x : \tau_1 & t : \tau_2 \\ \multicolumn{2}{c}{\{x \text{ occurs once in } t\}} \end{array}}{\lambda x \,.\, t : \tau_1 \to \tau_2} \qquad \frac{t : \tau_1 \to \tau_2 \qquad\qquad u : \tau_1}{t(u) : \tau_2}$$

Like in the simply second-order typed λ-calculus, we also consider the β and η equations:

$$\begin{array}{l} (\lambda x \,.\, t)(u) =_\beta t[x \mapsto u] \\ \lambda x \,.\, t(x) =_\eta t \end{array}$$

Notice that the side condition $x \notin \mathcal{FV}(t)$ is not necessary in the η-rule because, if $\lambda x \,.\, t(x)$ is well-typed, then this condition is ensured. Notice also that these rules transform linear terms into linear terms with the same type. These equations, used as rewriting rules:

$$\begin{array}{ll} (\lambda x \,.\, t)(u) \to_\beta t[x \mapsto u] & \\ t \to_\eta \lambda x \,.\, t(x) & \text{if it does not introduce new } \beta\text{-redex} \end{array}$$

constitute a normalizing rewriting system. The normal form of a term t is denoted by $t|_{\beta\eta}$ and has the form $\lambda x_1 \dots x_n \,.\, a(t_1, \dots, t_m)$ where a can be either a bound variable, a free variable or a constant, $a(t_1, \dots, t_m)$ is a first-order typed term, and $t_1 \dots t_m$ are also normal terms. We require linearity to prove the following lemma.

Lemma 5.2 *For any pair of linear second-order terms t and u, if $t =_{\beta\eta} u$ then $\mathcal{FV}(t) = \mathcal{FV}(u)$.*

Proof: For the η-equation it is trivial because $\mathcal{FV}(\lambda x \,.\, t(x)) = \mathcal{FV}(t) \setminus \{x\}$, but since $x \notin \mathcal{FV}(t)$ we have $\mathcal{FV}(\lambda x \,.\, t(x)) = \mathcal{FV}(t)$. For the β-equation it is necessary to take linear terms. Thus, if $(\lambda x \,.\, t)(u)$ is a well-formed linear term then $x \in \mathcal{FV}(t)$. Therefore $\mathcal{FV}(t[x \mapsto u]) = \big(\mathcal{FV}(t) \setminus \{x\}\big) \cup \mathcal{FV}(u) = \mathcal{FV}\big((\lambda x \,.\, t)(u)\big)$. ∎

We consider any kind of second-order unification problem, and we only restrict the set of possible unifiers.

Definition 5.3 *A* **second-order unification** *(SOU)* **problem** *is a finite set of pairs $\{t_1 \stackrel{?}{=} u_1, \dots, t_n \stackrel{?}{=} u_n\}$, where t_i and u_i are second-order typed terms and have the same type, for any $i \in [1..n]$.*
A **linear second order** *(LSO)* **substitution** *σ is a second-order typed substitution such that $\sigma(X)$ is a LSO-term for any $X \in \mathcal{Dom}(\sigma)$.*
A substitution σ is said to be an **LSO-unifier** *of a SOU problem $\{t_1 \stackrel{?}{=} u_1, \dots, t_n \stackrel{?}{=} u_n\}$ if $\sigma(t_i) =_{\beta\eta} \sigma(u_i)$ for any $i \in [1..n]$ and σ is idempotent.*[4]

[4]The relation $=_{\beta\eta}$ is the congruence defined by the β and η rules of the λ-calculus. A sufficient condition for the idempotence of σ is either $X = \sigma(X)$ or $\mathcal{Dom}(\sigma) \cap \mathcal{FV}(\sigma(X)) = \emptyset$. This restriction does not suppose a loss of generality.

Then we can prove the following lemmas.

Lemma 5.4 *The composition of two LSO-substitutions is also a LSO-substitution.*
Given two terms t and u, there are finitely many (not necessarily minimal) LSO-substitutions σ such that $t = \sigma(u)$ and $\mathcal{D}om(\sigma) \subseteq \mathcal{FV}(u)$.

Proof: Like in the non-linear case, the composition of two substitutions is also type preserving and idempotent. Let us prove that linearity is preserved by composition. It is not difficult to see that if t is a LSO-term and σ is a LSO-substitution, then $\sigma(t)$ is also a linear term. Moreover, β-reduction and η-expansion are both linearity preserving, therefore even if we normalize the new term after applying the substitution $\sigma(t)|_{\beta\eta}$ would be linear.

The proof of the second part of the lemma is based on the matching algorithm described at the end of the section 5.5. ■

LSO-unification recovers some properties of first-order unification that we lose when we pass to the second-order. In particular, we recover the finiteness of the (non-minimal) matching problem. This makes LSO-unification specially adequate for defining LSO rewriting systems because we avoid the infinitely branching problem.

5.5 A Second-Order Unification Procedure

Like in the first-order case, to prove the completeness of a second-order bi-rewriting system we have to generate all the possible (extended) critical pairs between rules in $R_\subseteq$ and rules in $R_\supseteq$ and prove their bi-confluence. This process requires the use of a unification procedure for the linear second-order λ-calculus.

The first sound and complete second-order unification procedure was described by Pietrzykowski (Pietrzykowski, 1973), and subsequently a modified version of this algorithm was proposed to solve the unification problem for the simply typed λ-calculus (Jensen and Pietrzykowski, 1976). Based on it, Huet (Huet, 1975) proposed the computation of the so called independent pre-unifiers using a pre-unification procedure. This procedure does not try to solve the flexible-flexible pairs of a unification problem for which there always exist a unifier, thus a pre-unification procedure is enough if we only want to check if a unification problem is satisfiable. Unfortunately, the simply typed λ-calculus unification problem, and even the second-order unification problems are undecidable (Goldfarb, 1981).

Since then many decidable classes of higher-order unification problems have been described. Miller (Miller, 1991a; Miller, 1991b) in the context of logic programming and Nipkow (Nipkow, 1991; Nipkow, 1992) in the context of rewriting systems, propose a restricted higher-order language –which expressions they call *patterns*– preserving the good properties of the first-order logic. If there exists a minimum unifier of two patterns, then it is unique. They also define a unification algorithm (Nipkow, 1991; theorem 3.2) to find *this* most general pattern unifier and prove its termination. However, in our case we need a more expressive language. If we consider the rule $G(X) \cup G(Y) \xrightarrow{\subseteq} G(X \cup Y)$ for example, we realize that neither the left-part nor the right part of the rule is a pattern. In general, if we look at the particular form of extended critical pairs we will see that they always contain a subexpression $F(t)$ where F is a free variable and t is the right hand side of a rewriting rule, so we can not suppose that t is a list of distinct bounded variables, as the definition of patterns requires. Recently Prehofer has proved in his thesis (Prehofer, 1995) decidability results for some unification problems based on Nipkow's patterns. He proves, for instance, that unification of linear second-order systems is decidable (theorem 5.3.1). Unfortunately, *linear* refers here to the system, not to terms: a linear second-order system of equations is of the form $\overline{\lambda \overline{x_k} \,.\, X_n(\overline{t_{n_m}}) \overset{?}{=} \lambda \overline{x_k} \,.\, t_n}$ where $\overline{X_n}$ are distinct and not occurring elsewhere second-order variables and $\overline{\lambda \overline{x_k} \,.\, t_{n_m}}$ and $\overline{\lambda \overline{x_k} \,.\, t_n}$ are patterns. This decidable case neither covers our needs. Comon (Comon, 1993), as we have said, describes a first-order language based on context variables (a second-order language without λ-abstractions and where second-order variables are restricted to be unary). He also proves the decidability of the unification problem for his language and provides a unification algorithm. However, a rather strong condition is imposed: any occurrence of a free variable F is always applied to the same argument t. This restriction is also violated in our case: in the rule $G(X) \cup G(Y) \xrightarrow{\subseteq} G(X \cup Y)$, the second-order variable G is applied to two different terms, X and Y. Finally, Schmidt-Schauβ proves that second-order unification of stratified terms is decidable. Here stratified terms means that the string of second-order variables on the path from the root of the term to every occurrence of a given variable is always the same.

Other cases are currently being proposed, but none of them is adequate for the computation of extended critical pairs. The most specific unification problem subsuming ours is the general second-order case studied by Pietrzykowski. However our particular case turns up to be attracting as long as it enjoys better properties, as we shall see at the end of this section. On the other hand, the linear second-order unification problem generalizes the associative unification (Makanin, 1977) and monadic second-order unification (Farmer, 1988) problems. These unification problems are known to be decidable, although such result is not as easy to prove as one may suppose in a first approach. Thus, as far as we know, the decidability of the linear second-order λ-calculus unification problem is still an open question, and the procedure we give in this section is in general non-terminating.

In the definition of the algorithm we use a compact notation based on sets of indexes and indexed sets of indexes. For any set of indexes $P = \{p_1, \ldots, p_n\}$, the expression $a(\overline{b_P})$ denotes $a(b_{p_1}, \ldots, b_{p_n})$, and for any P-indexed set of indexes $Q_P = \{Q_{p_1}, \ldots, Q_{p_n}\} = \{\{q_1^1, \ldots, q_1^{m_1}\}, \ldots, \{q_n^1, \ldots, q_n^{m_n}\}\}$ the expression $a(\overline{b_P(\overline{c_{Q_P}})})$ denotes $a(b_{p_1}(c_{q_1^1} \ldots c_{q_1^{m_1}}), \ldots, b_{p_n}(c_{q_n^1} \ldots c_{q_n^{m_n}}))$. Notice that capital letters denote set of indexes whereas lower case letters denote concrete indexes.

We use the notation on transformations introduced by Gallier and Snyder (Gallier and Snyder, 1990) for describing unification processes. Any state of the process is represented by a pair $\langle S,\sigma\rangle$ where $S=\{t_1 \stackrel{?}{=} u_1,\dots,t_n \stackrel{?}{=} u_n\}$ is the set of unification problems still to be solved and σ is the substitution leading from the initial problem to the actual one. The algorithm is described by means of transformation rules on states $\langle S,\sigma\rangle \Rightarrow \langle S',\sigma'\rangle$. The initial state is $\langle S_0, Id\rangle$. If it can be transformed into a state where the unification problem is empty $\langle S_0, Id\rangle \Rightarrow^* \langle \emptyset,\sigma\rangle$ then σ is a solution –unifier– of the unification problem S_0.

The normal forms of a unification problem and of a substitution are defined as follows.

$$\begin{array}{lcll}
\{t_1 \stackrel{?}{=} u_1,\dots,t_n \stackrel{?}{=} u_n\}|_{\beta\eta} & \stackrel{def}{=} & \multicolumn{2}{l}{\{t_1|_{\beta\eta} \stackrel{?}{=} u_1|_{\beta\eta},\dots,t_n|_{\beta\eta} \stackrel{?}{=} u_n|_{\beta\eta}\}} \\
[X_1\mapsto t_1,\dots,X_n\mapsto t_n]|_{\beta\eta} & \stackrel{def}{=} & [\ X_1 & \mapsto\ t_1[X_2\mapsto t_2,\dots,X_n\mapsto t_n]|_{\beta\eta}, \\
 & & & \dots \\
 & & X_{n-1} & \mapsto\ t_{n-1}[X_n\mapsto t_n]|_{\beta\eta}, \\
 & & X_n & \mapsto\ t_n|_{\beta\eta}]
\end{array}$$

The projection of a substitution σ over a set of variables V is a substitution σ' such that $\mathcal{Dom}(\sigma')=V$ and for any $x\in V$ we have $\sigma(x)=\sigma'(x)$. We suppose that the initial state $\langle S_0, Id\rangle$ is in normal form, and that after applying any transformation rule the resulting unification problem is normalized and the substitution is also normalized and projected over $\mathcal{FV}(S_0)$. Therefore, we can suppose that any pair $t \stackrel{?}{=} u \in S$ has the form $\lambda\overline{x_N}\,.\,a(\overline{t_P}) \stackrel{?}{=} \lambda\overline{x_N}\,.\,b(\overline{u_Q})$ where a, b may be either a constant, a bound or a free variable; and if $\langle S_0, Id\rangle \Rightarrow^* \langle S,\sigma\rangle$ then $\mathcal{Dom}(\sigma)\subseteq \mathcal{FV}(S_0)$.

Definition 5.5 *The* **unification problem transformation rules** *have the form:*

$$\langle S\cup\{t \stackrel{?}{=} u\},\sigma\rangle \Rightarrow \langle \rho(S\cup R), \rho\circ\sigma\rangle$$

where the transformation $t \stackrel{?}{=} u \Rightarrow R$ and the LSO-substitution ρ are defined by the following cases.

1. Rigid-rigid rule (or Simplification rule). *If a is a constant, or a bound variable then*

$$\begin{array}{l}
\lambda\overline{x_N}\,.\,a(\overline{t_P}) \stackrel{?}{=} \lambda\overline{x_N}\,.\,a(\overline{u_P}) \ \Rightarrow\ \bigcup_{i\in P}\left\{\lambda\overline{x_N}\,.\,t_i \stackrel{?}{=} \lambda\overline{x_N}\,.\,u_i\right\} \\
\rho = Id
\end{array}$$

2. Imitation rule. *If a is a constant and F is a free variable, and $\{R_i\}_{i\in P}$ is a P-indexed family of disjoint lists of indexes satisfying*[5] *$\bigcup_{i\in P} R_i = Q$, then*

$$\begin{array}{l}
\lambda\overline{x_N}\,.\,a(\overline{t_P}) \stackrel{?}{=} \lambda\overline{x_N}\,.\,F(\overline{u_Q}) \ \Rightarrow\ \bigcup_{i\in P}\left\{\lambda\overline{x_N}\,.\,t_i \stackrel{?}{=} \lambda\overline{x_N}\,.\,F'_i(\overline{u_{R_i}})\right\} \\
\rho = \left[F\mapsto \lambda\overline{y_Q}\,.\,a\left(\overline{F'_P(\overline{y_{R_P}})}\right)\right]
\end{array}$$

where $\{F'_j\}_{j\in Q}$ are fresh free variables of the appropriate type that can be inferred from the context.

[5] Union and comparison of lists of indexes is computed without considering their order.

3. Projection rule. *If a is a constant or a bound variable and F is a free variable, and $a(\overline{t_P})$ and u have the same type, then*

$$\begin{array}{ll}\lambda\overline{x_N}\,.\,a(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,F(u) \;\Rightarrow\; \left\{\lambda\overline{x_N}\,.\,a(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,u\right\} \\ \rho = [F \mapsto \lambda y\,.\,y]\end{array}$$

4. Flexible-flexible rule with equal heads (or Simplification rule). *If F is a free variable, then*

$$\begin{array}{ll}\lambda\overline{x_N}\,.\,F(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,F(\overline{u_P}) \;\Rightarrow\; \displaystyle\bigcup_{i\in P}\left\{\lambda\overline{x_N}\,.\,t_i \overset{?}{=} \lambda\overline{x_N}\,.\,u_i\right\} \\ \rho = Id\end{array}$$

5. Flexible-flexible rule with distinct heads (or Distinct-heads rule). *If F and G are free different variables, $P' \subseteq P$ and $Q' \subseteq Q$ are two lists of indexes, and $\{R_j\}_{j\in Q'}$ is a Q'-indexed and $\{S_i\}_{i\in P'}$ a P'-indexed family of disjoint lists of indexes satisfying*

$$\begin{array}{ll}\left(\bigcup_{j\in Q'} R_j\right) \cup P' = P & \left(\bigcup_{j\in Q'} R_j\right) \cap P' = \emptyset \\ \left(\bigcup_{i\in P'} S_i\right) \cup Q' = Q & \left(\bigcup_{i\in P'} S_i\right) \cap Q' = \emptyset\end{array}$$

then

$$\begin{array}{lll}\lambda\overline{x_N}\,.\,F(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,G(\overline{u_Q}) \;\Rightarrow & \displaystyle\bigcup_{j\in Q'} & \left\{\lambda\overline{x_N}\,.\,F'_j(\overline{t_{R_j}}) \overset{?}{=} \lambda\overline{x_N}\,.\,u_j\right\} \cup \\ & \displaystyle\bigcup_{i\in P'} & \left\{\lambda\overline{x_N}\,.\,t_i \overset{?}{=} \lambda\overline{x_N}\,.\,G'_i(\overline{u_{S_i}})\right\} \\ \rho = & \multicolumn{2}{l}{\left[F \mapsto \lambda\overline{y_P}\,.\,H'\left(\overline{F'_{Q'}(\overline{y_{R_{Q'}}})}\,,\,\overline{y_{P'}}\right)\right]} \\ & \multicolumn{2}{l}{\left[G \mapsto \lambda\overline{z_Q}\,.\,H'\left(\overline{z_{Q'}}\,,\,\overline{G'_{P'}(\overline{z_{S_{P'}}})}\right)\right]}\end{array}$$

where H', $\{G'_i\}_{i\in P'}$ and $\{F'_j\}_{j\in Q'}$ are fresh free variables of the appropriate types.

