UNIVERSITÀ DEGLI STUDI DI PISA
DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA
Università di Pisa-Udine

Ph.D. Thesis: TD-6/98

Themes in Final Semantics

Marina Lenisa

Abstract. The broad objective of this thesis is to give evidence to the fact that *Final Semantics*, originated and developed by Aczel and Rutten, is a quite general methodology for semantics. We show in fact that it can apply equally fruitfully to *process algebra* languages, *higher order imperative concurrent* languages, *functional* languages, and *calculi for mobile processes.* But furthermore, we show that Final Semantics can be successfully carried out *already* in naive set-theoretical categories, without having to resort to complex mathematical settings. Hence this thesis can be viewed also as an investigation in the applicability of *hypersets. Final Semantics* is a principled way of understanding, and hence manipulating and reasoning rigorously on, those infinite and circular objects and those recursively defined notions, which arise in computation theory through some kind of *maximal fixed point* definition. In this thesis we carry out investigations of maximal fixed points from *set-theoretical*, *categorical*, and *logical* standpoints. In particular, we discuss the purely set-theoretical account of *coinduction principles* and *coiterative functions.* We give a categorical generalization of this account in terms of *final coalgebras*, which encompasses a great deal of the diversity of coinduction schemes. We discuss also logical axiomatizations of largest bisimulation equivalences on syntactical objects. We develop both syntactical and semantical techniques for giving coinductive descriptions of observational (contextual) equivalences, thereby providing a number of (often new) coinduction principles for reasoning on program and process behaviour.

March 1998

Address: Corso Italia 40, 56125 PISA, Italy. Tel:+39-50-887000 - Fax: 887226
E-mail: lenisa@dimi.uniud.it

Author's address:

Marina Lenisa
Dipartimento di Matematica e Informatica
Università degli Studi di Udine
Via delle Scienze, 206 (loc. Rizzi)
33100 Udine (Italy)
E-mail: lenisa@dimi.uniud.it

C'era una volta un re
seduto in canapè,
che disse alla regina
raccontami una storia.
La regina cominciò:
"C'era una volta un re
seduto in canapè
che disse..."

Contents

List of Tables

Acknowledgements

This thesis would not have been possible, if I would not have had the fortunate opportunity to be part of the wonderfully stimulating community of the European Science Project MASK (Mathematical Structures for Concurrency) 1992-1996. This community, led by Jan Rutten, was composed by researchers from six different institutions: CWI, IRISA Rennes, and the Universities of Pisa, Udine, Mannheim, and Koblenz. Jan Rutten, Philippe Darondeau, Ugo Montanari, Furio Honsell, Mila Maister-Cederbaum, Lutz Priese were the site leaders. To them, and to all the other members of the project goes my deepest gratitude for the many discussions, which have contributed so much to my understanding of the topics of this thesis. Special gratitude for illuminating and fruitful discussions goes to Jaco de Bakker, Franck van Breugel, Ennio De Giorgi, Mariangiola Dezani Ciancaglini, Gian Luigi Ferrari, Marco Forti, Ian Mason, Marco Pistore, Andrew Pitts, Jan Rutten, Daniele Turi.

The members of my advisory committee of the *Dottorato* of the Pisa-Udine consortium: Pierpaolo Degano, Giorgio Ghelli, Ugo Montanari have greatly helped me with their inspiring suggestions, recommendations and indications during the two years period of preparation of this thesis. I am particularly indebted to my advisors Furio Honsell and Ugo Montanari for their guidance.

Special thanks go to my external referees Peter Aczel and Jan Rutten for their careful readings of a first draft of this thesis, and for their valuable comments and suggestions.

Chapter 1

Introduction

Both in the practice and in the theory of computation there are many situations were we have to define or reason about *potentially infinite* and *circular* objects or concepts. Consider for example *higher order objects*, e.g. implicitly defined functions, *lazy data types* such as streams, exact reals, processes, etc., *observational* (*operational*, *behavioural*) equivalences.

Structural induction trivially fails on infinite and non-wellfounded objects. It can be applied only in rather contrived ways, and always indirectly, often utilizing inefficient implementations of these objects, e.g. streams as inductively defined functions on natural numbers. Elaborate mathematical theories and structures have been invented in order to support rigorous treatment of such objects, e.g. *domain theory* ([GS90, Plo85]) and *metric semantics* ([BZ82, AR89]).

In recent years, considerable energy has been devoted in the scientific community towards the development of simple principles and techniques for understanding, defining and reasoning on infinite and circular objects. The ideal framework should allow to deal with infinite computational objects in a *natural*, *operationally based*, *implementation-independent* way, without requiring an heavy mathematical overhead. *Coinductive definitions* and *coinduction proof principles* appear to be a rather natural and satisfactory candidate, see e.g. [BM96, Coq94, Fio96, Gor94, Jac97, RP95, Gor95, Pit95, Pit96a, Rut96].

Techniques based on coinduction are natural, in that infinite and circular objects and concepts often arise in connection with a *maximal fixed point* construction of some kind.

In the context of the semantics of programming languages, a possible uniform methodology for introducing and explaining techniques based on coinduction has taken the name of *Final Semantics*, after the seminal works of Aczel ([Acz88, Acz93]), Rutten and Turi ([RT93]).

The gist of Final Semantics is to view the *interpretation function* from *syntax* to *semantics* as a *final arrow* in a suitable category. To this end the semantics has to be construed as a *terminal coalgebra* for a suitable functor F, and the syntax has to be cast into the form of an F-coalgebra. In a sense, this approach is driven by the *operational* semantics of the language, because it is the operational semantics which determines the structure of the functor F. This is dual to the syntax-driven approach of *initial semantics*. Here, it is syntax which determines the form of the functor F, in such a way that the syntax can be construed as an initial F-algebra, and it is the semantics which now, has to be cast in the form of an F-algebra. As Rutten and Turi ([RT94, Rut96]) have clearly illustrated, this duality can be pushed very far and it is rather fruitful. By way of example we just mention a few intriguing correspondences: *congruences* are dual to *bisimulations*, so that *induction* (i.e. identity is the least congruence) is dual to *coinduction* (i.e. identity is the largest bisimulation); *inductively defined* functions (i.e. unique arrows out of the *initial algebra*) are dual to *coiterative* functions (i.e. unique maps into the *final coalgebra*).

More specifically, the paradigm of Final Semantics consists in representing the operational semantics of a programming language as a *syntactical F-coalgebra*, for a suitable functor F, in such a way that the appropriate behavioural equivalence (e.g. bisimilarity) can be characterized as the equivalence induced on terms by the unique morphism from the syntactical coalgebra to the final one. Final Semantics is a fruitful way of understanding the correspondences between syntax and operational semantics. First of all, it naturally validates, and sometimes suggests, *coinductive* proof techniques for establishing behavioural equivalences, as well as other properties. It provides also a natural way of justifying definitions of semantical operators by corecursion. But furthermore, it allows to phrase proofs by coinduction, in a more principled and uniform way. At a more conceptual level, it provides also a unifying view of certain constructions. The definition of functions by coiteration and proofs by coinduction appear as two aspects of the same phenomenon.

The broad objective of this thesis is to give evidence to the thesis that the final semantics methodology, as developed by Aczel, Rutten, Turi, Jacobs, Honsell, and also myself ([RT93, TJ93, RT94, Rut96, RV97, HL95, Len96, Jac97]), is a quite general methodology for semantics. We will show that it can apply equally fruitfully to concurrent as well as functional languages. But more than this, we want to show that Final Semantics can be successfully carried out *already* in naive set-theoretical categories, without having to resort to complex mathematical settings. This thesis can be viewed also as an investigation in the applicability of *hypersets*, i.e. non-wellfounded sets belonging to a universe satisfying the antifoundation axiom X_1 of Forti and Honsell [FH83] (Aczel's AFA of [Acz88]). In this sense, the work in this thesis goes in the line of ([Bar91, BM96]). Particularly interesting in this respect are the connections that we draw with metric semantics techniques in Chapter 6, Section 6.3.5.

Particularly important in achieving our objective, is to show (Chapter 3, Section 3.3) that Final Semantics can capture in a principled way a great deal of the diversity of coinduction schemes and techniques which arise in practice, and not just the basic ones.

Of course Final Semantics is, ultimately, just one of the various possible categorical reformulations ([HJ95, JNW94, Pit96a]), albeit a very sharp one, of coinduction proof principles and coinductive definition schemes. Often, the real technically difficult issues arise in connection with showing that coinductive characterizations are possible. In this respect, we shall introduce and investigate various techniques for showing that observational congruences in programming languages (especially λ-calculus and higher order process algebras) allow coinductive characterizations (see Chapter 6, Sections 6.3.4, Chapter 7, Section 7.5).

Synopsis of the thesis This thesis consists of two parts, an introduction and a conclusion.

In Part I we present various contributions towards a general theory of coinductive techniques. This part consists of three chapters, each one of them devoted to a different theoretical standpoint: set-theoretical, categorical and logical.

In Chapter 2, Section 2.1 we discuss maximal fixed points of operators on complete lattices. We illustrate the diversity of coinduction schemes, and we determine abstract properties of operators which imply soundness and completeness of such schemes. In Section 2.2 we discuss two paradigm examples of coinductive types: hypersets and streams. We recall some basic results on hypersets and we study the Antifoundation Axiom X_1 from the point of view of cardinality. Streams are used as a running example in order to present the connections between coinduction principles and the mechanism of coiterative and coinductive definitions.

In Chapter 3, Section 3.1 we study categorical properties of F-coalgebras, and F-bisimulations in general categories. In Section 3.2, the theory developed in the previous section is specialized to set-theoretical categories. In Section 3.3 we provide categorical counterparts to some of the diverse coinduction schemes which arise set-theoretically, such as bisimulation up-to. We analyze categorically also corecursion. In Section 3.4 we investigate connections between categorical coinduction and the purely set-theoretical coinduction. In Section 3.5 we discuss existence of final coalgebras in various categories. We give a sharpening of Aczel's Special Final Coalgebra Theorem in the way of cardinality.

In Chapter 4, Section 4.1 we present a coinductive axiomatization of the bisimulation equivalence on regular binary trees. In Section 4.2 we do the same for regular non-deterministic processes.

In Part II we develop four extensive case studies in final semantics and in coinductive characterizations of observational equivalences. This part consists of four chapters. The first chapter is devoted to the study of a wide spectrum of branching and linear equivalences on process algebras. The second chapter is devoted to higher order imperative concurrent languages, which normally necessitate of contravariant functors for receiving semantics, and hence which are rather challenging from the point of view of Final Semantics. The third chapter is devoted to untyped λ-calculus, yet another language which necessitates of contravariant functors for semantics. We devote particular attention to the techniques for providing coinductive characterizations of various observational equivalences. The fourth chapter is devoted to a language with an all together problematic semantics: the π-calculus.

In Chapter 5, Section 5.1 we axiomatize in $GSOS$ format strong operational semantics of a generic process algebra language. In Section 5.2 we introduce a general definition of branching bisimulation and we provide final semantics to it. Our general definition comprises weak, progressing and branching bisimulation, and weak congruence. Hypersets are essential to achieve this generality. In Section 5.3 we do the same for linear equivalences. Our general definition in this case subsumes partial (complete) trace equivalence, failure equivalence, etc.. Finally, in Section 5.4, we discuss compositionality of the final semantics with special care for the μ-recursion operator.

In Chapter 6, Section 6.1 we present the syntax of a generic imperative concurrent language with higher order features, and we axiomatize its operational semantics in a sort of $GSOS$ format for languages with global state. In Section 6.2 we provide two alternative final semantics, and we prove formally compositionality of the former. In Section 6.3 we specialize our general theory to the case of a language with higher order assignment and to that of a language with higher order communication. In particular, for the first language, we prove a full abstraction result of the metric denotational semantics with respect to the hyperset-based final semantics.

In Chapter 7, Section 7.1 we introduce basic definitions. In Section 7.2 we introduce two alternative final descriptions of λ-theories, which induce two logically different coinduction principles. The second is new, to the best of our knowledge. In Section 7.3 we present various examples of λ-theories many of which arise from reduction strategies. In Section 7.4 we present computationally adequate models for the λ-theories of the previous section, and we give finitary logical descriptions of them in terms of intersection type assignment systems. In Section 7.5 we introduce three general techniques for giving coinductive characterizations to λ-theories. This is essential for showing the full abstraction of the first final semantics description. The first technique is syntactical, the other two are semantical. In Section 7.7 we discuss the full abstraction of the second final semantics description.

In Chapter 8, Section 8.1 we present the syntax and the early and late operational semantics of the π-calculus. In Section 8.2 we give a presentation of the π-calculus which makes use of higher order syntax. This allows us to focus on the nature of the various binding operators of this language. In Section 8.3 we formalize various notions of bisimulation and congruences for the π-calculus, using our higher order presentation. In Section 8.4, again exploiting our higher order presentation, we discuss final semantics for the π-calculus. Finally, in Section 8.5 we illustrate the advantages of the final semantics formalism for proving process equivalences.