Example. This procedure can be more easily understood if we compare if with the following string unification procedure. Hence, an string is a sequence of constants or variables and $S_1 \cdot S_2$ denotes the concatenation of S_1 and S_2. Concatenation operator satisfies the associative property $S_1 \cdot (S_2 \cdot S_3) = (S_1 \cdot S_2) \cdot S_3$.

The transformations defining the procedure have also the following form

$$\langle\{t \overset{?}{=} u\} \cup S, \sigma\rangle \Rightarrow \langle\rho(S' \cup S), \rho\circ\sigma\rangle$$

where the transformation $\{t \overset{?}{=} u\} \Rightarrow S'$ and the substitution ρ are defined by the same cases.

1. *Rigid-rigid step.* If a is a constant then

$$\begin{array}{l}a \cdot S_1 \overset{?}{=} a \cdot S_2 \Rightarrow \{S_1 \overset{?}{=} S_2\} \\ \rho = Id\end{array}$$

2. *Imitation step.* If a is a constant and F is a variable then

$$\begin{array}{l} F \cdot S_1 \overset{?}{=} a \cdot S_2 \Rightarrow \{H \cdot S_1 \overset{?}{=} S_2\} \\ \rho = [F \mapsto a \cdot H] \end{array}$$

   where H is a fresh free variable.

3. *Projection step.* If a is a constant and F is a variable then

$$\begin{array}{l} F \cdot S_1 \overset{?}{=} a \cdot S_2 \Rightarrow \{S_1 \overset{?}{=} a \cdot S_2\} \\ \rho = [F \mapsto \langle\rangle] \end{array}$$

4. *Flexible-flexible step with equal heads.* If F is a variable then

$$\begin{array}{l} F \cdot S_1 \overset{?}{=} F \cdot S_2 \Rightarrow \{S_1 \overset{?}{=} S_2\} \\ \rho = Id \end{array}$$

5. *Flexible-flexible step with distinct heads.* If F and G are both variables with $F \neq G$ then either

$$\begin{array}{l} F \cdot S_1 \overset{?}{=} G \cdot S_2 \Rightarrow \{W \cdot S_2 \overset{?}{=} S_1\} \\ \rho = [F \mapsto H][G \mapsto H \cdot W] \end{array}$$

   or

$$\begin{array}{l} F \cdot S_1 \overset{?}{=} G \cdot S_2 \Rightarrow \{V \cdot S_1 \overset{?}{=} S_2\} \\ \rho = [F \mapsto H \cdot V][G \mapsto H] \end{array}$$

   where H, W and V are fresh variables.

Notice that in the string case, as well as in the second-order case, we have the following possibilities:

(i) Both expressions have a constant a and b in the head (rigid-rigid case). Then they have to be the same constant for both expressions $a = b$.

(ii) One expression has a constant a and the other a variable F in the head (rigid-flexible case). There are two possibilities: either the variable is instantiated with the empty string $[F \mapsto \langle\rangle]$ (projection step) or it is instantiated with a string with the constant as head $[F \mapsto a \cdot H]$ (imitation step). In the second case a fresh variable H is introduced.

(iii) Both expressions have a variable in the head (flexible-flexible case). There are two possibilities: either both expressions have the same variable in the head or both variables are different.

In the following we will prove the soundness and completeness theorems for our unification procedure.

Theorem 5.6 soundness. *If $\langle S, Id\rangle \Rightarrow^* \langle \emptyset, \sigma\rangle$, then σ is an unifier of the unification problem S.*

Proof: To prove this theorem we will prove previously the following induction lemma.

Lemma 5.7 *For every transformation rule $\langle \{t \overset{?}{=} u\} \cup S, \sigma\rangle \Rightarrow \langle \rho(S' \cup S), \rho\circ\sigma\rangle$,*

(i) *if $\{t \overset{?}{=} u\} \cup S$ is a SOU problem and σ is a LSO substitution, then $\rho(S' \cup S)$ also is a SOU problem and $\rho\circ\sigma$ is a LSO substitution.*

(ii) *if τ is a LSO unifier of $\rho(S' \cup S)$ then $\tau\circ\rho$ is a LSO unifier of $\{t \overset{?}{=} u\} \cup S$.*

Proof: Firstly, we prove that in all the cases ρ is a LSO substitution. It is a straightforward exercise to cheek that ρ always instantiate free second-order variables by linear second-order terms with the same type, and it only introduces second-order typed fresh free variables. In order to ensure that, we add side conditions on the lists of indexes $\overline{R_P}$ of the imitation rule; conditions on the lists of indexes P', Q', $\overline{R_{Q'}}$ and $\overline{S_{P'}}$ of the flexible-flexible rule; and a side condition on the type of $F : \overline{\tau \to \tau}$ of the projection rule.

To prove that $\{t_i \overset{?}{=} u_i\}$ is a SOU problem, we have to make sure that $\overline{t_i}$ and $\overline{u_i}$ are well-typed terms, which only contains second-order free variables, and t_i and u_i have the same type. Given that S is a SOU problem, the union of two SOU problems is also a SOU problem, and the instantiation of a SOU problem, using a LSO substitution, is also a SOU problem, it suffices to prove that S' is a SOU problem. This is trivial for the simplification rules, provided that they do not free bound variables. For the rest of rules it is also easy to prove, if we decompose them as a LSO instantiation step followed by a simplification step. For instance, the imitation rule is decomposed as follows

$$\begin{array}{rl} \lambda\overline{x_N} \,.\, a(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N} \,.\, F(\overline{u_Q}) & \Rightarrow\ \lambda\overline{x_N} \,.\, a(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N} \,.\, a(\overline{F'_P(\overline{u_{R_P}})}) \\ & \Rightarrow\ \bigcup_{i\in P} \left\{ t_i \overset{?}{=} F'_i(\overline{u_{R_i}}) \right\} \end{array}$$

Composition of LSO substitutions also is a LSO substitution, therefore $\rho\circ\sigma$ will be a LSO substitution.

For the second part of the lemma, if τ is a LSO of $\rho(S' \cup S)$, then it is also a LSO unifier of $\rho(S)$, and $\tau\circ\rho$ is a LSO unifier of S. Now, if we take into account that $\rho(S')$ is the simplification of $\rho(\{t \overset{?}{=} u\})$, and simplification rules preserve unifiers, we can conclude that if τ is a LSO unifier of S', then it is also a LSO unifier of $\rho(\{t \overset{?}{=} u\})$. Therefore, $\tau\circ\rho$ is a LSO unifier of $\{t \overset{?}{=} u\}$. ■

Once we have proved the induction lemma, we are in a position to prove that the predicate P defined as follows

$$P(\langle S, \sigma\rangle) \overset{def}{=} \forall\tau \,.\, \tau \text{ is an unifier of } S \Rightarrow \tau\circ\sigma \text{ is an unifier of } S_0$$

holds for any state $\langle S, \sigma\rangle$ derived from the initial state. We prove it by induction on the length of the sequence of transformations leading from $\langle S_0, Id\rangle$ to $\langle S, \sigma\rangle$.

The initial case $P(\langle S_0, Id\rangle)$ is clearly a tautology.

For the induction case we use the induction lemma. Suppose that τ is a unifier of $\rho(S' \cup S)$, then $\tau\circ\rho$ will be a unifier of $\{t \overset{?}{=} u\} \cup S$. Using now the induction hypothesis $P(\langle \{t \overset{?}{=} u\} \cup S, \sigma\rangle$, we have $\tau\circ\rho\circ\sigma$ is an unifier of S_0 and therefore $P(\langle \rho(S' \cup S), \rho\circ\sigma\rangle)$.

We conclude that the property P holds for all accessible states. For the final state this property is written $P(\langle \emptyset, \sigma\rangle) = \forall\tau \,.\, \tau \text{ is an unifier of } \emptyset \Rightarrow$

$\tau{\circ}\sigma$ is an unifier of S_0. In particular given that $\tau = Id$ is an unifier of $\emptyset$, we have σ is an unifier of S_0. The first part of the induction lemma ensures also that σ is a LSO-substitution. ■

Theorem 5.8 completeness. *If σ is a unifier of the unification problem S, then there exists a transformation sequence $\langle S, Id\rangle \Rightarrow^* \langle\emptyset, \sigma\rangle$.*

Proof: The proof of this theorem is organized in two parts. First we prove that, given a unification problem S_0 and one of its minimum linear unifiers σ, we can generate a transformation sequence

$$\langle S_0, Id\rangle \Rightarrow \langle S_1, \sigma_1\rangle \Rightarrow \cdots \Rightarrow \langle S_n, \sigma_n\rangle \Rightarrow \cdots$$

satisfying $\sigma_n \preceq \sigma$ for any $n \geq 0$, and either this sequence is infinite or its last state is $\langle\emptyset, \sigma\rangle$. Second, we prove that such transformation sequence always terminates, if $\sigma_n \preceq \sigma$ for any $n \geq 0$. The following lemma suffices to generate such transformation sequence.

Lemma 5.9 *() If σ is a unifier of S then either $S = \emptyset$ or there exists an unification problem S' and a substitution ρ such that $\sigma = \sigma'{\circ}\rho$, where σ' is an unifier of S', and $\langle S, \tau\rangle \Rightarrow \langle S', \rho{\circ}\tau\rangle$ for any LSO substitution τ.*

Proof: If S is nonempty, let us consider the first unification problem $\lambda\overline{x_N}\,.\,a(\overline{t_P}) \stackrel{?}{=} \lambda\overline{x_N}\,.\,b(\overline{u_Q})$ of S. If the unification problem is satisfiable (there exists an unifier σ) then there are five possibilities. We present an sketch of the proof for each case.

(i) a and b are both constants or bound variables. Then, if the problem is satisfiable, they have to be the same constant or bound variable applied to the same number of parameters. Moreover, these parameters have to be unificable one to one. Therefore, the rigid-rigid rule is applicable.

(ii) a is a bound variable and b is a free variable (or vice versa). The solution σ has to assign an identity function $\lambda x\,.\,x$ to b and it can be proved that the projection rule is applicable. Notice that if we were looking for a non-second order unifier, other possibility would be $F \mapsto \lambda\overline{y_Q}\,.\,y_i(\overline{F'_P(\overline{y_{R_P}})})]$ where $\exists i \in Q\,.\,\lambda\overline{x_N}\,.\,u_i \stackrel{?}{=} \lambda\overline{X_N}\,.\,a$. However, we have to discard such possibility if we consider that F is second-order typed.

(iii) a is a constant and b a free variable (or vice versa). The substitution σ assigns b either a function with a in the head (in this case it can be proved that the imitation rule is applicable) or the identity function (then projection rule would be applicable).

(iv) a and b are both the same free variable. Whatever the value that σ assigns to this variable, the number of parameters have to be the same, and they are unificable one-to-one, therefore the simplification transformation is applicable. If we were considering any kind of second-order unifier, then σ could instantiate a by a term which discards one of its arguments. Then, we would not have to unify one-to-one *all* the parameters of a, and the elimination rule would be necessary for completeness, to eliminate the parameters discarded by $\sigma(a)$.

(v) a and b are two distinct free variables. This is the most complex case. We prove that the flexible-flexible transformation is enough to treat this case, and the iteration and elimination rules of the general second-order unification procedure (Jensen and Pietrzykowski, 1976) are not necessary. We have $\sigma(\lambda\overline{x_N} \,.\, a(\overline{t_P})) = \sigma(\lambda\overline{x_N} \,.\, b(\overline{u_Q}))$, and since we only consider linear substitutions, the instantiations of the parameters $\overline{\sigma(t_P)}$ and $\overline{\sigma(u_Q)}$ will appear once, and only once, in $\sigma(\lambda\overline{x_N} \,.\, a(\overline{t_P}))$. Then we reason about the relative positions of such occurrences. So P' is the list of indexes of the terms $\sigma(t_i)$ which are not below any $\sigma(u_j)$, and R_j for $j \in Q'$ is the list of indexes of such terms $\sigma(t_i)$ which are below $\sigma(u_j)$. (Notice that if $\sigma(t_i)$ is bellow $\sigma(u_j)$, then $\sigma(u_j)$ can not be bellow any other $\sigma(t_k)$, therefore the list of indexes R_j is indexed by Q', and not by Q). We can see that $\sigma(H')$ is the comon part of $\sigma(a)$ and $\sigma(b)$, i.e. the part of $\sigma(\lambda\overline{x_N} \,.\, a(\overline{t_P})) = \sigma(\lambda\overline{x_N} \,.\, b(\overline{u_Q}))$ which does not overlaps with any of the parameters of a nor b. The iteration and elimination rules of the general case are avoided because we do not need to *eliminate* or *duplicate* occurrences of the parameters. ■

Now we prove by induction that the predicate

$$P(\langle S_n, \sigma_n \rangle) \overset{def}{=} \exists \tau_n \,.\, \sigma = \tau_n \circ \sigma_n \;\wedge\; \tau_n \text{ is a unifier of } S_n$$

holds for any state of a sequence $\langle S_0, Id\rangle \Rightarrow^* \langle S_n, \sigma_n\rangle \Rightarrow \cdots$, generated from a SOU problem S_0 with minimum unifier σ.

For the initial state $\langle S_0, Id\rangle$ we have $\tau_0 = \sigma$ and it is trivially true. For the other states, if $P(\langle S_n, \sigma_n\rangle$ holds then, using the previous lemma, we have that either $S_n = \emptyset$ or there exists a unification problem S_{n+1} and a substitution ρ such that $\tau_n = \tau_{n+1} \circ \rho$ where $\sigma_{n+1} = \rho \circ \sigma_n$ and τ_{n+1} is a unifier of S_{n+1}. Therefore $\sigma = \tau_n \circ \sigma_n = \tau_{n+1} \circ \rho \circ \sigma_n = \tau_{n+1} \circ \sigma_{n+1}$. This proves $\sigma_n \preceq \sigma$ for any $n \geq 0$.

Now, to prove the termination of the sequence $\langle S_0, Id\rangle \Rightarrow^* \langle S_n, \sigma_n\rangle$, we have to define a well-founded ordering on the transformation states. However, we know that the transformation relation is in general non terminating. Take as counter-example the imitation step:

$$\langle \{F(a) \overset{?}{=} G(F(a))\}, \sigma\rangle \Rightarrow \langle \{F'(a) \overset{?}{=} G(F'(a))\}, [F \mapsto \lambda x \,.\, G(F'(x))] \circ \sigma\rangle$$

which generates an infinite transformation sequence. It becomes clear that we have to use the fact that our particular transformation sequence $\langle S_0, Id\rangle \Rightarrow^* \langle S_n, \sigma_n\rangle$ satisfies $\sigma_n \preceq \sigma$ for any $n \geq 0$, where σ is a minimum unifier of S, to prove its termination. Therefore the well-founded ordering we define depends on the unifier of S_0 we are considering.

Dealing with first-order terms, we can define the size of a term as the number of applications it contains, and the size of a substitution as the sum of the sizes of $\sigma(X)$ for all variables X of its domain. It is easy to prove that $size(t) \leq size(\rho(t))$ and $size(\sigma) \leq size(\rho \circ \sigma)$, for any term t and substitutions σ and ρ.

Then the sequence $Id \preceq \sigma_1 \preceq \cdots \preceq \sigma_n \preceq \cdots \preceq \sigma$ would be evidently finite because $\sigma_n = \rho \circ \sigma_{n-1}$ and either ρ composed with σ_{n-1} strictly increases its size or $\rho = Id$ and S_n is strictly smaller than S_{n-1}. On the other hand, if $\sigma_n \preceq \sigma$, the size of the σ_n can not exceed the size of σ. However, this reasoning is not valid for second-order terms, because the substitution $\rho = [F \mapsto \lambda x \,.\, x]$ decreases the size of any term t if $F \in \mathcal{FV}(t)$, and of any substitution σ it is composed with, if $F \in \mathcal{FV}(\sigma)$. Therefore, we will have to consider projection rule separately.

The ordering we define is based on a function from substitutions to integers named free arity (which will be decreased by projection rules), and a function to measure the size of terms and substitutions w.r.t. another substitution.

Definition 5.10 *The* **free arity of a substitution** σ *is defined as follows,*

$$arity(\sigma) = \sum_{X \in \mathcal{FV}(\sigma)} arity(X)$$

where as usual the arity n of a variable X is the maximal number of parameters it admits.[6]

The *size* of a term and a substitution is defined like we would do it for first-order substitutions, but we use the unifier of the original unification problem as reference.

Definition 5.11 *The* **size of a term** t *w.r.t. a substitution ρ satisfying $\mathcal{D}om(\rho) \subseteq \mathcal{FV}(t)$ is defined as follows*

$$size(\lambda \overline{x_P} \,.\, a(\overline{t_Q}), \rho) = \sum_{q \in Q} size(t_q, \rho) + \begin{cases} 0 & \text{if } \rho(a) = \lambda x \,.\, x \\ \#Q & \text{otherwise} \end{cases}$$

where $\#Q$ is the cardinality of Q, and for LSO substitutions is defined as follows

$$size(\sigma, \rho) \stackrel{def}{=} \sum_{X \in \mathcal{D}om(\sigma)} size(\sigma(X), \rho)$$

We can prove that any substitution increases the size of a term it is applied to and of a substitution it is composed with. However, we have to consider a change in the substitution which we use as reference.

Lemma 5.12 *For any LSO term t and LSO substitutions σ, ρ and τ we have*

$$\begin{array}{c} size(\, t \,,\, \tau \circ \rho \,) \leq size(\, \rho(t) \,,\, \tau \,) \\ size(\, \sigma \,,\, \tau \circ \rho \,) \leq size(\, \rho \circ \sigma \,,\, \tau \,) \end{array}$$

Proof: For the first part of the lemma suppose that $t = \lambda \overline{x_P} \,.\, a(\overline{t_Q})$. We prove the inequality by structural induction of the term t, therefore we will suppose that $size(t_q, \tau \circ \rho) \leq size(\rho(t_q), \tau)$ for any $q \in Q$. Now, if a is a bound variable

[6]If $X : \tau_1 \to \ldots \to \tau_n \to \tau$ and τ is a first-order type, then $arity(X) = n$.

or a constant then induction hypothesis are enough to prove the result directly. If a is a free variable then

$$\begin{array}{rcl} size(t, \tau{\circ}\rho) & = & \sum_{q\in Q} size(t_q, \tau{\circ}\rho) + \left\{ \begin{array}{ll} 0 & \text{if } \tau{\circ}\rho(a) = \lambda x\,.\,x \\ \#Q & otherwise \end{array} \right. \\ size(\rho(t), \tau) & = & \sum_{q\in Q} size(\rho(t_q), \tau) + size(\rho(a), \tau) \end{array}$$

Induction hypothesis prove that the first summary is smaller or equal than the second summary. Comparing second terms of both expressions we have to consider three cases. If $\tau{\circ}\rho(a) = \lambda x\,.\,x$ then independently from the value of $\rho(a)$ we can prove that $size(\rho(a), \tau) = 0$. If $\tau{\circ}\rho(a)$ is equal to a free variable then $size(\rho(a), \tau) = \#Q$. Finally for any other value of $\tau{\circ}\rho(a)$ there are several possibilities, but in all cases $size(\rho(a), \tau) \geq \#Q$.

Second inequality of the lemma is easy to prove once we have proved the first part. ■

This lemma allows us to define a distance between LSO substitutions.

Definition 5.13 *Given two LSO substitutions α and β satisfying $\alpha \preceq \beta$, the* **distance between substitutions** *α and β is defined as follows*

$$dist(\alpha, \beta) = size(\beta, Id) - size(\alpha, \rho)$$

where ρ a LSO substitution satisfying $\beta = \rho{\circ}\alpha$ and $\mathcal{D}om(\rho) = \mathcal{FV}(\alpha)$.

Corollary 5.14 *() For any three LSO substitutions $\alpha \preceq \beta \preceq \gamma$ we have*

$$dist(\beta, \gamma) \leq dist(\alpha, \gamma)$$

Now we can define the size of state $\langle S_n, \sigma_n \rangle$ as a triplet where first and second component are integers and the third component is a multiset of integers:

$$size(\langle S_n, \sigma_n \rangle, \tau_n) = \left\langle arity(\sigma_n) \ , \ dist(\sigma_n, \sigma) \ , \ \bigcup_{t \overset{?}{=} u \in S_n} \{size(t, \tau_n), size(u, \tau_n)\} \right\rangle$$

where τ_n is a LSO substitution satisfying $\sigma = \tau_n{\circ}\sigma_n$.

This size may be used to compare states of our particular transformation sequence:

$$\langle S_n, \sigma_n \rangle \preceq \langle S_m, \sigma_m \rangle \ \text{ iff } \ size(\langle S_n, \sigma_n \rangle) \leq size(\langle S_m, \sigma_m \rangle)$$

where $\leq$ is lexicographic extension of the usual ordering on integers and the multiset extension of the ordering on integers. As far as the usual ordering on integers is well-founded, as well as any lexicographic or multiset extension of a well-founded ordering, this ordering based on the size of a state will be well-founded.

To conclude the proof we have to prove that any transformation step in our sequence satisfies $size(\langle S_n, \sigma_n \rangle) > size(\langle S_{n+1}, \sigma_{n+1} \rangle)$, where $>$ is the strict ordering resulting from the well-founded ordering we have defined.

The reader may check that any projection step reduces the arity of the substitution, leaving its size and the size of the unification problem unchanged. Any other transformation does not affect to the arity of the substitution. In imitation steps and flexible-flexible steps the result depends on the size of the variable (or variables) being instantiated. If it is zero, then the size of the substitution remains unchanged but the size of the problem decreases. Otherwise, although the size of the problem increases, the size of the substitution also increases (the difference $size(\sigma, Id) - size(\sigma_n, \tau_n)$ decreases). In simplification steps, the substitution does not change (neither its size) but the size of the problem decreases. ■

Notice that this result proves the completeness of the unification procedure, but not its termination and, therefore, not the decidability of the unification problem. The function *size* defined in the proof could be used to prove the termination of the procedure if we would be able to fix an upper bound $size(\sigma) \leq k$ for the size of a minimum unifier of an unification problem.

Compared with the general procedure (Jensen and Pietrzykowski, 1976), we avoid the use of the prolific *elimination* and *iteration* rules. These rules always compromise the termination of Jensen and Pietrzykowski's procedure. On the contrary, our procedure always finishes for the practical cases where we have used it. In particular, if no free variable occurs more than twice in an unification problem (as use to be the case), then the procedure always finishes. This fact is related with the termination of the *naive* string unification procedure when variables occurs at most twice (Schulz, 1991).

Theorem 5.15 termination. *If no free variable occurs more than twice in a linear second-order unification problem, then this problem is decidable.*

Proof: We define the following *size* function, where we suppose that any term is normalized previously to compute its size.

$$\begin{array}{lcl} size(a) & = & 0 \text{ For any constant, free or bound variable} \\ size(\lambda x_1 \dots x_n \,.\, a(t_1, \dots, t_p)) & = & p + \sum_{i=1}^{n} size(t_i) \\ size(\{t_1 \stackrel{?}{=} u_1, \dots, t_n \stackrel{?}{=} u_n\}) & = & \sum_{i=1}^{n} size(u_i) + size(t_i) \end{array}$$

We prove now that if $\langle S, \sigma\rangle \Rightarrow \langle S', \rho\circ\sigma\rangle$ and any free variable appears at most twice in S then $size(S') \leq size(S)$ and any free variable also appears at most twice in S'. There are five cases.

1. Rigid-rigid rule

$$\lambda\overline{x_N} \,.\, a(\overline{t_P}) \stackrel{?}{=} \lambda\overline{x_N} \,.\, a(\overline{u_P}) \;\Rightarrow\; \bigcup_{i\in P} \left\{\lambda\overline{x_N} \,.\, t_i \stackrel{?}{=} \lambda\overline{x_N} \,.\, u_i\right\}$$

The size of the problem decreases in $2 \cdot \#P$, where $\#P$ is the cardinality of the list of indexes P, and no new variable occurrences are introduced.

2. Imitation rule

$$\begin{array}{l}\lambda\overline{x_N}\,.\,a(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,F(\overline{u_Q}) \;\Rightarrow\; \bigcup_{i\in P}\left\{\lambda\overline{x_N}\,.\,t_i \overset{?}{=} \lambda\overline{x_N}\,.\,F'_i(\overline{u_{R_i}})\right\} \\ \rho = \left[F \mapsto \lambda\overline{y_Q}\,.\,a\Big(\overline{F'_P(\overline{y_{R_P}})}\Big)\right]\end{array}$$

The size of the problem decreases in $\#P$ and is increased in $\#P$ for any other occurrence of the variable F. Thus as far as there is at most one more occurrence, the net decrement will be zero or $\#P$. We also introduce, at most, one occurrence of each one of the fresh variables $\{F'_P\}$.

3. Projection rule

$$\begin{array}{l}\lambda\overline{x_N}\,.\,a(\overline{t_p}) \overset{?}{=} \lambda\overline{x_N}\,.\,F(u) \;\Rightarrow\; \left\{\lambda\overline{x_N}\,.\,a(\overline{t_p}) \overset{?}{=} \lambda\overline{x_N}\,.\,u\right\} \\ \rho = [F \mapsto \lambda y\,.\,y]\end{array}$$

The size of the problem decreases one unity if F occurs once or 2 if it occurs twice. No new free-variable occurrences are introduced.

4. Flexible-flexible rule with equal heads

$$\lambda\overline{x_N}\,.\,F(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,F(\overline{u_P}) \;\Rightarrow\; \bigcup_{i\in P}\left\{\lambda\overline{x_N}\,.\,t_i \overset{?}{=} \lambda\overline{x_N}\,.\,u_i\right\}$$

The problem decreases $2\cdot\#P$ and no new variables are introduced.

5. Flexible-flexible rule with different heads

$$\begin{array}{rl}\lambda\overline{x_N}\,.\,F(\overline{t_P}) \overset{?}{=} \lambda\overline{x_N}\,.\,G(\overline{u_Q}) \;\Rightarrow & \bigcup_{j\in Q'}\left\{\lambda\overline{x_N}\,.\,F'_j(\overline{t_{R_j}}) \overset{?}{=} \lambda\overline{x_N}\,.\,u_j\right\} \cup \\ & \bigcup_{i\in P'}\left\{\lambda\overline{x_N}\,.\,t_i \overset{?}{=} \lambda\overline{x_N}\,.\,G'_i(\overline{u_{S_i}})\right\} \\ \rho = \left[F \mapsto \lambda\overline{y_P}\,.\,H'\Big(\overline{F'_{Q'}(\overline{y_{R_{Q'}}})}\,,\,\overline{y_{P'}}\Big)\right]\left[G \mapsto \lambda\overline{z_Q}\,.\,H'\Big(\overline{z_{Q'}}\,,\,\overline{G'_{P'}(\overline{z_{S_{P'}}})}\Big)\right] & \end{array}$$

The size of the problem decreases in $\#P+\#Q-\sum_{i\in P'}\#S_i-\sum_{j\in Q'}\#R_j = \#P'+\#Q'$ and is increased in $\#Q'$ for any instantiation of F and in $\#P'$ for any instantiation of G. Therefore, in the worst case, the size of the problem remains equal. It is also easy to see that in the worst case we introduce two occurrences of each one of the fresh variables H, $\{F'_i\}$ and $\{G'_j\}$.

■

Although the condition of this theorem may seem very restrictive, it is not so. Given an inclusion, or a critical pair, where a variable occurs more than twice in one of its sides, we can find a set of refutationally equivalent inclusions such that no variable occurs more than twice. Let us see an example.

$$a(F(X),F(Y),F(Z),F(T)) \subseteq b(X,Y,Z,T,\lambda x\,.\,F(x))\ ,\ \Delta\ \vdash \Box$$
$$\text{iff}$$
$$\begin{array}{l} a(F(X),F(Y),F'(Z),F'(T)) \subseteq equals(\lambda x\,.\,F(x),\lambda x\,.\,F'(x)),\\ equals(\lambda x\,.\,F(x),\lambda x\,.\,F'(x)) \subseteq b(X,Y,Z,T,\lambda x\,.\,F(x))\ ,\ \Delta \end{array} \vdash \Box$$

Where *equals* is supposed to be a fresh function symbol. A similar process can be applied for any number of occurrences of any free variable. The details of such kind of transformations are left for further work.

We can define a matching algorithm for this linear second-order language based on the unification procedure. In this case, we only use the first four rules (the ***flexible-flexible step with distinct heads*** rule is not necessary) and we must take into account that $\stackrel{?}{=}$ is not symmetric now and the substitution ρ applied to a matching problem $S = \{t_1 \stackrel{?}{=} u_1,\dots,t_n \stackrel{?}{=} u_n\}$ only instantiates the left side of the equalities $\rho(S) = \{\rho(t_1) \stackrel{?}{=} u_1,\dots,\rho(t_n) \stackrel{?}{=} u_n\}$. Both changes make the procedure terminating apart from sound and complete. Proof of lemma 5.4 is based on this matching algorithm.

5.6 The Critical Pairs Lemma for Second-Order Bi-rewriting Systems

Second-order bi-rewriting rules are defined, as usual, as pairs of linear second-order terms. However we need to impose two restrictions to second-order bi-rewriting systems.

Definition 5.16 *Given two sets of second-order bi-rewriting rules $R_\subseteq$ and $R_\supseteq$, we say that $\langle R_\subseteq, R_\supseteq\rangle$ is a* **second-order bi-rewriting system** *if any rule $l \xrightarrow{\subseteq} r$ in $R_\subseteq$ and any rule $l \xrightarrow{\supseteq} r$ in $R_\supseteq$ satisfy $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$ and l and r have (the same)* base *type.*

The first restriction is imposed to avoid the infinitely branching problem. The second restriction is required to avoid the introduction of free variables with type order higher than two during the completion process, as it will be motivated later.

Rewriting relations are defined as usual.

Definition 5.17 *We say that t* **rewrites to** *u using the set of bi-rewriting rules R, noted $t \xrightarrow[R]{} u$, if there exists an occurrence p in t, a rule $l \longrightarrow r \in R$, and a substitution σ such that $t|_p = \sigma(l)$ and $u = t[\sigma(r)]_p$.*

We can prove then the following result.

Lemma 5.18 *For any second-order bi-rewriting system we have:*

(i) *If the terms t and u are related by $t \xrightarrow[R]{} u$, then $\mathcal{FV}(u) \subseteq \mathcal{FV}(t)$.*
(ii) *For any term t there are finitely many terms u such that $t \xrightarrow[R]{} u$.*

Proof: (i) If $s \xrightarrow[R]{} t$ then there exists a context $u[\ ]_p$, a rule $l \longrightarrow r \in R$ and a substitution σ such that $s =_{\beta\eta} u[\sigma(l)]_p$ and $t = u[\sigma(r)]_p|_{\beta\eta}$. Relation $=_{\beta\eta}$ preserves free variables in linear second-order λ-calculus (lemma 5.2), therefore we only need to prove $\mathcal{FV}(\sigma(r)) \subseteq \mathcal{FV}(\sigma(l))$. This may be concluded from $\mathcal{FV}(\sigma(s)) = (\mathcal{FV}(s) \backslash \mathcal{D}om(\sigma)) \cup \mathcal{FV}(\sigma)$ and $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$ which holds for any rule $l \longrightarrow r$.

(ii) We can apply finitely many rules $l \longrightarrow r$ on finitely many different positions p of a term s to rewrite it. We only have to consider substitutions σ satisfying $\mathcal{D}om(\sigma) \subseteq \mathcal{FV}(r)$ in order to instantiate r. Now, if $\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$, lemma 5.4 ensures that we only obtain finitely many substitutions σ satisfying $s|_p =_{\beta\eta} \sigma(l)$ and $\mathcal{D}om(\sigma) \subseteq \mathcal{FV}(r) \subseteq \mathcal{FV}(l)$. Any rule, position and substitution determine completely a term $t = s[\sigma(r)]_p|_{\beta\eta}$, thus we will obtain finitely many of them. ■

It means that no new variables are introduced during the rewriting process and it guarantees that the rewriting relation is finitely branching. We can prove, then, that any terminating bi-rewriting system is quasi-terminating (globally terminating).