In the conclusion (Chapter 9), we list possible directions for future work. These include solving technical open problems raised in the previous parts, and comparison with alternative approaches to coinduction in the literature.

Finally, in the Appendix we recall some categorical concepts from [FS90].

Part I

Set-theoretical, Categorical, and Logical Foundations

Chapter 2

Set-theoretical Preliminaries

In this chapter we discuss *infinite* and *circular* objects, and *coinduction principles* for reasoning on them, from an elementary purely set-theoretic standpoint. Coinduction principles are dual to *induction* principles, and they arise in computation theory in connection with *coinductive data types*, dual to *inductive data types.* More simply, set-theoretically, coinduction principles arise when an object is viewed as greatest fixed point of a monotone operator.

This chapter is organized as follows. In Section 2.1, we present a list of coinduction principles: almost an inventory. We start from the classical coinduction principle which arises from Tarski's characterization of maximal fixed points, and then we present many variants and generalizations of it, arising from alternative (seemingly sharper) characterizations of maximal fixed points. Our goal is not to produce an exhaustive list, but rather to put traditional coinduction in a broader context of alternative proof principles. We just want to suggest the existence of a theory of coinduction, which probably parallels the dual theory of induction schemes. In Section 2.2, we present two examples of infinite and circular objects: *hypersets and streams.* Both can be viewed as maximal fixed points. Hypersets are particular *non-wellfounded sets*, which are often used for modeling possibly circular *processes* (see Part II of this thesis). We illustrate the *coiteration* and *corecursion* schemata for defining functions into coinductive data types only for the special case example of streams, but everything can be generalized easily to other coinductive data types. The graph of such functions can be viewed as maximal fixed points. The equivalences induced by functions defined by coiteration and by corecursion can be characterized coinductively. These two schemata are dual to the *iteration* and *recursion* schemata, respectively, for defining functions from inductive data types.

2.1 Maximal Fixed Points and Coinduction Principles

In this section we recall Tarski's Fixed Point Theorem, and we give a list of coinduction principles, ranging from the classical coinduction principle, which immediately arises from Tarski's characterization of maximal fixed points, to variants and generalizations of Milner's *bisimulation up-to* ([Mil89]), and to *mixed induction-coinduction principles.* These variants of the classical coinduction principle are apparently more powerful, since they arise from alternative characterizations of maximal fixed points. In this thesis we will see applications of many of the principles introduced in this section.

We will consider *monotone* operators on complete lattices:

Definition 2.1.1 *Let $(X, \leq)$ be a complete lattice. The operator $\Phi : X \to X$ is* monotone *over the complete lattice $(X, \leq)$ if, for all $x, y \in X$,*

$$x \leq y \implies \Phi(x) \leq \Phi(y) \ .$$

Theorem 2.1.2 (Tarski) *Let $(X, \leq)$ be a complete lattice. If $\Phi : X \to X$ is a monotone operator, then Φ has a complete lattice of fixed points, as well as a complete lattice of prefixed points and postfixed points.*

Proof Let $Y \subseteq X$ be a set of fixed points of Φ. One can easily show that

1. If $Z^{\mu}_{Y} = \{x \in X \mid \Phi(x) \leq x \ \wedge \ \forall y \in Y.\ x \geq y\}$, then $\mu_Y = \bigwedge Z^{\mu}_{Y}$ is the least fixed point of Φ, greater or equal to than any element of Y.

2. $Z^{\nu}_{Y} = \{x \in X \mid x \leq \Phi(x) \ \wedge \ \forall y \in Y.\ y \geq x\}$, then $\nu_Y = \bigvee Z^{\nu}_{Y}$ is the greatest fixed point of Φ, less or equal to than any element of Y.

E.g., in order to show 1, one proves that both μ_Y and $\Phi(\mu_Y)$ belong to Z^{μ}_{Y}.
$\Box$

Corollary 2.1.3 *Let $(X, \leq)$ be a complete lattice, and let $\Phi : X \to X$ be a monotone operator. Then*

1. *the least fixed point of Φ is*

$$\mu_{\Phi} = \bigwedge \{x \in X \mid \Phi(x) \leq x\} \ ;$$

2. *the greatest fixed point of Φ is*

$$\nu_{\Phi} = \bigvee \{x \in X \mid x \leq \Phi(x)\} \ .$$

Remark 2.1.4 *The monotonicity of Φ in Theorem 2.1.2 is a sufficient but not necessary condition. In fact the weaker condition of commutativity of Φ with inf (sup) preservation of the sets $\{x \in X \mid \Phi(x) \leq x \ \wedge \ \forall y \in Y.x \geq y\}$ and $\{x \in X \mid x \leq \Phi(x) \ \wedge \ \forall y \in Y.y \geq x\}$ is, obviously, already sufficient.*

From the characterization of the greatest fixed point given by Corollary 2.1.3, we immediately get the following:

Principle 2.1.5 (Coinduction) *Let X be a set, and let $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ be a monotone operator on the complete lattice $(\mathcal{P}(X), \subseteq)$. Then the following principle is* sound*:*

$$\frac{d \in x \quad x \subseteq \Phi(x)}{d \in \nu_\Phi} \ .$$

This principle is also complete in the sense that

$$d \in \nu_\Phi \implies \exists x. \ (d \in x \ \wedge \ x \subseteq \Phi(x)) \ .$$

In the following, we will say that the principle

$$\frac{\Psi_1(x)}{\Psi_2}$$

is *complete* if

$$\Psi_2 \implies \exists x. \ \Psi_1(x) \ .$$

Now we introduce the crucial notion of Φ*-bisimulation*. When specializing the Principle 2.1.5 to maximal fixpoints of relations, the object x which appears in the premise of the Coinduction Principle 2.1.5 is called Φ*-bisimulation*, following [Par81]. It is interesting to point out that, in the context of non-wellfounded set theory, the same notion was independently called *conservative relation* by Forti and Honsell ([FH83]), and *contraction* by Hinnion ([Hin80]).

Definition 2.1.6 *Let X be a set, and let $\Phi : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ be a monotone operator on the complete lattice of relations $(\mathcal{P}(X \times X), \subseteq)$. A* Φ-bisimulation *is a relation $\mathcal{R}$ such that $\mathcal{R} \subseteq \Phi(\mathcal{R})$.*

There are various ways to weaken the characterization of ν_Φ given by Corollary 2.1.3, in order to view ν_Φ as supremum of a larger set of elements. These alternative characterizations yield more powerful coinduction principles.

Theorem 2.1.7 *Let $(X, \leq)$ be a complete lattice, let $\Phi : X \to X$ be a monotone operator, and let $y \in X$. If $y \leq \Phi(y)$, then*

1.

$$\nu_\Phi = \bigvee \{x \mid x \leq \Phi(x) \vee y\} \ .$$

2.

$$\nu_\Phi = \bigvee \{x \mid x \leq \Phi(x \vee y)\} \ .$$

Theorem 2.1.7 gives rise to the following two generalizations of the Coinduction Principle 2.1.5:

Principle 2.1.8 (Coinduction up-to $\cup$) *Let X be a set, let $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ be a monotone operator on the complete lattice $(\mathcal{P}(X), \subseteq)$, and let $y \subseteq X$ be such that $y \subseteq \Phi(y)$. Then the following principles are sound (and complete):*

$$1. \quad \frac{d \in x \quad x \subseteq \Phi(x) \cup y}{d \in \nu_\Phi} \ ;$$

$$2. \quad \frac{d \in x \quad x \subseteq \Phi(x \cup y)}{d \in \nu_\Phi} \ .$$

E.g. y in the coinduction up-to $\cup$ principles above can be taken to be any fixed point of Φ, or, if Φ is defined on a complete lattice of binary relations, and ν_Φ is a reflexive relation, then y can be taken to be the identity relation.

The following theorem gives rise to a coinduction principle which generalizes Milner's bisimulation-up-to principle:

Theorem 2.1.9 *Let $(X, \leq)$ be a complete lattice, let $\Phi : X \to X$ be a monotone operator on the complete lattice $(X, \leq)$, and let $\bullet : X \times X \to X$ be an associative operation. If*

1. *for all x, y, x_1, y_1, $(x \leq \Phi(x_1) \ \wedge \ y \leq \Phi(y_1)) \implies x \bullet y \leq \Phi(x_1 \bullet y_1)$ and*
2. *$\nu_\Phi \leq \nu_\Phi \bullet \nu_\Phi$,*

then

$$\nu_\Phi = \bigvee \{x \mid x \leq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)\} \ .$$

Proof Using Corollary 2.1.3, it is sufficient to prove that

a) $x \leq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi) \implies \exists y.\ x \leq y \leq \Phi(y)$.
b) $x \leq \Phi(x) \implies \exists y.\ x \leq y \leq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)$.

Proof of item a): We prove that $\Phi(\nu_\Phi \bullet x \bullet \nu_\Phi) \leq \Phi(\Phi(\nu_\Phi \bullet x \bullet \nu_\Phi))$. Then we can take $y = \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)$. From $x \leq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)$ and $\nu_\Phi \leq \Phi(\nu_\Phi)$, by item 1, $\nu_\Phi \bullet x \bullet \nu_\Phi \leq \Phi(\nu_\Phi \bullet \nu_\Phi \bullet x \bullet \nu_\Phi \bullet \nu_\Phi)$. From $\nu_\Phi \leq \Phi(\nu_\Phi)$, by item 1, using Corollary 2.1.3, $\nu_\Phi \bullet \nu_\Phi \leq \nu_\Phi$. Hence, by item 2, $\nu_\Phi \bullet \nu_\Phi = \nu_\Phi$ and $\nu_\Phi \bullet x \bullet \nu_\Phi \leq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)$. Finally, by monotonicity of Φ, $\Phi(\nu_\Phi \bullet x \bullet \nu_\Phi) \leq \Phi(\Phi(\nu_\Phi \bullet x \bullet \nu_\Phi))$.

Proof of item b): Since, by the proof of item a), $\nu_\Phi \bullet \nu_\Phi = \nu_\Phi$, then $\nu_\Phi \leq \Phi(\nu_\Phi) \implies \nu_\Phi \leq \Phi(\nu_\Phi \bullet \nu_\Phi \bullet \nu_\Phi)$. Hence take $y = \nu_\Phi$. $\square$

Principle 2.1.10 (Coinduction up-to ν_Φ) *Let X be a set, let $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ be a monotone operator on the complete lattice $(\mathcal{P}(X), \subseteq)$, and let $\bullet : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathcal{P}(X)$ be an associative operation such that*

1. *for all x, y, x_1, y_1, $(x \subseteq \Phi(x_1) \ \wedge \ y \subseteq \Phi(y_1)) \implies x \bullet y \subseteq \Phi(x_1 \bullet y_1)$ and*
2. *$\nu_\Phi \subseteq \nu_\Phi \bullet \nu_\Phi$,*

Then the following principle is sound and complete:

$$\frac{d \in x \quad x \subseteq \Phi(\nu_\Phi \bullet x \bullet \nu_\Phi)}{d \in \nu_\Phi} \ .$$

Remark 2.1.11 *1. If Φ is a monotone operator on a complete lattice of binary relations, $\bullet$ is the relational composition, and ν_Φ is reflexive, then hypothesis 2 of Theorem 2.1.9 is verified.*

2. *Hypothesis 1 in Theorem 2.1.9 can be viewed as a* generalized transitivity. *In fact, If Φ is a monotone operator on a complete lattice of binary relations and $\bullet$ is the relational composition, then hypothesis 1 implies transitivity of the relation ν_Φ.*

3. *Dropping hypothesis 2 in Theorem 2.1.9, and assuming the monotonicity of $\bullet$ w.r.t. $\leq$, i.e., for all x, x_1, y, y_1, $x \leq x_1 \ \wedge \ y \leq y_1 \implies x \bullet y \leq x_1 \bullet y_1$, we get soundness of the coinduction up-to ν_Φ principle, but we loose completeness.*

The following Theorem 2.1.12 guarantees the soundness and completeness of another generalization of Milner's bisimulation-up-to principle. The coinduction principle which arises is related to the proof principle introduced in [San95] for labelled transition systems and *respectful functions*. The argument used for proving the theorem below is inspired to that used in [San95] for proving the soundness of respectful functions.