The use of second-order terms simplifies the definition of critical pairs.

Definition 5.19 *Let $\alpha_1 \xrightarrow{\subseteq} \beta_1$ in $R_{\subseteq}$ and $\alpha_2 \xrightarrow{\supseteq} \beta_2$ in $R_{\supseteq}$ be two second-order bi-rewriting rules (with distinct free variables). If σ belongs to the set of minimum unifiers of α_1 and $F(\alpha_2)$, being F a fresh free variable, then*

$$\sigma(F(\beta_2)) \subseteq \sigma(\beta_1)$$

is a **(second-order) critical pair**. *The same for critical pairs between $R_{\supseteq}$ and $R_{\subseteq}$.*

Nipkow (Nipkow, 1991) cannot define critical pairs in this way because $F(\alpha_2)$ violates his definition of *pattern*. In our case, we have to take into account that the variable F in $\alpha_1 \stackrel{?}{=} F(\alpha_2)$ has to be second-order typed, therefore we have to require all rewriting rules to be first-order typed. If this condition is satisfied, then $\rho(\beta_1)$ and $\rho(F(\beta_2))$ will also be base typed, and if we have to introduce $\rho(F(\beta_2)) \xrightarrow{\subseteq} \rho(\beta_1)$ or $\rho(\beta_1) \xrightarrow{\supseteq} \rho(F(\beta_2))$ as new rewriting rules during the completion process, they will also be base typed.

We can prove then the following critical pairs lemma.

Theorem 5.20 *A terminating second-order bi-rewriting system $\langle R_{\subseteq}, R_{\supseteq} \rangle$ is Church-Rosser if all the second-order critical pairs are bi-confluent.*

Proof: The most general way in which two expressions α_1 and α_2 (the left part of two rules) can overlap is given by $\sigma(F(\alpha_2)) \subseteq \sigma(\alpha_1)$. All these pairs of captured by the definition of critical pair, and as far as when the two left parts of the rules do not overlap the resulting pair is always commutative, we can conclude that the system is locally commutative iff all second-order critical pairs are commutative. The Church-Rosser property is proved by noetherian induction in the usual way. ■

The conditions for the termination of second-order rewriting systems are not studied in this thesis but will be considered in future works. The decidability of the linear second-order unification problem remains as an open question, and it seems not to have an easy answer, although we think it is a decidable problem. These two issues are left as further research work.

5.7 An Example of Completion

To conclude, we illustrate the use of the second-order completion method by means of the example in figure 4.2 (the same example is completed in subsection 4.4.1 for the first-order case). We start with the rules r_1, r_1^{ext}, r_3, r_4 and r_5. The commutativity and associativity properties of the union operator (rules r_4 and r_5) make necessary to consider bi-rewriting modulo a set of inclusions. This theory was developed in section 4.3, and it will not be considered in detail in this example. We shall use a set of non-oriented rules I, and we shall suppose that the second-order unification algorithm can be extended to second-order unification modulo commutativity and associativity.

The initial rules define the inclusion theory of the union, but they do not form a canonical bi-rewriting system. We can generate an extended critical pair unifying the left part of the rule r_3 with a subexpression of the left part of the rule r_1. The solution ρ of this unification problem $F(X \cup Y) \stackrel{?}{=} Z \cup Z$ is used to compute the critical pair $\rho(F(X)) \subseteq \rho(Z)$. This unification problem has two minimum unifiers (up to $\cup$ associativity and commutativity):

$$\begin{aligned}
\rho &= [F \mapsto \lambda x \,.\, x][X \mapsto Z][Y \mapsto Z] \\
\rho &= [F \mapsto \lambda G(x) \cup G(X \cup Y)][Z \mapsto G(X \cup Y)]
\end{aligned}$$

Let us see how they are computed using the unification procedure:

$$\begin{array}{ll}
\multicolumn{2}{l}{\langle\{F(X \cup Y) \stackrel{?}{=} Z \cup Z\}\ ,\ Id\rangle} \\
\Rightarrow_{projection} & \langle\{X \cup Y \stackrel{?}{=} Z \cup Z\}\ ,\ [F \mapsto \lambda x \,.\, x]\rangle \\
\Rightarrow_{rigid-rigid} & \langle\{X \stackrel{?}{=} Z, Y \stackrel{?}{=} Z\}\ ,\ [F \mapsto \lambda x \,.\, x]\rangle \\
\Rightarrow_{imitation} & \langle\{Y \stackrel{?}{=} Z\}\ ,\ [F \mapsto \lambda x \,.\, x][X \mapsto Z]\rangle \\
\Rightarrow_{imitation} & \langle\emptyset\ ,\ [F \mapsto \lambda x \,.\, x][X \mapsto Z][Y \mapsto Z]\rangle
\end{array}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{\langle\{F(X \cup Y) \stackrel{?}{=} Z \cup Z\}\ ,\ Id\rangle} \\
\Rightarrow_{imitation} & \langle\{H_1(X \cup Y) \stackrel{?}{=} Z, H_2 \stackrel{?}{=} Z\}\ ,\ [F \mapsto \lambda x \,.\, H_1(x) \cup H_2]\rangle \\
\Rightarrow_{imitation} & \langle\{H_2 \stackrel{?}{=} Z\}\ ,\ [F \mapsto \lambda x \,.\, H_1(x) \cup H_2][Z \mapsto H_1(X \cup Y)]\rangle \\
\Rightarrow_{imitation} & \langle\emptyset\ ,\ [F \mapsto \lambda x \,.\, H_1(x) \cup H_2][Z \mapsto H_1(X \cup Y)][Z \mapsto H_2]\rangle
\end{array}$$

These two unifiers generate two critical pairs. The first one is confluent. The second one makes necessary to introduce the following rule:

$$r_5 : G(X) \cup G(X \cup Y) \xrightarrow{\subseteq} G(X \cup Y)$$

This rule generates new critical pairs with r_3 (the only rule belonging to $R_{\supseteq}$), the following one among them:

$$r_6 : G(X) \cup G(Y) \xrightarrow{\subseteq} G(X \cup Y)$$

This new rule r_6 subsumes r_5.

Contrary to the previous cases, the orientation of the rule r_6 is no so clear, but we do not consider the problem of orienting second-order rules in this thesis. Nevertheless, r_6 generates new critical pairs with r_3. The unification problem of the left part of r_6, $G(X') \cup G(Y')$, and the variable F applied to the left part of r_1, $X \cup Y$, has the following unifiers (up to $\cup$ commutativity and associativity, and interchange of X' and Y'):

$$\begin{array}{lcl}
\rho & = & [F \mapsto \lambda x \,.\, x][X \mapsto G(X')][Y \mapsto G(Y')] \\
\rho & = & [F \mapsto \lambda x \,.\, H(x, X') \cup H(X \cup Y, Y')][G \mapsto \lambda x \,.\, H(X \cup Y, x)] \\
\rho & = & [F \mapsto \lambda x \,.\, G(H(x)) \cup G(Y')][X' \mapsto H(X \cup Y)] \\
\rho & = & [F \mapsto \lambda x \,.\, H(x) \cup H(I(Y') \cup Y)][X \mapsto I(X')][G \mapsto \lambda x \,.\, H(I(x) \cup Y)] \\
\rho & = & [F \mapsto \lambda x \,.\, H(x) \cup H(I(Y') \cup X)][X \mapsto I(X')][G \mapsto \lambda x \,.\, H(I(x) \cup X)] \\
\rho & = & [F \mapsto \lambda x \,.\, G(x) \cup G(Y')][X' \mapsto X \cup Y]
\end{array}$$

They generate the following critical pairs:

$$\begin{array}{l}
G(X') \subseteq G(X' \cup Y) \\
H(X, X') \cup H(X \cup Y, Y') \subseteq H(X \cup Y, X' \cup Y') \\
G(H(X)) \cup G(Y') \subseteq G(H(X \cup Y) \cup Y') \\
H(I(X')) \cup H(I(Y') \cup Y) \subseteq H(I(X' \cup Y') \cup Y) \\
H(Y) \cup H(I(Y') \cup Y) \subseteq H(I(X' \cup Y') \cup Y) \\
G(X) \cup G(Y') \subseteq G(X \cup Y \cup Y')
\end{array}$$

All of them, except the second one, are confluent. The second one makes necessary to introduce the following rule:

$$r_7 : H(X, X') \cup H(X \cup Y, Y') \xrightarrow{\subseteq} H(X \cup Y, X' \cup Y')$$

Again this rule generates a new critical pair with r_3 and requires introducing:

$$r_8 : H(X, X') \cup H(Y, Y') \xrightarrow{\subseteq} H(X \cup Y, X' \cup Y')$$

which subsumes r_7 (the rule r_7 can be decomposed into one application of r_8 followed by an application of r_1).

In this way we introduce, among others, the following infinitely many rules:

$$H(X_1, \ldots, X_n) \cup H(Y_1, \ldots, Y_n) \xrightarrow{\subseteq} H(X_1 \cup Y_1, \ldots, X_n \cup Y_n)$$

The bi-rewriting system can not be completed in this way. A solution to prevent the non-termination of this completion process is using β-reduction explicitly. We use now three symbols in the signature:

$$\begin{array}{ll} \cup : & \tau \to \tau \to \tau \\ \texttt{lambda} : & (\tau \to \tau) \to \tau \\ \texttt{apply} : & \tau \to \tau \to \tau \end{array}$$

and the following initial set of rules:

$$\begin{array}{ll} r_1 : & X \cup X \xrightarrow{\subseteq} X \\ r_1^{ext} : & X \cup X \cup Y \xrightarrow{\subseteq} X \cup Y \\ r_2 : & \texttt{apply}(\texttt{lambda}(F), X) \xrightarrow{\subseteq} F(X) \\ r_3 : & X \cup Y \xrightarrow{\supseteq} X \\ r_4 : & \texttt{apply}(\texttt{lambda}(F), X) \xrightarrow{\supseteq} F(X) \end{array}$$

All standard critical pairs of this system are bi-confluent, thus we have to concentrate our attention on two cases, the critical pairs obtained by overlapping the repeated variable of rule r_1 (or of rule r_1^{ext}) with the left part of rule r_3 in the first case and with r_4 in the second case. In the second case, as far as the rule r_4 also appears in the other rewriting system (as rule r_2), all *extended* critical pairs generated by it will be trivially bi-confluent. Therefore, we only have to consider the extended critical pair generated by r_1 and r_3, i.e.:

$$r_5 : \; F(X) \cup F(X \cup Y) \xrightarrow{\subseteq} F(X \cup Y)$$

As we know, this rule generates a new rule r_6 which properly oriented subsumes r_5.

$$r_6 \; F(X) \cup F(Y) \xrightarrow{\subseteq} F(X \cup Y)$$

This rule is non-left-linear and may initiate an infinite sequence of non-confluent critical pairs, as we have seen. However, it also generates a standard critical pair with r_4. It is interesting to see that, using second-order bi-rewriting systems, we can generate standard critical pairs between rules not sharing any symbol of the signature. The reader can figure out that the same happens dealing with equational second-order rewriting systems.

Let's concentrate our attention on this standard critical pair. It is obtained unifying $H(\texttt{apply}(\texttt{lambda}(F), X))$ and $G(Y) \cup G(Z)$ using:

$$\begin{array}{rl} \sigma = & [H \mapsto \lambda x \,.\, H_1(x) \cup H_1(\texttt{apply}(Z, X))] \\ & [G \mapsto \lambda x \,.\, H_1(\texttt{apply}(x, X))] \\ & [Y \mapsto \texttt{lambda}(F)] \end{array}$$

The resulting rule is:

$$r_7 : H_1(F(X)) \cup H_1(\texttt{apply}(Z, X)) \xrightarrow{\subseteq} H_1(\texttt{apply}(\texttt{lambda}(F) \cup Z, X))$$

This rule generates a new critical pair with r_4 which introduces r_8, and r_8 a critical pair with r_3 which introduces r_9, and finally r_9 a critical pair with r_3

which introduces r_{10}.

$$\begin{aligned}
r_8 :\ & H_1(F(X)) \cup H_1(G(X)) \xrightarrow{\subseteq} H_1(\texttt{apply}(\texttt{lambda}(F) \cup \texttt{lambda}(G), X)) \\
r_9 :\ & H_1(F(H_2(X))) \cup H_1(G(H_2(X \cup Y)) \xrightarrow{\subseteq} \\
& \qquad \xrightarrow{\subseteq} H_1(\texttt{apply}(\texttt{lambda}(F) \cup \texttt{lambda}(G), H_2(X \cup Y))) \\
r_{10} :\ & H_1(F(H_2(X))) \cup H_1(G(H_2(Y))) \xrightarrow{\subseteq} \\
& \qquad \xrightarrow{\subseteq} H_1(\texttt{apply}(\texttt{lambda}(F) \cup \texttt{lambda}(G), H_2(X \cup Y)))
\end{aligned}$$

It is easy to see that we only need the instance of r_{10} obtained by $[H_1 \mapsto \lambda x\,.\,x][H_2 \mapsto \lambda x\,.\,x]$ to subsume rule r_5 and to make bi-confluent all critical pairs obtained from it.

$$r'_{10} :\ F(X) \cup G(Y) \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(F) \cup \texttt{lambda}(G), X \cup Y)$$

However, this rule generates new critical pairs with r_4 which introduce the following rules.

$$\begin{aligned}
r_{11} :\ & F(G(X)) \cup H(Y) \xrightarrow{\subseteq} \\
& \quad \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(\texttt{apply}(x, X))) \cup \texttt{lambda}(H), \texttt{lambda}(G) \cup Y) \\
r_{12} :\ & F(G(X)) \cup H(I(Y)) \xrightarrow{\subseteq} \\
& \quad \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(\texttt{apply}(x, X))) \cup \texttt{lambda}(\lambda x\,.\,H(\texttt{apply}(x, Y))), \\
& \qquad\qquad \texttt{lambda}(G) \cup \texttt{lambda}(I))
\end{aligned}$$

Rule r_{12} concludes the completion process which results in a finite canonical bi-rewriting system shown in figure 5.1.

$$R_{\subseteq} = \begin{cases}
X \cup X \xrightarrow{\subseteq} X \\
\texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(x)), X) \xrightarrow{\subseteq} F(X) \\
F(X) \cup G(Y) \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(x)) \cup \texttt{lambda}(\lambda x\,.\,G(x)), X \cup Y) \\
F(G(X)) \cup H(Y) \xrightarrow{\subseteq} \\
\quad \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(\texttt{apply}(x, X))) \cup \texttt{lambda}(H), \texttt{lambda}(G) \cup Y) \\
F(G(X)) \cup H(I(Y)) \xrightarrow{\subseteq} \\
\quad \xrightarrow{\subseteq} \texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(\texttt{apply}(x, X))) \cup \texttt{lambda}(\lambda x\,.\,H(\texttt{apply}(x, Y))), \\
\qquad\qquad \texttt{lambda}(G) \cup \texttt{lambda}(I))
\end{cases}$$

$$R_{\supseteq} = \begin{cases}
X \cup Y \xrightarrow{\supseteq} X \\
\texttt{apply}(\texttt{lambda}(\lambda x\,.\,F(x)), X) \xrightarrow{\supseteq} F(X)
\end{cases}$$

$$I = \begin{cases}
X \cup Y \xleftrightarrow{\subseteq} Y \cup X \\
(X \cup Y) \cup Z \xleftrightarrow{\subseteq} X \cup (Y \cup Z)
\end{cases}$$

Figure 5.1: A canonical higher-order bi-rewriting system for the inclusion theory of the union with β-reduction.

5.8 Conclusions

The use of higher-order terms in rewriting systems introduces some problems. Some of them have been expounded in this chapter. Because of that, there is not a unique proposal of higher-order rewriting system in the literature. We have discussed some of them and we have also proposed a definition of second-order bi-rewriting systems based on the use of the linear second-order typed λ-calculus. This proposal can also be seen as a new kind of higher-order rewriting system. We have described a new sound and complete second-order unification procedure for such restricted second-order language. This procedure avoids the use of the iteration and elimination transformation rules of the general second-order unification procedure defined in (Jensen and Pietrzykowski, 1976). These transformation rules, in the general case, always make the procedure non terminating. Unfortunately, the decidability of our unification problem is still an open question and the termination of the procedure we have described is not guaranteed. Anyway, in the examples we have completed, the procedure always finishes and is therefore usable.

Chapter 6

Implementing Nondeterministic Specifications

Abstract: In this chapter we show the applicability of bi-rewriting systems to the verification of nondeterministic specifications. If nondeterministic specifications are viewed as inclusion specifications, then bi-rewriting systems are a sound and complete deduction method with respect to the class of models based on preorders. However, the models usually proposed for these specifications are multialgebras, and both classes of models are not equivalent. We show how a nondeterministic specification can be completed in order to get the equivalence between both semantics. We see also that these requirements prove the initiality of a model based on sets of normal forms. Moreover, the completion process does not modify the rewriting relation $\xrightarrow[R_{\supseteq}]{}$ used to model the nondeterministic computation.

6.1 Introduction

It is well known that term rewriting techniques can be used to test the equivalence of terms in a equational logic specification E. The method consists in finding the normal form of both sides of the tested equality and checking if they are equal. The method is sound and complete for ground terms if the set of ground normal forms is an initial model of the specification; and for terms with variables if the set of normal forms is isomorphic to $\mathcal{T}(\Sigma, \mathcal{X})/E$ (Dershowitz and Jouannaud, 1990). It is also well known that the confluence and termination of the rewriting system resulting from orienting the equations is a sufficient condition for this completeness result.

Term rewriting techniques have also been proposed as the implementation language of nondeterministic specifications (Kaplan, 1986a; Hussmann, 1992).

In all these approaches the signature includes a nondeterministic choice operator —noted by $\uparrow$ in (Kaplan, 1986a; Kaplan, 1988), by $or(\_,\_)$ in (Hussmann, 1991; Hussmann, 1992), or by $\cup$ in our work— which makes nondeterministic computation loose the symmetry property. Otherwise, the rules $X \cup Y \longrightarrow X$ and $X \cup Y \longrightarrow Y$ proposed for the choice operator would allow to prove the equivalence of any two terms. Therefore, the confluence property makes no sense, and a nondeterministic specification is presented in general as a set of (non symmetric) inclusions. This suggests the use of bi-rewriting systems to verify such nondeterministic specifications. However, the models of inclusion specifications are based on preorder algebras, whereas the models of nondeterministic specifications are based on multialgebras, and both classes of models are not equivalent (as we show in a counter-example).

In section 6.2, we describe how a nondeterministic specification can be completed in order to obtain the same semantics using the preorder or the multialgebra class of models. In section 6.3 we prove that the set of normal forms (w.r.t. the rewriting relation $\xrightarrow[R_{\supseteq}]{}$) of a completed nondeterministic specifications forms an initial model of the specification. Moreover, the method consisting on, given a pair of terms a and b, finding the set of normal forms of both terms and checking if one set is included into the other one is a sound and complete method to test $a \subseteq b$. Theorem 6.16 summarizes all these results. Finally, in section 6.4 we show by means of an example how a nondeterministic specification can be completed.

6.2 Using Bi-rewriting Systems to Verify Specifications

The models usually proposed for nondeterministic specifications are based on Σ-multialgebras (Hesselink, 1988; Nipkow, 1986), which capture the essence of nondeterminism better than the Σ-algebras used in equational specifications.

Definition 6.1 *A* **Σ-multialgebra** *A is a tuple $\langle S^A, \mathcal{F}^A \rangle$ where S^A is a non empty carrier set, and $\mathcal{F}^A$ is a set of set-valued functions $f^A : S^A \times \overset{n}{\ldots} \times S^A \to \mathcal{P}^+(S^A)$ for each $f \in \Sigma^n$ function symbol of the signature.*

Models are defined as follows.

Definition 6.2 *Given a nondeterministic specification I over a signature Σ, a Σ-multialgebra A is said to be a model of I, noted $A \in MAlg(I)$, if the interpretation function $I^A_{\_}[\_] : (\mathcal{X} \to S^A) \to \mathcal{T}(\Sigma, \mathcal{X}) \to \mathcal{P}^+(S^A)$ defined inductively by*[1]

$$\begin{array}{lcll} I^A_\rho[x] & = & \{\rho(x)\} & \textit{for any } x \in \mathcal{X} \\ I^A_\rho[f(t_1,\ldots,t_n)] & = & \bigcup\{f^A(v_1,\ldots,v_n) \mid v_i \in I^A_\rho[t_i]\} & \textit{for any } f \in \Sigma^n \end{array}$$

[1]$\mathcal{P}^+(S)$ denotes the set of nonempty subsets of S, and $\bigcup S$, where S is a set of sets, denotes the union of all the elements belonging to S, i.e. $\bigcup_{x \in S} X$.

satisfies $I^A_\rho[t] \subseteq I^A_\rho[u]$ for any axiom $t \subseteq u$ in the specification I, and any valuation function $\rho : \mathcal{X} \to S^A$.
An inclusion $t \subseteq u$ is valid in a Σ-multialgebra model A, noted $A \models t \subseteq u$, if for any valuation ρ we have $I^A_\rho[t] \subseteq I^A_\rho[u]$.

Bi-rewriting systems introduced in chapters 4 and 5 automate the deduction in the *Partial Order Logic* POL (also for the rewriting logic of Meseguer (Meseguer, 1992)). The models of this logic are preorder algebras, defined as follows.

Definition 6.3 *A Σ-**preorder algebra** A is a triplet $\langle S^A, \subseteq_A, \mathcal{F}^A \rangle$ where S^A is a carrier set, $\subseteq_A$ is a preorder relation and $\mathcal{F}^A$ is a set of monotonic functions $f^A : S^A \times \overset{n}{\ldots} \times S^A \to S^A$ for each symbol $f \in \Sigma^n$.*

Definition 6.4 *Given a specification I over Σ a Σ-preorder algebra A is said to be a model of I, noted $A \in POAlg(I)$, if the interpretation function $I^A_{\_}[\_] : (\mathcal{X} \to S^A) \to \mathcal{T}(\Sigma, \mathcal{X}) \to S^A$ defined inductively by*

$$\begin{array}{lcll} I^A_\rho[x] & = & \rho(x) & \text{for any } x \in \mathcal{X} \\ I^A_\rho[f(t_1, \ldots, t_n)] & = & f^A(I^A_\rho[t_1], \ldots, I^A_\rho[t_n]) & \text{for any } f \in \Sigma^n \end{array}$$

satisfies $I_\rho[t] \subseteq_A I_\rho[u]$ for any axiom $t \subseteq u$ in the specification I and any valuation $\rho : \mathcal{X} \to S^A$.

A soundness and completeness theorem, similar to the Birkhoff theorem, can be stated for this logic.

Lemma 6.5 *For any specification I and any pair of terms t and u we have $POAlg(I) \models t \subseteq u$ iff $I \vdash_{POL} t \subseteq u$.*

Commutative and terminating bi-rewriting systems automate the deduction in $\vdash_{POL}$. They are a sound and complete method w.r.t. the semantics of specifications based on preorder algebras. However, $POAlg(I) \models t \subseteq u$ and $MAlg(I) \models t \subseteq u$ are not equivalent (the implication does not hold in none of both directions) as the following counter-example shows.

Example. A counter-example to $MAlg(I) \models t \subseteq u \Rightarrow POAlg(I) \models t \subseteq u$ is given by the following additivity axiom which is sound in multialgebra models, but not in preorder algebra models.

$$\frac{}{f(X \cup Y) \subseteq f(X) \cup f(Y)}\,Aditivity$$

The counter-example to $POAlg(I) \models t \subseteq u \Rightarrow MAlg(I) \models t \subseteq u$ is not so evident, and causes more problems. The following substitution rule is sound in preorder models, but not in multialgebra models, in the presence of repeated variables.

$$\frac{t \subseteq u}{\sigma(t) \subseteq \sigma(u)}\,Substitution$$

For instance, the deduction

$$f(X,X) \subseteq g(X)\ ,\ X \subseteq X \cup Y\ ,\ Y \subseteq X \cup Y\ \vdash_{POL}\ f(X,Y) \subseteq g(X \cup Y)$$

is correct in POL. However, it is not sound in a multialgebra model. The multialgebra $A = \langle S^A, \mathcal{F}^A \rangle$ defined by:

$$S^A = \{a,b\} \qquad \begin{array}{lcl} f^A(x,y) & = & \mathit{if}\ x = y\ \mathit{then}\ \{a\}\ \mathit{else}\ \{b\} \\ g^A(x,y) & = & \{a\} \\ x \cup^A y & = & \{x,y\} \end{array}$$

is a model of $I = \{f(X,X) \subseteq g(X), X \subseteq X \cup Y, Y \subseteq X \cup Y\}$, however $I^A_\rho[f(X,Y)] \not\subseteq I^A_\rho[g(X \cup Y)]$ for $\rho = [a \leftarrow X, b \leftarrow Y]$.

We understand variables in a specification denoting terms and being universally quantified. Therefore, we think that the substitution rule has to be sound in any specification model. Multialgebra models may satisfy this requirement if we modify the definition of interpretation and model:

Definition 6.6 *A Σ-multialgebra A is said to be a* **strong model** *of a specification I, noted $A \in \overline{MAlg}(I)$, if the interpretation function $I^A_{\_}[\_] : (\mathcal{X} \to \mathcal{P}^+(S^A)) \to \mathcal{T}(\Sigma,\mathcal{X}) \to \mathcal{P}^+(S^A)$ defined inductively by*

$$\begin{array}{lcll} I^A_\rho[x] & = & \rho(x) & \mathit{for\ any}\ x \in \mathcal{X} \\ I^A_\rho[f(t_1,\ldots,t_n)] & = & \bigcup\{f^A(v_1,\ldots,v_n) \mid v_i \in I^A_\rho[t_i]\} & \mathit{for\ any}\ f \in \Sigma^n \end{array}$$

satisfies $I_\rho[t] \subseteq I_\rho[u]$ for any axiom $t \subseteq u$ in the specification I, and any valuation $\rho : \mathcal{X} \to \mathcal{P}^+(S^A)$.

Notice that the valuation function ρ now ranges over sets and not over values.

Lemma 6.7 *For any specification I we have $\overline{MAlg}(I) \subseteq MAlg(I)$.*

Using this smaller class of models the preorder logic entailment $\vdash_{POL}$ becomes sound in it.

Theorem 6.8 *If $POAlg(I) \models t \subseteq u$ holds, then $\overline{MAlg}(I) \models t \subseteq u$ also holds. Therefore, bi-rewriting is a sound deduction method.*

Proof: It is sufficient to prove that

$$\forall A \in \overline{MAlg}\,.\,\exists B \in POAlg\,.\,(\forall \rho\,.\,I^A_\rho[t] \subseteq I^A_\rho[u]) \Leftrightarrow (\forall \rho'\,.\,I^B_{\rho'}[t] \subseteq_B I^B_{\rho'}[u])$$

Notice that we use one implication direction to prove $A \in \overline{MAlg}(I) \Rightarrow B \in POAlg(I)$, and the opposite direction to prove $B \models t \subseteq u \Rightarrow A \models t \subseteq u$.

Any Σ-multialgebra A has a Σ-preorder algebra B naturally associated. This preorder algebra B is defined by

$$\begin{array}{l} S^B \overset{def}{=} \mathcal{P}^+(S^A) \\ f^B(s_1,\ldots,s_n) \overset{def}{=} \bigcup\{f^A(v_1,\ldots,v_n) \mid v_i \in s_i\} \text{ for any } f \in \Sigma^n \end{array}$$

The carrier S_B is a power set, and the set inclusion relation $\subseteq$ used in the multialgebra model A, and the partial order relation $\subseteq_B$ used in the preorder model B are equal. We can prove by structural induction on the term t that $I^A_\rho[t] = I^B_\rho[t]$.

$$\begin{aligned} I^B_\rho[x] &= \rho(x) = I^A_\rho[x] \\ I^B_\rho[f(t_1 \dots t_n)] &= f^B(I^B_\rho[t_1] \dots I^B_\rho[t_n]) = \bigcup\{f^A(v_1 \dots v_n) \mid v_i \in I^B_\rho[t_i]\} \\ &= \bigcup\{f^A(v_1 \dots v_n) \mid v_i \in I^A_\rho[t_i]\} = I^A_\rho[f(t_1 \dots t_n)] \end{aligned}$$

Then the initial double implication becomes a tautology. ■

In the following we will study which conditions I has to satisfy in order to be $POAlg(I) \models t \subseteq u$ and $\overline{MAlg}(I) \models t \subseteq u$ equivalent.

Theorem 6.9 *If the specification I satisfies:*

(i) *I contains the union theory as a subtheory:*
$I \vdash_{POL} X \cup X \subseteq X,\ X \subseteq X \cup Y,\ Y \subseteq X \cup Y$.

(ii) *$I \vdash_{POL} t = \cup\{u \in Atomic(I) \mid I \vdash_{POL} u \subseteq t\}$, for any term t, where $Atomic(I) \stackrel{def}{=} \{u \in \mathcal{T}(\Sigma, \mathcal{X}) \mid \text{ if } I \vdash_{POL} v \subseteq u \text{ then } v = u\}$.*

(iii) *$I \vdash_{POL} f(\dots t \cup u \dots) \subseteq f(\dots t \dots) \cup f(\dots u \dots)$ for any n-ary symbol $f \in \Sigma^n$.*

(iv) *If $t \in Atomic(I)$ and $I \vdash_{POL} t \subseteq u \cup u'$ then either $I \vdash_{POL} t \subseteq u$ or $I \vdash_{POL} t \subseteq u'$.*

Then, whenever $\overline{MAlg}(I) \models t \subseteq u$ holds, then $POAlg(I) \models t \subseteq u$ also holds. Therefore, bi-rewriting is a complete deduction method for these specifications.

Proof: It is sufficient to prove that

$$\forall B \in POAlg \,.\, \exists A \in \overline{MAlg} \,.\, (\forall \rho \,.\, I^A_\rho[t] \subseteq I^A_\rho[u]) \Leftrightarrow (\forall \rho' \,.\, I^B_{\rho'}[t] \subseteq_B I^B_{\rho'}[u])$$

We can also associate a multialgebra A to each preorder algebra B as follows.

$$\begin{aligned} &S^A \stackrel{def}{=} Atomic(S^B) \\ &f^A(v_1, \dots, v_n) \stackrel{def}{=} \{s \in S^A \mid s \subseteq_B f^B(v_1, \dots, v_n)\} \text{ for any } f \in \Sigma^n \end{aligned}$$

where for any preorder S, we define $Atomic(S) \stackrel{def}{=} \{s \in S \mid s' \subseteq s \Rightarrow s = s'\}$.[2]

Notice that in this case $\subseteq$ is the set inclusion in $\mathcal{P}^+(S^B)$, and $\subseteq_B$ is a preorder relation on S^B, and they are different relations.

Case $\forall \rho' \,.\, \exists \rho \,.\, I^A_\rho[t] \subseteq I^A_\rho[u] \Rightarrow I^B_{\rho'}[t] \subseteq_B I^B_{\rho'}[u]$.

The conditions of the theorem can be translated directly to properties of the preorder algebra B:

$$\begin{aligned} &v \cup^B v \subseteq_B v \quad v_1 \subseteq_B v_1 \cup^B v_2 \quad v_2 \subseteq_B v_1 \cup^B v_2 \\ &f^B(\dots v_1 \cup^B v_2 \dots) \subseteq_B f^B(\dots v_1 \dots) \cup^B f^B(\dots v_2 \dots) \\ &v = \cup^B\{v' \in Atomic(S^B) \mid v' \subseteq_B v\} \\ &v \in Atomic(S^B) \ \wedge\ v \subseteq v_1 \cup v_2 \ \Rightarrow\ v \subseteq v_1 \ \vee\ v \subseteq v_2 \end{aligned}$$

[2]Notice that for the free algebra of terms $\mathcal{T}(\Sigma, \mathcal{X})/I$ this definition and the previous one becomes equivalent.

If we define $\rho(x) \stackrel{def}{=} \{s \in S^A \mid s \subseteq_B \rho'(x)\}$ then using the properties below we can prove by structural induction on the term t that

$$I^B_{\rho'}[t] = \cup^B I^A_{\rho}[t]$$

where, as usual $\cup^B\{v_1, \ldots, v_n\} = v_1 \cup^B \cdots \cup^B v_n$ for any $v_1 \ldots v_n \in S^B$.

Then the monotonicity of $\cup^B$ proves that $I^A_{\rho}[t] \subseteq I^A_{\rho}[u]$ implies $I^B_{\rho'}[t] \subseteq I^B_{\rho'}[u]$.