Theorem 2.1.12 *Let $(X, \leq)$ be a complete lattice and let $\Phi, T : X \to X$ be monotone operators on the complete lattice $(X, \leq)$. If*

1. *T is a closure operator, i.e., for all $x \in X$, $T(x) \geq x$ and*

2. *for all $x \in X$, $(T \circ \Phi)(x) \leq (\Phi \circ T)(x)$,*

then

$$\nu_\Phi = \bigvee \{x \mid x \leq (\Phi \circ T)(x)\} \ .$$

Proof If $x \leq \Phi(x)$, then, since T is a closure operator, $x \leq T(x)$. Using the monotonicity of Φ, $x \leq \Phi(x) \leq \Phi(T(x))$.
Vice versa, if $x \leq \Phi \circ T(x)$, we have to show that $\exists y$ such that $x \leq y \wedge y \leq \Phi(y)$. Consider the following inductively defined sequence $\{x_n\}_{n \geq 0}$:

$$\begin{aligned} x_0 &= x \\ x_{n+1} &= T(x_n) \ . \end{aligned}$$

We prove by induction on n that $x_n \leq \Phi(x_{n+1})$:
For $n = 0$ the thesis is immediate, since $x \leq \Phi \circ T(x)$ by hypothesis.
Let $n > 0$:
$x_n = T(x_{n-1})$
$\leq T \circ \Phi(x_n)$, by induction hypothesis and monotonicity of T,
$= \Phi \circ T(x_n)$, by hypothesis 2,
$= \Phi(x_{n+1})$, by definition of the sequence.
Hence, taking $y = \bigvee_n x_n$, we have $x \leq y$ and $y \leq \Phi(y)$. □

Under the hypotheses of Theorem 2.1.12, the following generalization of Milner's bisimulation-up-to principle is sound and complete:

Principle 2.1.13 (Coinduction up-to T) *Let X be a set, let $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ be a monotone operator on the complete lattice $(\mathcal{P}(X), \subseteq)$, and let $T : \mathcal{P}(X) \to \mathcal{P}(X)$. If*

1. *T is a monotone operator on the complete lattice $(\mathcal{P}(X), \subseteq)$,*
2. *T is a closure operator, i.e., for all $x \in X$, $T(x) \supseteq x$ and*
3. *for all $x \in X$, $(T \circ \Phi)(x) \subseteq (\Phi \circ T)(x)$,*

then the following principle is sound and complete:

$$\frac{d \in x \quad x \subseteq (\Phi \circ T)(x)}{d \in \nu_\Phi} \ .$$

Remark 2.1.14 • *If we drop hypothesis 1) in Theorem 2.1.12, then we can prove soundness, but not completeness of Principle 2.1.13. A simple counterexample is the following. If the operator T is the constant operator equal to the least fixed point of Φ, μ_Φ, and moreover $\mu_\Phi \neq \nu_\Phi$, then $\nu_\Phi \neq \bigvee\{x \mid x \leq (\Phi \circ T)(x)\}$.*

- *The Coinduction Principle 2.1.13 can be viewed as a variant of the principle introduced in [San95]. In particular, the coinduction principle of [San95] is obtained by replacing the hypotheses 1, 2 and 3 in Principle 2.1.13 by the hypothesis of* respectfulness *of T^1. But then, having fixed T, the principle obtained is sound, but* not *complete. In fact, the respectfulness condition is already implied by the hypotheses 1 and 3 of Principle 2.1.13, without assuming that T is a closure operator.*
- *We leave as an open problem to find conditions on Φ and T, such that $\nu_\Phi = \bigvee\{x \mid x \subseteq (T \circ \Phi)(x)\}$.*

Finally, we point out an interesting class of operators, i.e. the *mixed* operators, from which *mixed induction-coinduction* principles arise. Let $\mathcal{R}_1, \mathcal{R}_2 \subseteq X \times X$. We denote by $\mathcal{R}_1 \otimes \mathcal{R}_2 \subseteq X^2 \times X^2$ the following relation

$$\{((x_1, x_2), (x_1', x_2')) \mid (x_1, x_1') \in \mathcal{R}_1 \ \wedge (x_2, x_2') \in \mathcal{R}_2\} \ .$$

For $\mathcal{R} \subseteq X \times X$, and $n > 0$, let $\mathcal{R}^n$ denote the following n-ary relation on X

$$\underbrace{\mathcal{R} \otimes \ldots \otimes \mathcal{R}}_{n} \ .$$

Definition 2.1.15 *Let $(X, \leq)$ be a complete lattice. A* mixed operator *over the complete lattice $(X^{m+n}, \leq^{m+n})$, for $m + n \geq 1$, is an operator $\Phi : X^{m+n} \to X^{m+n}$ monotone in the first m components and antimonotone in the remaining n components.*

[1] $T : X \to X$ is *respectful* if, for all $x, y \in X$, $(x \leq y \wedge x \leq \Phi(y)) \ \Rightarrow \ (T(x) \leq T(y) \wedge T(x) \leq \Phi(T(y)))$.

It is immediate to see that a mixed operator Φ gives rise to a monotone operator $\widetilde{\Phi}$ on the complete lattice $((X^{m+n}, \leq^m \otimes (\leq^{-1})^n)$, obtained by reversing the order on X for the last n components.

For simplicity, in the following theorem we consider a binary operator, but the theorem is obviously generalizable to $(m+n)$-ary operators. It is immediate to see that

Theorem 2.1.16 *Let $(X, \leq)$ be a complete lattice, and let $\Phi : X \times X \to X \times X$ be a mixed operator on the complete lattice $(X \times X, \leq^2)$, monotone in the first component and antimonotone in the second component, such that $sc \circ \Phi \circ sc = \Phi$, where $sc : X \times X \to X \times X$ is the operator which exchanges the componenents of a pair, i.e. $sc(x_1, x_2) = (x_2, x_1)$. If least and greatest fixed point of $\widetilde{\Phi}$ coincide, then $\widetilde{\Phi}$ has a unique fixed point (fix_Φ, fix_Φ).*

Mixed operators give rise to *mixed induction-coinduction principles*:

Principle 2.1.17 (Induction-coinduction) *Let X be a set, let $\Phi : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathcal{P}(X) \times \mathcal{P}(X)$ be a mixed operator on the complete lattice $(\mathcal{P}(X) \times \mathcal{P}(X), \subseteq^2)$, monotone in the first component and antimonotone in the second component, such that $sc \circ \Phi \circ sc = \Phi$. If least and greatest fixed point of $\widetilde{\Phi}$ coincide, then the unique fixed point (fix_Φ, fix_Φ) of $\widetilde{\Phi}$ satisfies the following induction-coinduction principle:*

$$\frac{x^- \subseteq \pi_1(\Phi(x^-, x^+)) \quad x^+ \supseteq \pi_2(\Phi(x^-, x^+))}{x^- \subseteq fix_\Phi \subseteq x^+} \quad .$$

2.2 Examples of Infinite and Circular Objects

In this section we shall present two important examples of *infinite* and *circular* objects: *hypersets* and *streams*. Hypersets are *non-wellfounded sets* belonging to a Universe of a Zermelo Frænkel-like set-theory $ZFC^0_-(\overline{X}_1)$ ($ZFC^0_-(FCU)$). $ZFC^0_-(\overline{X}_1)$ is the theory obtained by replacing the Axiom of Foundation by the Antifoundation Axiom $\overline{X}_1$ of [FH83] (or AFA of [Acz88]). $ZFC^0_-(FCU)$ is the theory obtained by replacing the Axiom of Foundation by the axiom FCU (see Definition 2.2.8), and by weakening the axiom of extensionality so as to allow for a proper class of atoms U. It is easy to see that in both theories the membership relation is non-wellfounded and circular. Non-wellfounded sets will be used extensively to model possibly *circular processes* in Part II of this thesis. This was pioneered by Aczel in [Acz88].

Streams are one of the most elementary examples of a coinductive data type in a functional programming language. We shall use it to exemplify the *coiteration* and *corecursion* schemata for defining functions into coinductive data types. Graphs of functions defined by coiteration and corecursion can be viewed as maximal fixed points. Moreover, such functions induce equivalences which can be characterized coinductively. In particular, we will show that the coinduction principle associated to the corecursion scheme is the coinduction-up-to

ν_Φ. Notice that the functions themselves defined by coiteration and corecursion can be viewed as examples of *circular objects*. We discuss only streams, but all the results can be extended to all coinductive types.

2.2.1 Non-wellfounded Sets

First of all, we recall the Antifoundation Axiom $\overline{X}_1$ of [FH83], which allows for the circularity of the membership relation. The Axiom $\overline{X}_1$ is equivalent to the Axiom AFA of [Acz88]. Then we present a generalization of the Axiom $\overline{X}_1$ to set theories with atoms (Urelementen), i.e. the *Free Construction Principle* FCU introduced in [FHL94]. In this thesis, we will consider the strong version FCU of the Antifoundation Axiom, where a proper class U of atoms is available. Of course atoms can be explained away. But it is conceptually clearer not to take a reductionist attitude and to use atoms when modeling concepts which are not immediately analyzable as sets. In this thesis atoms will be used to model directly atomic actions, without the need to codify atomic actions using sets.

First we work in the set theory $ZFC_0^- + Extensionality$, where the theory ZFC_0^- consists of the axioms *Pairing*, *Union*, *Power Set*, *Replacement*, *Infinity*, *Choice*.

Let V denote the Universe of sets (without atoms).

Definition 2.2.1 **($\overline{X}_1$)** *Let X be a class. For every function $f : X \to \mathcal{P}(X)$,*[2] *there exists a* unique *function $g : X \to V$ which makes the following diagram commute*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "g"'] & \mathcal{P}(X) \arrow[dl, "g^+"] \\ V & \end{tikzcd}$$

where $g^+ : \mathcal{P}(X) \to V$ is defined by $g^+(x) = \{g(y) \mid y \in x\}$.
I.e., for all $x \in X$, $g(x) = \{g(y) \mid y \in f(x)\}$.

It is interesting to point out that $\overline{X}_1$ expresses precisely the fact that the universe V is a *final* coalgebra for the functor $\mathcal{P}(\cdot)$ (see Chapter 3).

The Antifoundation Axiom $\overline{X}_1$ yields immediately a coinductive characterization of equality on sets, i.e. strong extensionality ([FH83, Acz88]):

SExt Two sets x, y are equal if and only if there exists a Φ^+*-bisimulation* $\mathcal{R}$ such that $x \mathrel{\mathcal{R}} y$, where Φ^+ is the following operator on relations of the universe V: $\Phi^+(\mathcal{R}) = \{(x,y) \mid \forall x_1 \in x.\ \exists y_1 \in y.\ x_1 \mathrel{\mathcal{R}} y_1 \wedge \forall y_1 \in y.\ \exists x_1 \in x.\ x_1 \mathrel{\mathcal{R}} y_1\}$.

The notion of Φ^+-bisimulation is called *id-admissible relation* in [FH83].

[2] $\mathcal{P}(X)$ denotes the class of all subsets of X

Proposition 2.2.2

$$ZFC_0^- \ + \ Extensionality \ + \ \overline{X}_1 \implies SExt \ .$$

One can say that the Axiom $\overline{X}_1$ axiomatizes the fact that the Universe V is the richest universe of sets compatible with the axioms of ZFC_0^- and strong extensionality. I.e., using the terminology of [FH83]: "X + Strong Extensionality $\implies \overline{X}_1$".

Remark 2.2.3 *Notice that, in the theory ZFC_0^-, Strong Extensionality is implied by the following alternative formulation of the axiom $\overline{X}_1$:*

Let X be a class. For every function $f : X \to \mathcal{P}(X)$, there is a unique *function $g : X \to V$ such that, for all $x \in X$, $\forall z.\ z \in g(x) \ \Leftrightarrow \ \exists y \in f(x).\ z = g(y)$.*

In the sequel, it will be useful to be able to control the cardinality of the image of X under g. First of all we need the following definitions:

Definition 2.2.4 *Let x be a set.*

- *x is* transitive *if $x \subseteq \mathcal{P}(x)$.*
- *The* transitive closure *of x is defined by*

$$TC(x) = \bigcap\{y \mid y \ transitive \ \wedge \ x \in y\} \ .$$

Let HC_κ denote the set of sets whose hereditary cardinal is less than κ. I.e.:

Definition 2.2.5 *Let HC_κ be defined by:*

$$x \in HC_\kappa \iff |TC(x)| < \kappa \ .$$

Definition 2.2.6 *Let X be a class, let $f : X \to \mathcal{P}(X)$, and let $x \in X$. The* f-transitive closure of *x is the set*

$$TC_f(x) = \bigcap\{y \mid x \in y \ \wedge \ \forall z.\ (z \in y \cap X \ \Rightarrow \ f(z) \subseteq y)\} \ .$$

The following proposition is an easy consequence of the Axiom $\overline{X}_1$:

Proposition 2.2.7 *Assume $\overline{X}_1$. Let X be a class, and let $f : X \to \mathcal{P}(X)$. If, for all $y \in X$, $|TC_f(y)| < \kappa$, then the image of X under g, $g^+(X)$, is such that $g^+(X) \in \mathcal{P}(HC_\kappa)$, where g is the unique function given by the Axiom $\overline{X}_1$.*

We work now in the set theory $ZFC_0^-(U)$ + *Extensionality up-to* U, $ZFC_0^-(U)$ consisting of the axioms of ZFC_0^- with an extra axiom which states that the elements of U are atoms and not sets, i.e., for all $u \in U$, $\forall z.\ z \notin u$. The axiom of *Extensionality up-to* U is defined as follows

Extensionality up-to U For all $x, y \notin U$, $\forall z.\ (z \in x \ \Leftrightarrow \ z \in y) \implies x = y$.