Case $\forall \rho\,.\,\exists \rho'\,.\,I^B_{\rho'}[t] \subseteq_B I^B_{\rho'}[u] \Rightarrow I^A_{\rho}[t] \subseteq I^A_{\rho}[u]$.

The last two conditions of the theorem prove that if $t \in \mathit{Atomic}(I)$ and $I \vdash_{POL} t \subseteq f(u_1, \ldots, u_n)$ then there exist $v_1, \ldots, v_n \in \mathit{Atomic}(I)$ such that $I \vdash_{POL} t \subseteq f(v_1, \ldots, v_n)$ for any $f \in \Sigma^n$.

If we define $\rho'(x) = \cup^B \rho(x)$ then we can prove

$$I^A_{\rho}[t] = \{s \in S^A \mid s \subseteq_B I^B_{\rho'}[t]\}$$

for any term t by structural induction.

Then $I^B_{\rho'}[t] \subseteq_B I^B_{\rho'}[u]$ implies $I^A_{\rho}[t] \subseteq I^A_{\rho}[u]$.

■

The conditions of the previous theorem are usually satisfied in any nondeterministic specification I. We will find the same conditions in the next subsection where we try to prove the existence and initiality of a model based on sets of normal forms.

6.3 Characterizing Terms by Sets of Normal Forms

In nondeterministic computations terms can not be characterized by a unique normal form, but we will try to characterize them by its set of normal forms. In this case, a method to test inclusions of terms in a nondeterministic specification would consist in searching the set of normal forms of each side of the inclusion, and checking if one set is included in the other one. We will prove that the soundness and completeness of this *nondeterministic computation* method relies on the existence and initiality of a model of *set of normal forms* –like in the equational case with the normal form model–. The main goal of this section is to give the conditions for the existence and for the initially of this model –like it is characterized in the equational case by the confluence and termination properties–.

First we will present the formal definition of the *set of normal forms* model, SNF-model for short, and later we will study the *nondeterministic computation* method, NDC-method for short.

Nondeterministic computation is based on the computation of normal forms only using the rewriting system $R_{\supseteq}$. As we will see, the other rewriting system

$R_{\subseteq}$ does not play a computational role, but its rules may be understood as semantic constraints on the class of models of the specification. The example at the end of the section shows this clearly. Adding new rules to $R_{\subseteq}$ we can prove a soundness and completeness result for the nondeterministic computation and the bi-rewriting methods w.r.t. the models of the extended specification.

Given a rewriting system $R_{\supseteq}$, we will denote the set of its $R_{\supseteq}$-normal forms by $NF^{\supseteq}$ and the set of $R_{\supseteq}$-normal forms of a term t by $NF^{\supseteq}[t]$.

The *set of normal forms* multialgebra, SNF-multialgebra for short, is defined as follows.

Definition 6.10 *Given a rewriting system $R_{\supseteq}$, the* **SNF-multialgebra** *$SNF = \langle S^{SNF}, \mathcal{F}^{SNF} \rangle$ is defined by the carrier set $S^{SNF} \overset{def}{=} NF^{\supseteq}$, and the set of functions $f^{SNF} : NF^{\supseteq} \times \overset{n}{\ldots} \times NF^{\supseteq} \longrightarrow \mathcal{P}^{+}(NF^{\supseteq})$ defined by $f^{SNF}(t_1, \ldots, t_n) = NF^{\supseteq}[f(t_1, \ldots, t_n)]$ for each functional symbol $f \in \Sigma^n$ of the signature.*

Notice that the SNF-multialgebra is defined syntactically using $R_{\supseteq}$, and independently of I. The rewriting rules of $R_{\supseteq}$ come from the orientation of the axioms of I, as explained in previous chapters. However, this fact is not enough to prove that the SNF-multialgebra is a multialgebra model of I.

Lemma 6.11 *Given a specification I and a rewriting system $R_{\supseteq}$, if the following conditions hold.*

(i) *For any inclusion $t \subseteq u$ in I, and any substitution $\rho : \mathcal{X} \longrightarrow NF^{\supseteq}$, we have $NF^{\supseteq}[\rho(t)] \subseteq NF^{\supseteq}[\rho(u)]$.*

(ii) *If $t \in NF^{\supseteq}[f(\ldots, u, \ldots)]$, then there exists $u' \in NF^{\supseteq}[u]$ such that $t \in NF^{\supseteq}[f(\ldots, u', \ldots)]$.*

then the SNF-multialgebra is a multialgebra model of I, $SNF \in MAlg(I)$, and the interpretation function is $I^{SNF}_{\rho}[t] = NF^{\supseteq}[\rho(t)]$.

Additionally, if the following condition also holds

(iii) *$NF^{\supseteq}[t \cup u] \subseteq NF^{\supseteq}[t] \cup NF^{\supseteq}[u]$,*

then the SNF-multialgebra is a strong multialgebra model of I, $SNF \in \overline{MAlg}(I)$, and $I^{SNF}_{\rho}[t] = NF^{\supseteq}[\rho'(t)]$, where for any $x \in \mathcal{X}$, $\rho'(x) = \cup\rho(x)$.

Proof: First we prove that $I^{SNF}_{\rho}[t] = NF^{\supseteq}[\rho(t)]$ are equal. That is, $NF^{\supseteq}[\rho(t)]$ satisfies the inductive definition of multialgebra interpretation function: 1) $I^{SNF}_{\rho}[x] = \rho(x)$ for any variable $x \in \mathcal{X}$. As far as ρ maps variables to normal forms, $NF^{\supseteq}[\rho(x)] = \{\rho(x)\}$. 2) $I^{SNF}_{\rho}[f(t_1, \ldots, t_n)] = \bigcup\{f^{SNF}(v_1, \ldots, v_n) \mid v_i \in I^{SNF}_{\rho}[t_i]\}$, which is equivalent to $NF^{\supseteq}[f(\rho(t_1), \ldots, \rho(t_n))] = \bigcup\{NF^{\supseteq}[f(v_1, \ldots, v_n)] \mid v_i \in NF^{\supseteq}[\rho(t_i)]\}$. The inclusion $\supseteq$ is always satisfied and it can be proved using the monotonicity of f. The inclusion $\subseteq$ is proved by the second condition of the lemma.

Second the first condition of the lemma and $I^{SNF}_{\rho}[t] = NF^{\supseteq}[\rho(t)]$ prove that $I^{SNF}_{\rho}[t] \subseteq I^{SNF}_{\rho}[u]$ for any inclusion $t \subseteq u$ of I, and any substitution ρ.

The proof of the second part of the lemma is quite similar. In this case we need the third condition to prove $I^{SNF}_{\rho}[t] = \rho(x) = NF^{\supseteq}[\cup\rho(x)] = NF^{\supseteq}[\rho'(x)]$. ■

As we have seen in the previous subsection we can associate a preorder algebra to the SNF-multialgebra, and this preorder algebra will be a preorder model of I if the SNF-multialgebra is a strong multialgebra model of I.

Lemma 6.12 *If the following conditions are satisfied:*
(i) *If $I \vdash_{POL} t \subseteq u$ then $NF^{\supseteq}[t] \subseteq NF^{\supseteq}[u]$.*
(ii) *If $t \in NF^{\supseteq}[f(\ldots, u, \ldots)]$, then there exists $u' \in NF^{\supseteq}[u]$ such that $t \in NF^{\supseteq}[f(\ldots, u', \ldots)]$.*
(iii) *$NF^{\supseteq}[t \cup u] \subseteq NF^{\supseteq}[t] \cup NF^{\supseteq}[u]$,*
then, the SNF-preorder algebra defined by the carrier set $S^{SNF} \overset{def}{=} \mathcal{P}^{+}(NF^{\supseteq})$ and the set of functions $f^{SNF}(s_1 \ldots s_n) \overset{def}{=} \cup\{NF^{\supseteq}[f(v_1 \ldots v_n)] \mid v_i \in s_i\}$ is a preorder model of I.
If in addition
(iv) *If $NF^{\supseteq}[t] \subseteq NF^{\supseteq}[u]$ then $I \vdash_{POL} t \subseteq u$.*
then the SNF-preorder model is initial in $POAlg(I)$, and the associated SNF-multialgebra is initial in $\overline{MAlg}(I)$.
Moreover, $\overline{MAlg}(I) \models t \subseteq u$ and $POAlg(I) \models t \subseteq u$ are equivalent.

Proof: The proof of the first part of the lemma is a consequence of the previous lemma. The proof for the initiality of the model relies on the completeness of $\vdash_{POL}$ w.r.t. the class of models *POAlg*. The initiality of the model SNF w.r.t. the class $POAlg(I)$, and the fact that its associated multialgebra is a strong multialgebra model of I proves the last equivalence. ■

The conditions of this lemma reproduce the condition of theorem 6.9. Before reducing the four conditions of this lemma to syntactic conditions more easily provable, we will discuss its meaning.

The first condition $NF^{\supseteq}[t_1] \supseteq NF^{\supseteq}[t_2] \Rightarrow I \vdash_{POL} t_1 \supseteq t_2$ expresses the soundness of the NDC-method with respect to the specification. However, the user usually only gives the rewriting rules $R_{\supseteq}$, leaving the specification incomplete –as we will see in the examples–. This specification must be completed in order to verify this condition. Hence, we prefer to name this condition completeness of the specification with respect to the NDC-method.

The forth condition $I \vdash_{POL} t_1 \supseteq t_2 \Rightarrow NF^{\supseteq}[t_1] \supseteq NF^{\supseteq}[t_2]$ expresses the completeness of the method with respect to the specification. This condition is very easily satisfied. As it is noticed by Hussmann (Hussmann, 1992) the more difficult point working with nondeterministic specifications is the soundness property of the method (or soundness of the Birkhoff theorem). Kaplan gives the theorem (theorem 2.3 in (Kaplan, 1986a)) $MOD_R \models M = N$ iff $\{NF(M)\} = \{NF(N)\}$, although he does not use multialgebra models, and the theorem is stated in terms of equality, instead of inclusions.

The second property $t_2 \in NF^{\supseteq}[f(\ldots, t_1, \ldots)] \Rightarrow \exists t_3 \in NF^{\supseteq}[t_1] \,.\, t_2 \in NF^{\supseteq}[f(\ldots, t_3, \ldots)]$ is named additivity property. It is related with the use of multialgebra models. The functions in these models (from values to sets) can be extended point wise to set arguments (from sets to sets) by the additive property of the functions, obtaining a preorder model. It means that the interpretation mapping I has to be defined inductively by additivity. As we will see, to ensure this property we will require the additivity property for all the functions in the signature. This condition is also required by Hussmann (Hussmann, 1992). In fact, it becomes his DET-*additive* property by translating $t_2 \in NF^{\supseteq}[f(t_1)]$ into $f(t_1) \longrightarrow t_2 \ \wedge \ \mathrm{DET}(t_2)$.

To reduce these four properties to syntactic ones, easier to prove, we need the following lemma.

Lemma 6.13 *Given a specification I containing at least the union axioms, if the orientation and completion of its axioms result in a commutative and terminating bi-rewriting system $\langle R_{\subseteq}, R_{\supseteq} \rangle$, then*

(i) *If $NF^{\supseteq} \subseteq NF^{\subseteq}$, then $I \vdash_{POL} t_1 \supseteq t_2$ implies $NF^{\supseteq}(t_1) \supseteq NF^{\supseteq}(t_2)$.*

(ii) *If $I \vdash_{POL} t \subseteq \bigcup\{t' \mid t \xrightarrow[R_{\supseteq}]{} t'\}$ for any term $t \notin NF^{\supseteq}$, then $NF^{\supseteq}(t_1) \supseteq NF^{\supseteq}(t_2)$ implies $I \vdash_{POL} t_1 \supseteq t_2$.*

(iii) *If in addition the additive property $f(\ldots, X \cup Y, \ldots) = f(\ldots, X, \ldots) \cup f(\ldots, Y, \ldots)$ for any function symbol $f \in \Sigma$ holds in the specification I, and the bi-rewriting system satisfies $NF^{\supseteq}[t_1 \cup t_2] = NF^{\supseteq}[t_1] \cup NF^{\supseteq}[t_2]$ for any pair of terms t_1 and t_2, then $t_2 \in NF^{\supseteq}[f(t_1)]$ implies $\exists t_3 \in NF^{\supseteq}[t_1] \,.\, t_2 \in NF^{\supseteq}[f(t_3)]$.*

Proof:

(i) Let $I \vdash_{POL} t_1 \supseteq t_2$ hold, the commutation and termination properties of $\langle R_{\subseteq}, R_{\supseteq} \rangle$ prove $t_1 \xrightarrow[\supseteq]{*} \circ \xleftarrow[\supseteq]{*} t_2$. Let $t \in NF^{\supseteq}[t_2]$ hold, we have then $t_2 \xrightarrow[\supseteq]{*} t$. The commutation and termination properties prove again $t_1 \xrightarrow[\supseteq]{*} \circ \xleftarrow[\supseteq]{*} t$. However $t \in NF^{\supseteq}$, thus, $t \in NF^{\subseteq}$ by hypothesis, and we have $t_1 \xrightarrow[\supseteq]{*} t$ and therefore $t \in NF^{\supseteq}[t_1]$.

(ii) The termination property and $I \vdash_{POL} t \subseteq \bigcup\{t' \mid t \xrightarrow{\supseteq} t'\}$ allow to prove by noetherian induction $I \vdash_{POL} t \subseteq \bigcup NF^{\supseteq}[t]$. The union axioms prove $I \vdash_{POL} t \supseteq \bigcup NF^{\supseteq}[t]$ and $I \vdash_{POL} \bigcup NF^{\supseteq}[t_1] \supseteq \bigcup NF^{\supseteq}[t_2]$ if $NF^{\supseteq}[t_1] \supseteq NF^{\supseteq}[t_2]$. Therefore, we have by transitivity $I \vdash_{POL} t_1 \supseteq t_2$.

(iii) Using the conditions of the previous point we proved $t_1 = \bigcup NF^{\supseteq}[t_1]$; and by the additional conditions of this point we have $f(\bigcup NF^{\supseteq}[t_1]) = \bigcup_{t_3 \in NF^{\supseteq}[t_1]} f(t_3)$ and $NF^{\supseteq}[\bigcup_{t_3 \in NF^{\supseteq}[t_1]} f(t_3)] = \bigcup_{t_3 \in NF^{\supseteq}[t_1]} NF^{\supseteq}[f(t_3)]$. Therefore, if t_2 belongs to this union of sets, then it belongs to one of them, that is, there exists a term $t_3 \in NF^{\supseteq}[t_1]$ such that $t_2 \in NF^{\supseteq}[f(t_3)]$. ∎

Inspired in this SNF-model we can define a new method for checking inclusions. We name this method *nondeterministic computation* method, NDC-method for short, and we define it as follows.

Definition 6.14 *Given a rewriting system $R_{\supseteq}$ and two terms t and u, the* **NDC-method** *is defined by $NDC(t, u) = true$ if, and only if, $NF^{\supseteq}[t] \subseteq NF^{\supseteq}[u]$.*

Lemma 6.15 *If the conditions $I \vdash_{POL} t \subseteq u$ and $NF^{\supseteq}[t] \subseteq NF^{\supseteq}[u]$ are equivalent, the the NDC-method is sound and complete w.r.t. the class of models $POAlg(I)$.*

The following theorem is the main result of this section, and summarizes the results of all the previous lemmas.

Theorem 6.16 *Given a nondeterministic specification I, and a bi-rewriting system $\langle R_{\subseteq}, R_{\supseteq}\rangle$ resulting from the orientation of its axioms, if the following conditions are satisfied*

(i) *the bi-rewriting system is commutative and terminating,*

(ii) *the axioms defining the union operator can be deduced from I,*

(iii) $NF^{\supseteq} \subseteq NF^{\subseteq}$,

(iv) $I \vdash_{POL} t \subseteq \bigcup\{t' \mid t \xrightarrow{\supseteq} t'\}$ *holds for any term* $t \notin NF^{\supseteq}$,

(v) $I \vdash_{POL} f(\ldots, X \cup Y, \ldots) = f(\ldots, X, \ldots) \cup f(\ldots, Y, \ldots)$ *for any symbol* $f \in \Sigma$

(vi) $NF^{\supseteq}[t_1 \cup t_2] = NF^{\supseteq}[t_1] \cup NF^{\supseteq}[t_2]$ *for any terms t_1 and t_2,*

then the following sentences are equivalent:

$$\begin{array}{lll} POAlg(I) \models t \subseteq u & I \vdash_{POL} t \subseteq u & t \xrightarrow[R_{\subseteq}]{*} \circ \xleftarrow[R_{\supseteq}]{*} u \\ MAlg(I) \models t \subseteq u & & NF^{\supseteq}[t] \subseteq NF^{\supseteq}[u] \end{array}$$

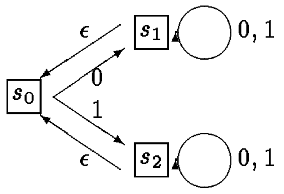
Although these conditions could seem very strange, they hold (or may hold) in most of the nondeterministic specifications. As we will see in the next example, when they do not hold is due to the incompleteness of the specification, the lack of inclusions in $R_{\subseteq}$ without computational meaning, and not to the incompleteness of the rewriting rules $R_{\supseteq}$ used to compute. In these cases it is necessary to add new axioms to the specification, which of course, reduce the number of models, and make the NDC-method and the bi-rewriting method sound and complete.

The same kind of specification completion method has been studied by Hussmann (Hussmann, 1992).

6.4 An Example of Nondeterministic Specification

To show this specification completion method we will use the classical nondeterministic specification of a nondeterministic automata, in this case an automata to recognize the patterns $(0 \cup 1)^* 0 (0 \cup 1)^*$ and $(0 \cup 1)^* 1 (0 \cup 1)^*$. A first attempt to get a specification is shown in figure 6.1 where all inclusions can be oriented to the right, obtaining a commutative bi-rewriting system (where $R_{\subseteq} = \emptyset$).

However, it it easy to see that $trans(s_1, X)$ can be reduced by $\xrightarrow{\supseteq}$ to s_1 or to $trans(s_0, X)$, and $I \vdash_{POL} trans(s_1, X) \subseteq s_1 \cup trans(s_0, X)$ does not hold. Therefore the condition $I \vdash_{POL} t \subseteq \bigcup\{t' \mid t \xrightarrow{\supseteq} t'\}$ does not hold for all reducible terms t. This problem can be avoided adding the axiom $trans(s_1, X) \subseteq s_1 \cup trans(s_0, X)$ to the specification. The same happens with $X \cup X$ that can be



$X \cup Y \supseteq X$	$X \cup Y \supseteq Y$
$trans(s_0, 0) \supseteq s_1$	$trans(s_0, 1) \supseteq s_2$
$trans(s_1, X) \supseteq s_1$	$trans(s_1, X) \supseteq trans(s_0, X)$
$trans(s_2, X) \supseteq s_2$	$trans(s_2, X) \supseteq trans(s_0, X)$
$prog(X, nill) \supseteq X$	
$prog(X, cons(Y, Z)) \supseteq prog(trans(X, Y), Z)$	

Figure 6.1: A nondeterministic automata and its nondeterministic specification.

reduced only to X but $X \cup X \subseteq X$ does not hold; and so on. The additivity condition makes necessary to introduce $trans(X \cup Y, Z) \subseteq trans(X, Z) \cup trans(Y, Z)$ and the same for the second argument and for *prog*. If we complete the specification in this way we obtain the completed specification shown in figure 6.2. This specification can be oriented and completed using the Knuth-Bendix completion process to obtain the bi-rewriting system of figure 6.3. This bi-rewriting system satisfies all the restrictions of the theorem 6.16.

$X \cup Y \supseteq X$	$X \cup Y \supseteq Y$
$X \supseteq X \cup X$	
$trans(s_0, 0) \supseteq s_1$	$trans(s_0, 1) \supseteq s_2$
$trans(s_1, X) = s_1 \cup trans(s_0, X)$	$trans(s_2, X) = s_2 \cup trans(s_0, X)$
$prog(X, nill) = X$	
$prog(X, cons(Y, Z)) = prog(trans(X, Y), Z)$	
$trans(X, Z) \cup trans(Y, Z) \supseteq trans(X \cup Y, Z)$	
$trans(Z, X) \cup trans(Z, Y) \supseteq trans(Z, X \cup Y)$	
$prog(X, Z) \cup prog(Y, Z) \supseteq prog(X \cup Y, Z)$	
$prog(Z, X) \cup prog(Z, Y) \supseteq prog(Z, X \cup Y)$	

Figure 6.2: The completed specification of the automata.

The process described in this example, where a specification is completed –leaving the computational rewriting system $\xrightarrow{\supseteq}$ unchanged– corresponds to the selection of a maximally deterministic model described by Hussmann in (Hussmann, 1992).

6.5 Conclusions

We have shown the usefulness of bi-rewriting systems to relate the mathematical and the operational semantics of nondeterministic specifications. We have given the conditions for the soundness and completeness of a normal form computation procedure and the bi-rewriting method, used to automate the deduction in nondeterministic specifications. We have also given the conditions for the existence

$$R_{\supseteq} = \left\{ \begin{array}{l} X \cup Y \xrightarrow{\supseteq} X \\ X \cup Y \xrightarrow{\supseteq} Y \\ \mathit{trans}(s_0, 0) \xrightarrow{\supseteq} s_1 \\ \mathit{trans}(s_0, 1) \xrightarrow{\supseteq} s_2 \\ \mathit{trans}(s_1, X) \xrightarrow{\supseteq} s_1 \\ \mathit{trans}(s_1, X) \xrightarrow{\supseteq} \mathit{trans}(s_0, X) \\ \mathit{trans}(s_2, X) \xrightarrow{\supseteq} s_2 \\ \mathit{trans}(s_2, X) \xrightarrow{\supseteq} \mathit{trans}(s_0, X) \\ \mathit{prog}(X, \mathit{nill}) \xrightarrow{\supseteq} X \\ \mathit{prog}(X, \mathit{cons}(Y, Z)) \xrightarrow{\supseteq} \mathit{prog}(\mathit{trans}(X, Y), Z) \end{array} \right.$$

$$R_{\subseteq} = \left\{ \begin{array}{l} X \cup X \xrightarrow{\subseteq} X \\ \mathit{trans}(X \cup Y, Z) \xrightarrow{\subseteq} \mathit{trans}(X, Z) \cup \mathit{trans}(Y, Z) \\ \mathit{trans}(Z, X \cup Y) \xrightarrow{\subseteq} \mathit{trans}(Z, X) \cup \mathit{trans}(Z, Y) \\ \mathit{prog}(X \cup Y, Z) \xrightarrow{\subseteq} \mathit{prog}(X, Z) \cup \mathit{prog}(Y, Z) \\ \mathit{prog}(Z, X \cup Y) \xrightarrow{\subseteq} \mathit{prog}(Z, X) \cup \mathit{prog}(Z, Y) \\ \mathit{trans}(X_1, X_2) \cup \mathit{trans}(Y_1, Y_2) \xrightarrow{\subseteq} \mathit{trans}(X_1 \cup Y_1, X_2 \cup Y_2) \\ \mathit{prog}(X_1, X_2) \cup \mathit{prog}(Y_1, Y_2) \xrightarrow{\subseteq} \mathit{prog}(X_1 \cup Y_1, X_2 \cup Y_2) \end{array} \right.$$

(Modulo the associative and commutative axioms for the union)

Figure 6.3: The completed bi-rewriting system automating the deduction in the automata specification.

and initiality of a model based on sets of normal forms.

Chapter 7

Conclusions and Further Work

The course of this thesis has been oriented by one main objective: the definition of the formal basis for the development of a specification methodology based on monotonic inclusion relations. This goal has leaded us to use a wide variety of different formal techniques[1] with a common purpose. Thus, the contributions of the thesis reach different research areas. We would like to distinguish the following ones:

1. The definition of the *Calculus of Refinements* (COR), an extension of the λ-calculus with lattice operations and based on monotonic inclusions. This calculus is proposed as an unified formalism. On the one hand inclusion relations generalize equational relations, and the calculus is an extension of the equational specification formalisms; on the other hand the inclusion relation can be used instead of the typing relation ":", and the calculus can be seen as a typing calculus.

2. The definition of a class of models for the Calculus of Refinements as a restriction of the environment models used for the λ-calculus. This was possible because the operationallity of COR, like the λ-calculus, is based on the β-reduction rule, and the lattice operators are a natural extension not interfering with the semantics of the rest of operators (λ-abstraction and application).

3. Some standard models of the λ-calculus, the D_∞ and the P_ω models, are also models of the Calculus of Refinements. However, in all these models the *computational* ordering (the one providing the lattice structure of the models) and the *structural* ordering (the one used to model the inclusion relation) are identified. In order to distinguish them we have to define a new model.

[1] This, in principle, additional difficulty has allowed us to visit a wide spectrum of theoretical issues and in this way to have now a broad picture of what is going on in computer science.

4. The definition of a new kind of COR-models, the *ideal model*, based on the set of order ideals of a domain. Here, the *structural* ordering used to define the order ideals, and the inverse of the *computational* ordering, used to define continuous functions, are identified. This is the usual intended semantics of type theories. Thus, such model makes the proposal of COR to be a typing formalism. It is a means of interpreting functions as sets, i.e. of interpreting functions as types. We prove that the set of order ideals of a functional domain (where continuous functions can be interpreted as elements of the domain) is also a functional domain.

5. The proposal of a new rewriting technique, the *bi-rewriting systems*, intended to automate the deduction in inclusion theories. This methodology extends the result of term rewriting techniques to the case of non-symmetric relations. Main properties of rewriting systems are kept, and we point where difficulties turn up.

6. The definition of a new concept of higher-order rewriting systems based on a restricted second-order typed language, and the description of a unification procedure for this language. This definition is an alternative to the existing definitions of higher-order rewriting systems, and solves the *extended critical pairs* problem of first-order bi-rewriting systems.

7. A contribution to the automatic verification of non-deterministic specifications, based on the use of bi-rewriting systems. Nondeterministic computation can be modeled by means of an inclusion relation as follows. If a can be evaluated nondeterministically to b then $a \supseteq b$. However, the nondeterministic computation relation fulfills more properties than the inclusion relation. For instance if t only can be evaluated to a or b, then $f(t)$ only can be evaluated to $f(a)$ or $f(b)$. We prove that if we complete a nondeterministic specifications with these additional properties not shared by the inclusion relation, then the bi-rewriting technique is a sound and complete method to test the verification of these specifications.

7.1 Further Work and Open Problems

Some problems appeared during this research have a still pending solution. We want to emphasize the following ones:

1. The description of application areas and the practical use of this methodology, as well as the creation (implementation) of a specification language. The examples we have studied (not described in this thesis) suggested us the use of the Calculus of Refinements as a typing formalism where the inclusion relation substitutes the typing relation. The ideal model formalizes this idea, however the utility of such bold proposal has still to be motivated.

2. The statement of an initiality property for the *ideal model.* The initiality property of a semantic domain is important in order to ensure the validity of induction principles. The category-theoretic solution of recursive domain equations ensures that this property holds for the resulting solution. We use this category-theoretic technique to obtain a value domain, and we prove that the set of order ideals of such domain satisfies a type recursive equation, however the initiality of this ideal domain is not proved. We think that the ideal domain is the initial solution of a pair of recursive equations, one is the type recursive equation, and the other is a still not determined equation ensuring that the domain has enough different elements.

3. The decidability of the linear second-order typed λ-calculus unification problem. On the one hand, we know that the general second-order unification problem is semi-decidable, on the other, we know that the string unification problem is decidable. The linear second-order unification problem is just between both of them, and it has still not been proved if it is decidable or not. We have an unfinished proof of the decidability of such problem. However, the proof is not easy and has to be finished before stating such important result.

4. The termination problem in second-order bi-rewriting systems. As far as we know the termination of higher-order rewriting systems as never been studied. In our case, it is also left as a future research line.

References

Agustí, J., Esteva, F., García, P. and Levy, J. (1992). A Calculus of Refinements: its class of models. In *1º Congreso de Programación Declarativa, ProDe'92*, pages 118–126, Madrid, Spain.

Aït-Kaci, H., Podelski, A. and Goldstein, S. C. (1993). Order-sorted feature theory unification. Technical report, Digital Paris Research Laboratory.

Allester, D. M., Givan, B. and Fatima, T. (1989). Taxonomic syntax for first order inference. In *Proc. of the First Int. Conf. on Princ. of Knowledge Representation and Reasoning*, pages 289–300.

Antimirov, V. (1992). Term rewriting in unified algebras: an order-sorted approach. In *9th WADT - 4th Compass Workshop*, Caldes de Malavella, Spain.

Bachmair, L. and Dershowitz, N. (1986a). Commutation, transformation and termination. In Siekmann, J., editor, *8th Conference in Automated Deduction, CADE-8*, volume 230 of *Lecture Notes in Computer Science*, pages 5–20.

Bachmair, L., Dershowitz, N. and Hsiang, J. (1986b). Orderings for equational proofs. In *Proc. Symp. on Logic in Computer Science, LICS'86*, pages 346–357, Boston, Massachusetts.

Bachmair, L. and Dershowitz, N. (1989a). Completion for rewriting module a congruence. *J. of Theoretical Computer Science*, 67:173–201.

Bachmair, L., Dershowitz, N. and Plaisted, D. (1989b). Completion without failure. In Aït-Kaci, H. and Nivat, M., editors, *Resolution of Equations in Algebraic Structures*, volume 2: Rewriting Techniques, chapter 1, pages 1–30. Academic Press, New York.

Bachmair, L. (1991). *Canonical Equational Proofs.* Birkhäuser, Boston, Massachusetts.

Bachmair, L., Ganzinger, H., Lynch, C. and Snyder, W. (1992). Basic paramodulation and superposition. In Kapur, D., editor, *Int. Conference on Automated Deduction CADE'11*, volume 607 of *Lecture Notes in Computer Science*, pages 462–476. Springer-Verlag.

Bachmair, L. and Ganzinger, H. (1993a). Ordered chaining for total orderings. Technical Report MPI-I-93-250, Max-Planck-Institut für Informatik, Saarbrücken, Germany. Short version in Proceedings of CADE'94.

Bachmair, L. and Ganzinger, H. (1993b). Rewrite techniques for transitive relations. Technical Report MPI-I-93-249, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Bachmair, L. and Ganzinger, H. (1993c). Rewrite techniques for transitive relations. Technical Report MPI-I-93-249, Max-Planck-Institut für Informatik, Saarbrücken, Germany. Short version in Proceedings of LICS'94.

Barendregt, H. (1981). *The Lambda Calculus: its syntax and semantics.* Studies in Logic and the Foundations of Mathematics. Elsevier Science Publishers B. V.

Bäumer, H. (1992). On the use of relation algebra in the theory of reduction systems. Technical report, Dept. Informatica, Univ. of Twente, Enschede, The Netherlands.

Bidoit, M., Kreowski, H.-J., Lescanne, P., Orejas, F. and Sannella, D. (1991). *Algebraic System Specification and Development. A Survey and Annotated Bibliography.* Lecture Notes in Computer Science 501, Springer-Verlag, Berlin.

Birkhoff, G. (1935). On the structure of abstract algebras. *Proc. Cambridge Philos. Soc.*, 31:433–454.

Bledsoe, W. and Hines, L. M. (1980). Variable elimination and chaining in a resolution-based prover for inequalities. In Bibel, W. and Kowalski, R., editors, *5th Conference in Automated Deduction, CADE-5*, volume 87 of *Lecture Notes in Computer Science*, pages 70–87, Les Arcs, France. Springer-Verlag.

Bledsoe, W., Kunen, K. and Shostak, R. (1985). Completeness results for inequality provers. *Artificial Intelligence*, 27:255–288.

Cardelli, L. and Wegner, P. (1985). On understanding types, data abstraction and polymorphism. *ACM Computing Surveys*, 17(4):471–522.

Cardelli, L. (1988). A semantics of multiple inheritance. *Information and Computation*, 76:138–164.

Cardelli, L. and Longo, G. (1990). A semantic basis for quest. Technical Report 55, DIGITAL Systems Research Center, Palo Alto, California.

Comon, H. (1993). Completion of rewrite systems with membership constraints. Technical report, CNRS and LRI, Université de Paris Sud.

Constable, R. L., Allen, S. F., Bromley, H. M. and Cleaveland, W. R. (1986). *Implementing Mathematics with the Nuprl Proof Development System.* Series in Computer Science. Prentice-Hall International.

Coquand, T. and Huet, G. (1988). The calculus of constructions. *Information and Computation*, 76:95–120.