Let V_U denote the universe of sets with atoms in U, in the theory $ZFC_0^-(U)$ + *Extensionality up-to U*.

Now we introduce the *Free Construction Principle FCU*. This axiom is a version of the Axiom $\overline{X}_1$ up-to a class U of Urelementen (atoms).

Definition 2.2.8 (*FCU*) *Let U be a class of Urelementen. Let X be a class such that $X \cap U = \emptyset$. For every function $f : X \to \mathcal{P}(X \cup U)$, there is a* unique *function $g : X \to V_U$ which makes the following diagram commute*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "g"'] & \mathcal{P}(X \cup U) \arrow[dl, "(g \cup id_{U \setminus X})^+"] \\ V_U & \end{tikzcd}$$

I.e., for all $x \in X$, $g(x) = (f(x) \cap (U \setminus X)) \cup \{g(y) \mid y \in f(x) \cap X\}$.

In Chapter 3, we will see that the axiom FCU is equivalent to the fact that the universe of sets V_U is a *final* coalgebra for the functor $\mathcal{P}(\cdot \cup U)$.

The Antifoundation Axiom FCU yields the axiom of *Strong Extensionality up-to the class of atoms U*:

SExt up-to U Two sets x, y are equal if and only if there exists a Φ_U^+-*bisimulation* $\mathcal{R}$ such that $x \mathrel{\mathcal{R}} y$, where Φ_U^+ is the following operator on relations of the universe of sets V_U:
$\Phi_U^+(\mathcal{R}) = \{(x, y) \mid x \cap U = y \cap U \ \wedge$
$\forall x_1 \in (x \setminus U).\ \exists y_1 \in (y \setminus U).\ x_1 \mathrel{\mathcal{R}} y_1 \ \wedge\ \forall y_1 \in (y \setminus U).\ \exists x_1 \in (x \setminus U).\ x_1 \mathrel{\mathcal{R}} y_1\}$.

Proposition 2.2.9

$$ZFC_0^-(U) \ + \ \textit{Extensionality up-to } U \ + \ FCU \Longrightarrow \ \textit{SExt up-to } U \ .$$

Proposition 2.2.11 below is the version of Proposition 2.2.7 for the Axiom FCU. First, we need to generalize to a universe with atoms the definitions of transitive set, transitive closure of a set, f-transitive closure of a set, and the definition of the set of sets whose hereditarily cardinal is less than κ.

Definition 2.2.10

- *A set $x \in V_U$ is* transitive *if $x \subseteq \mathcal{P}(x) \cup U$.*
- *The* transitive closure *of x is defined by*

$$TC(x) = \bigcap \{y \mid y \ \textit{transitive} \ \wedge \ x \in y\} \ .$$

- *Let $HC_\kappa(U)$ be defined by:*

$$x \in HC_\kappa(U) \iff |TC(x)| < \kappa \ .$$

- *Let X be a class, let $f : X \to \mathcal{P}(X \cup U)$, and let $x \in X$. We define*

$$TC_f(x) = \bigcap \{y \mid x \in y \ \wedge \ \forall z.\ (z \in y \cap X \ \Rightarrow \ f(z) \subseteq y)\} \ .$$

Proposition 2.2.11 *Assume FCU. Let X be a class, and let $f : X \to \mathcal{P}(X \cup U)$. If, for all $y \in X$, $|TC_f(f(y))| < \kappa$, then the image of X under g, $g^+(X)$, is such that $g^+(X) \in \mathcal{P}(HC_\kappa(U))$, where g is the unique function given by the Axiom FCU.*

Finally, it is interesting to point out that, assuming FCU, the following axiom holds in the set-theoretic structure $< \mathcal{P}(HC_\kappa(U)), \in>$.

Proposition 2.2.12 (FCU_κ) *Let U be a class of Urelementen. Assume FCU. Let $X \subseteq HC_\kappa(U)$. For every function $f : X \to \mathcal{P}_{<\kappa}(X \cup U)$ such that $\forall x \in X.\ |TC_f(\{x\})| < \kappa$, there exists a* unique *function $g : X \to \mathcal{P}(HC_\kappa(U))$ which makes the following diagram commute*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "g"'] & \mathcal{P}_{<\kappa}(X \cup U) \arrow[dl, "(g \cup id_U)^+"] \\ \mathcal{P}(HC_\kappa(U)) & \end{tikzcd}$$

2.2.2 Coiteration and Corecursion on Streams

In this section we consider the coinductive data type of *infinite streams* on natural numbers, $Stream_{\mathbf{N}}$, and we present the schemata of *coiteration* and *corecursion* for defining functions into $Stream_{\mathbf{N}}$. Moreover, we study the coinduction principles arising from these schemata. In particular, the coinduction principle up-to ν_Φ of Section 2.1 arises in connection with the corecursion scheme. In Chapter 3, we will see that the coiteration and the corecursion schemata are particular instances of the general schemata of *categorical coiteration* and *categorical corecursion*, respectively, when $Stream_{\mathbf{N}}$ is viewed as the *final* F-coalgebra for a suitable functor F.

First of all, we introduce the coinductive data type of infinite streams on natural numbers. Set-theoretically, coinductive data types are viewed as maximal fixed points of monotone operators. Intuitively, a stream of natural numbers is a set of sets of the form $(n_0, (n_1, \ldots))$, for an infinite sequence $n_0, n_1, \ldots$ of natural numbers. Formally, the set of all streams can be described as a maximal fixed point:

Definition 2.2.13 ($Stream_{\mathbf{N}}$) *Assume that the class of Urelementen contains the set of natural numbers $\mathbf{N}$. Let $Stream_{\mathbf{N}}$ denote the greatest fixed point of the following operator $T : V_U \to V_U$:*

$$T(X) = \{(n, x) \mid n \in \mathbf{N} \ \wedge \ x \in X\} \ .$$

Two operations are naturally defined on $Stream_{\mathbf{N}}$:

1. *$head : Stream_{\mathbf{N}} \to \mathbf{N}$ defined by:*

$$head((n, s)) = n \ .$$

2. *$tail : Stream_{\mathbf{N}} \to Stream_{\mathbf{N}}$ defined by:*

$$tail((n, s)) = s \ .$$

It is easy to see that the above definition is well posed, since the operator T is monotone.