Darlington, J. L. (1971). A partial mechanization of second-order logic. *Machine Intelligence*, 6:91–100.

Darlington, J. L. (1973). Automatic program synthesis in second-order logic. In *Proc. of the 3rd Inter. Joint Conf. on Artificial Intelligence*, pages 537–542.

de Kogel, E. (1992). Relational algebra and equational proofs. Technical report, Department of Philosophy, Tilburg University.

Dershowitz, N. and Manna, Z. (1979). Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476.

Dershowitz, N. (1987). Termination of rewriting. *J. of Symbolic Computation*, 3:69–115.

Dershowitz, N. and Jouannaud, J.-P. (1990). Rewrite systems. In Leeuwen, J. V., editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers.

Ehrig, H., Jiménez, R. M. and Orejas, F. (1991). Compositionality results for different types of parameterization and parameter passing in specification languages. Technical report.

Farmer, W. M. (1988). A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39:131–174.

Fisch, A. M. and Cohn, A. G. (1992). An abstract view of sorted unification. In *11th International Conference on Automated Deduction*. LNCS North-Holland P. C.

Freese, R., Jezek, J. and Nation, J. B. (1993). Term rewrite systems for lattice theory. *J. of Symbolic Computation*, 16:279–288.

Gallier, J. (1985). The semantics of recursive programs with function parameters of finite types: n-rational algebras and logic of inequalities. In Nivat, N. and Reynolds, J., editors, *Algebraic Methods in Semantics*. Cambridge University Press.

Gallier, J. H. and Snyder, W. (1990). Designing unification procedures using transformations: A survey. *Bulletin of the EATCS*, 40:273–326.

Geser, A. (1990). *Relative Termination*. PhD thesis, Universität Passau.

Goguen, J. A. and Meseguer, J. (1992). Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *J. of Theoretical Computer Science*.

Goldfarb, D. (1981). The undecidability of the second order unification problem. *J. of Theoretical Computer Science*, 13:225–230.

Guesarian, I. (1981). *Algebraic Semantics*, volume 99 of *Lecture Notes in Computer Science*. Springer-Verlag.

Gunter, C. A. and Scott, D. S. (1990). Semantic domains. In Leeuwen, J. V., editor, *Handbook of Theoretical Computer Science*, pages 633–674. Elsevier Science Publishers.

Harper, R. (1986). Introduction to Standard ML. Technical Report ECS-LFCS-86-14, LFCS Laboratory for Foundations of Computer Science, Edinburgh.

Hesselink, W. H. (1988). A mathematical approach to nondeterminism in data types. *ACM Trans. Programming Languages and Systems*, 10:87–117.

Hindley, J. R. and Seldin, J. P. (1986). *Introduction to Combinators and λ-Calculus*. London Mathematical Society Student Texts. Cambridge University Press.

Hines, L. M. (1992). Completeness of a prover for dense linear logics. *J. of Automated Reasoning*, 8:45–75.

Hsiang, J. and Dershowitz, N. (1983). Rewrite methods for clausal and nonclausal theorem proving. In *10th Int. Colloquium on Automata, Languages and Programming*, Barcelona, Spain. Springer-Verlag.

Huet, G. (1975). A unification algorithm for typed λ-calculus. *J. of Theoretical Computer Science*, 1:27–57.

Huet, G. (1980). Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821.

Hullot, J.-M. (1980). A catalogue of canonical term rewriting systems. Technical Report CSL-113, Computer Science Laboratory, Menlo Park, California.

Hussmann, H. (1991). *Nondeterministic Algebraic Specifications*. PhD thesis, Institut für Informatik, Technische Universität München, München, Germany.

Hussmann, H. (1992). Nondeterministic algebraic specifications and nonconfluent term rewriting. *Journal of Logic Programming*, 12:237–255.

Jayaraman, B. (1992). Impplementation of subset-equational programs. *J. of Logic Programming*, 12:229–324.

Jensen, D. C. and Pietrzykowski, T. (1976). Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3:123–171.

Jouannaud, J.-P. and Kirchner, H. (1986). Completion on a set of rules modulo a set of equations. *SIAM J. computing*, 15(1):1155–1194.

Kaplan, S. (1986a). Rewriting with a nondeterministic choice operator: from algebra to proofs. In *Proc. 1986 European Symp. on Programming*, volume 213 of *Lecture Notes in Computer Science*, pages 351–374. Springer.

Kaplan, S. (1986b). Simplifying conditional term rewriting systems: Unification, termination and confluence. Technical Report 316, Laboratoire de Recherche en Informatique, Universite de Paris-Sud, Orsay, France.

Kaplan, S. (1988). Rewriting with a nondeterministic choice operator. *J. of Theoretical Computer Science*, 56:37–57.

Kirchner, C. (1985a). *Methodes et Outils de Conception Systematique d'Algorithmes d'Unification dans les Theories Equationnelles.* PhD thesis, Universite de Nancy I.

Kirchner, H. (1985b). *Preuves par Completion dans les Varietes d'Algebres.* PhD thesis, Universite de Nancy I.

Klop, J. W. (1987). Term rewriting systems: A tutorial. *Bulletin of the EATCS*, 32:143–183.

Knuth, D. E. and Bendix, P. B. (1970). Simple word problems in universal algebras. In Leech, J., editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Elmsford, N. Y.

Koymans, C. P. J. (1982). Models of tha lambda calculus. *Information and Control*, 52:306–332.

Lampson, B. and Burstall, R. M. (1988). Pebble, a kernel language for modules and abstract data types. *Information and Computation*, 76:278–346.

Landin, P. (1964). The next 700 programming languages. *Comm. ACM*, 9:157–166.

Levy, J., Agustí, J., Esteva, F. and García, P. (1990). COR: A calculus of refinements. Technical Report GRIAL-90-19, Centre d'Estudis Avançats de Blanes, Blanes, Spain.

Levy, J., Agustí, J., Esteva, F. and García, P. (1991). An ideal model for an extended λ-calculus with refinements. Technical Report ECS-LFCS-91-188, Laboratory for Foundations of Computer Science, Edinburgh, Great Britain.

Levy, J. and Agustí, J. (1992a). Implementing inequality and nondeterministic specifications with bi-rewriting systems. In Ehrig, H. and Orejas, F., editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 252–267, Caldes de Malavella, Spain. Springer-Verlag.

Levy, J. and Agustí, J. (1992b). Proving confluence without termination. Technical Report IIIA-92-27, Institut d'Investigació en Intel·ligència Artificial, Blanes, Spain.

Levy, J., Agustí, J. and Mañá, F. (1992c). Functional lattices for taxonomic reasoning. Technical report, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Great Britain.

Levy, J. (1993a). A higher-order unification algorithm for bi-rewriting systems. In *2º Congreso de Programación Declarativa, ProDe'93*, Blanes, Spain. Also in *2nd CCL workshop*, L'Escala, Spain, 1993.

Levy, J. (1993b). Second-order bi-rewriting systems. Technical Report IIIA-93-11, Institut d'Investigació en Intel·ligència Artificial, Blanes, Spain.

Levy, J. and Agustí, J. (1993c). Bi-rewriting, a term rewriting technique for monotonic order relations. In Kirchner, C., editor, *Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 17–31, Montreal, Canada. Springer-Verlag.

MacQueen, D., Plotkin, G. D. and Sethi, R. (1986). An ideal model for recursive polymorphic types. *Information and Control*, 71:95–130.

Makanin, G. S. (1977). The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198.

Mañá, F., Agustí, J., García, P. and Levy, J. (1992). Técnicas de reescritura en retículos funcionales. In *1º Congreso de Programación Declarativa, ProDe'92*, pages 165–172, Madrid, Spain.

Manca, V., Salibra, A. and Scollo, G. (1990). Equational type logic. *J. of Theoretical Computer Science*, 77:131–159.

Manna, Z. and Waldinger, R. (1986). Special relations in automated deduction. *J. of the ACM*, 33:1–60.

Manna, Z. and Waldinger, R. (1992). The special-relation rules are incomplete. In Kapur, D., editor, *11th Int. Conf. on Automated Deduction, CADE-11*, volume 607 of *Lecture Notes in Artificial Intelligence*, Saratoga Springs, New York. Springer-Verlag.

Martí-Oliet, N. and Meseguer, J. (1993). Rewriting logic as a logical and semantic framework. Technical Report SRI-CSL-93-05, SRI International, Menlo Park, California.

Martin-Löf, P. (1979). Constructive mathematics and computer programming. In *Proc. of the sixth International Congress for Logic, Methodology and Philosophy of Science*. North Holland.

Meseguer, J. (1990). Rewriting as a unified model of concurrency. In *Concur'90*, Lecture Notes in Computer Science, Amsterdam, The Netherlands. Springer-Verlag.

Meseguer, J. (1992). Conditional rewriting logic as a unified model of concurrency. *J. of Theoretical Computer Science*, 96:73–155.

Meseguer, J. (1993). A logical theory of concurrent objects and its realization in the maude language. In Agha, G., Wegner, P. and Yoneyawa, A., editors, *Re-*

search Directions in Concurrent Object-Oriented Programming. The MIT Press. (Also as technical report SRI-CSL-92-08R).

Meyer, A. R. (1982). What is a model of the lambda calculus? *Information and Control*, 52:87–122.

Miller, D. (1990). Abstractions in logic programs. In Odifreddi, P., editor, *Logic and Computer Science*, volume 31 of *APIC Studies in Data Processing*, pages 329–359. Academic Press.

Miller, D. (1991a). A logic programming language with lambda-abstraction, function variables, and simple unification. Technical Report ECS-LFCS-91-159, Laboratory for Foundations of Computer Science, Edinburgh, Great Britain.

Miller, D. (1991b). Unification of simply typed lambda-terms as logic programming. Technical Report ECS-LFCS-91-160, Laboratory for Foundations of Computer Science, Edinburgh, Great Britain.

Milner, R. (1978). A theory of type polymorphism in programming. *J. of Computer System Science*, 17(3):348–375.

Milner, R., Tofte, M. and Harper, R. (1990). *The definition of Standard ML.* MIT Press.

Mitchell, J. C. (1988). Polymorphic type inference and containment. *Information and Control*, 76:211–249.

Moreno-Navarro, J. J. and Rodríguez-Artalejo, M. (1992). Logic programming with functions and predicates: The language BABEL. *J. of Logic Programming*, 12:189–223.

Mosses, P. D. (1989a). Unified algebras and action semantics. In *Proceedings of the 6th Ann. Symp. on Theoretical Aspects of Computer Science, STACS'89*, volume 349 of *Lecture Notes in Computer Science*, pages 17–35. Springer-Verlag.

Mosses, P. D. (1989b). Unified algebras and institutions. In *Proceedings of the 4th IEEE Symp. on Logic in Computer Science, LICS'89*, pages 304–312.

Mosses, P. D. (1989c). Unified algebras and modules. In *Proceedings of the 16th ACM Symp. on Principles of Programming Languages, POPL'89*, pages 329–343.

Mosses, P. D. (1990). Denotational semantics. In Leeuwen, J. V., editor, *Handbook of Theoretical Computer Science*, pages 575–632. Elsevier Science Publishers.

Mosses, P. D. (1992). *Action Semantics*, volume 26 of *Cambridge Tracks in Theoretical Computer Science.* Cambridge University Press.

Nebel, B. (1990). *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence. Springer-Verlag.

Newman, M. H. A. (1942). On theories with a combinatorial definition of "equivalence". *Annals of Math.*, 43(2):223–243.

Nieuwenhuis, R. and Rubio, A. (1992). Basic superposition is complete. In *ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer-Verlag.

Nipkow, T. (1986). Nondeterministic data types: Models and implementations. *Acta Informatica*, 22:629–661.

Nipkow, T. (1991). Higher-order critical pairs. In *Proc. 6th IEEE Symp. Logic in Computer Science*, pages 342–349.

Nipkow, T. (1992). Functional unification of higher-order patterns. Technical report, Institut für Informatik, TU München.

Nipkow, T. (1993). Orthogonal higher-order rewrite systems are confluent. In *Typed Lambda Calculi and Applications.*

O'Donnell, M. J. (1987). Term-rewriting implementation of equational logic programming. In Lescanne, P., editor, *Proc. of Rewriting Techniques and Applications*, pages 1–12, Bordeaux, France. Springer-Verlag.

Orejas, F. (1987). A characterization of passing compatibility for parameterized specifications. *J. of Theoretical Computer Science*, 51:205–214.

Paulson, L. C. (1987). *Logic and Computation: Interactive Proof with Cambridge LCF.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, Great Britain.

Peterson, G. E. and Stickel, M. E. (1981). Complete sets of reductions for some equational theories. *Journal of the ACM*, 28(2):233–264.

Pierce, B. C. (1991). *Basic Category Theory for Computer Scientists.* The MIT Press, Cambridge, Massachusetts.

Pietrzykowski, T. (1973). A complete mechanization of second-order type theory. *J. of the ACM*, 20:333–364.

Plotkin, G. D. (1976). A powerdomain construction. *SIAM J. of Computing*, 5:452–487.

Plotkin, G. (1983). *Domains.* Department of Computer Science, University of Edinburgh. (Course Notes edited by Y. Kashiwagi and H. Kondoh).

Prehofer, C. (1995). *Solving Higher-Order Equations: From Logic to Programming.* PhD thesis, Technische Universität München.

Reynolds, J. C. (1985). *Three Approaches to Type Structure.* Springer-Verlag.

Robertson, D., Agustí, J., Hesketh, J. and Levy, J. (1993). Expressing program requirements using refinement lattices. In Komorowski, J. and Raś, Z. W., editors, *Metodologies for Intelligent Systems*, volume 689 of *Lecture Notes in Artificial Intelligence*, pages 245–254, Trondheim, Norway. Springer-Verlag. (To be published in *Fundamenta Informatica.*).

Robinson, G. A. and Wos, L. T. (1969). Paramodulation and theorem proving in first order theories with equality. *Machine Intelligence*, 4:133–150.

Sanchis, L. E. (1980). Reflexive domains. In Seldin, J. P. and Hindley, J. R., editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press.

Sannella, D. and Tarlecki, A. (1984). Program specification and development in Standard ML. *ACM*, 4:67–77.

Sannella, D., Sokolowski, S. and Tarlecki, A. (1990). Toward formal development of programs from algebraic specifications: parameterisation revisited. (Draft).

Sannella, D. and Tarlecki, A. (1991a). Extended ML: Past, present and future. In *Proc. 7 th Workshop on Specification of Abstract Data Types*, Wusterhausen, GDR. Springer-Verlag.

Sannella, D. and Tarlecki, A. (1991b). A kernel specification formalism with higher-order parameterisation. In *Proc. 7 th Workshop on Specification of Abstract Data Types*, Wusterhausen, GDR. Springer-Verlag.

Schmidt, D. A. (1988). *Denotational Semantics: A Methodology for Language Development.* Wm. C. Brown Publishers, Dubuque, Iowa.

Schulz, K. U. (1991). Makanin's algorithm, two improvements and a generalization. Technical Report CIS-Bericht-91-39, Centrum für Informations und Sprachverarbeitung, Universität München.

Scott, D. (1972). Continuous lattices. In Lawvere, F. W., editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer-Verlag.

Scott, D. (1976). Data types as lattices. *SIAM Journal on Computing*, 5(3):522–587.

Siekmann, J. H. (1989). Unification theory. *J. of Symbolic Computation*, 7:207–274.

Slagle, J. R. (1972). Automatic theorem proving for theories with built-in theories including equality, partial orderings, and sets. *J. of the ACM*, 19:120–135.

Smolka, G. and Aït-Kaci, H. (1989). Inheritance hierarchies: Semantics and unification. *Journal of Symbolic Computation*, 7:343–370.

Smyth, M. (1978). Power domains. *J. of Computer System Science*, 16:23–36.

Smyth, M. B. and Plotkin, G. D. (1982). Category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11:761–783.

Spivey, J. M. (1988). *Understanding Z: a specification language and its formal semantics.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Stoy, J. E. (1978). *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* MIT Press, Cambridge, Massachusetts.

Toyama, Y. (1987). On the Church-Roser property for the direct sum of term rewriting systems. *J. of the ACM*, 34(1):128–143.

von Wright, J. (1990). *A Lattice-theoretical Basis for Program Refinement.* PhD thesis, Åbo Akademi.

Wand, M. (1979). Fixed-point constructions in order-enriched categories. *J. of Theoretical Computer Science*, 8:13–30.

Index