Definition 2.2.14 (Coiteration on $Stream_{\mathbf{N}}$) *Let $A \in V_U$, and let $g : A \to \mathbf{N} \times A$. The* coiterative *function induced by g is the* unique *function $f : A \to Stream_{\mathbf{N}}$ which satisfies:*

$$< head, tail > \circ f = (id_{\mathbf{N}} \times f) \circ g \ ,$$

where $< head, tail >: Stream_{\mathbf{N}} \to \mathbf{N} \times Stream_{\mathbf{N}}$ is defined by
$< head, tail > ((n, s)) = (head((n, s)), tail((n, s)))$, and
$(id_{\mathbf{N}} \times f) : \mathbf{N} \times A \to \mathbf{N} \times Stream_{\mathbf{N}}$ is defined by
$(id_{\mathbf{N}} \times f)(n, a) = (id_{\mathbf{N}}(n), f(a))$.
I.e. f is the unique function such that, for all $a \in A$, $f(a) = (\pi_1 \circ g(a), f \circ \pi_2 \circ g(a))$.

The uniqueness of the function f can be easily proved using strong extensionality. The existence can be proved by showing that the graph of f is the maximal fixed point of the operator $\Psi : \mathcal{P}(A \times Stream_{\mathbf{N}}) \to \mathcal{P}(A \times Stream_{\mathbf{N}})$ defined by

$$\Psi(f) = \{(a, s) \mid head(s) = \pi_1(g(a)) \ \wedge \ (\pi_2(g(a)), tail(s)) \in f\} \ .$$

Strong extensionality again is used to show that the maximal fixed point is functional.

The equivalence induced by a function defined by coiteration can be characterized coinductively as follows:

Proposition 2.2.15 *Let $f : A \to Stream_{\mathbf{N}}$ be the coiteration function induced by the function $g : A \to \mathbf{N} \times A$. The equivalence induced by f is the greatest fixed point of the following operator $\Phi_f : \mathcal{P}(A \times A) \to \mathcal{P}(A \times A)$:*

$$\Phi_f(\mathcal{R}) = \{(a, a') \mid g(a) = (n, a_1) \ \wedge \ g(a') = (n, a_1') \ \wedge \ a_1 \mathcal{R} a_1'\} \ .$$

Proof It is immediate to see, by definition of f, that $\{(a, a') \mid f(a) = f(a')\}$ is a Φ_f-bisimulation. We are left to show that, if $(a, a') \in \mathcal{R} \subseteq \Phi_f(\mathcal{R})$, then $f(a) = f(a')$. If $\mathcal{R} \subseteq \Phi_f(\mathcal{R})$, then we can immediately define a function $g_{\mathcal{R}} : R \to \mathbf{N} \times R$ such that both functions $\pi_i \circ f : R \to Stream_{\mathbf{N}}$, for $i = 1, 2$, satisfy the coiteration scheme induced by $g_{\mathcal{R}}$. The thesis follows by the unicity of the coiteration function. $\square$

As we will see in Chapter 3, the coinduction principle above can be viewed as an instance of the general categorical coinduction principle, which characterizes equivalences induced by unique morphisms into final F-coalgebras.

We conclude this chapter by briefly discussing *corecursion* (see [Geu91] for more details).

Definition 2.2.16 (Corecursion on $Stream_{\mathbf{N}}$) *Let $A \in V_{\mathbf{N}}$, and let $g_1 : A \to \mathbf{N}$ and $g_2 : A \to A + Stream_{\mathbf{N}}$, where $+$ denotes the disjoint sum, i.e. $A + Stream_{\mathbf{N}} = \{(1,a) \mid a \in A\} \cup \{(2,s) \mid s \in Stream_{\mathbf{N}}\}$. The* corecursive *function induced by g_1, g_2 is the function $h : A \to Stream_{\mathbf{N}}$ defined by*

$$h = f \circ in_1 \ ,$$

where $f : A + Stream_{\mathbf{N}} \to Stream_{\mathbf{N}}$ is the coiteration function induced by $[< g_1, g_2 >, < head, in_2 \circ tail >] : A + Stream_{\mathbf{N}} \to \mathbf{N} \times (A + Stream_{\mathbf{N}})$, and $in_1 : A \to A + Stream_{\mathbf{N}}$ and $in_2 : Stream_{\mathbf{N}} \to A + Stream_{\mathbf{N}}$ are the canonical injections.
I.e. h is such that, for all $a \in A$, $h(a) = (g_1(a), f \circ g_2(a))$, where $f : A + Stream_{\mathbf{N}} \to Stream_{\mathbf{N}}$ is the unique function such that
$f \circ in_1(a) = (g_1(a), f \circ g_2(a))$ and $f \circ in_2(s) = s$.

Uniqueness can be proved using strong extensionality. Existence can be proved with a construction similar to the one used for coiteration.

The equivalence induced by a function defined by corecursion can be characterized via the Coinduction Principle up-to $\cup\nu_\Phi$ 2.1.8 of Section 2.1 as follows:

Proposition 2.2.17 *Let $h : A \to Stream_{\mathbf{N}} = f \circ in_1$ be the corecursive function induced by the functions $g_1 : A \to \mathbf{N}$, and $g_2 : A \to A + Stream_{\mathbf{N}}$. The equivalence induced by h, $\sim_h \subseteq A \times A$, can be characterized as follows:*

$$\sim_h = \bigcup\{\mathcal{R} \mid \mathcal{R} \subseteq A \times A \ \wedge \ in_1^+(\mathcal{R}) \subseteq \Phi_f(in_1^+(\mathcal{R}) \cup \sim_f)\} \ ,$$

where $\sim_f$ is the equivalence induced by the function f, and $in_1^+(\mathcal{R})$ denotes the image of $\mathcal{R}$ under in_1, i.e. $\{(in_1(a), in_1(a')) \mid (a, a') \in \mathcal{R}\}$.

Proof First of all notice that

$$\sim_h = in_1^{-1}(\sim_f) = \bigcup\{in_1^{-1}(\mathcal{R}) \mid \mathcal{R} \subseteq (A + Stream_{\mathbf{N}})^2 \ \wedge \ \mathcal{R} \subseteq \Phi_f(\mathcal{R})\} \ ,$$

where $in_1^{-1}(\mathcal{R})$ denotes the inverse image of $\mathcal{R}$ under in_1, i.e.
$\{(a, a') \mid (1, a)\mathcal{R}(1, a')\}$.

We prove that

$$\bigcup\{in_1^{-1}(\mathcal{R}) \mid \mathcal{R} \subseteq (A + Stream_{\mathbf{N}})^2 \ \wedge \ \mathcal{R} \subseteq \Phi_f(\mathcal{R})\} \ =$$

$$\bigcup\{\mathcal{R} \mid \mathcal{R} \subseteq A \times A \ \wedge \ in_1^+(\mathcal{R}) \subseteq \Phi_f(in_1^+(\mathcal{R}) \cup \sim_f)\} \ .$$

($\subseteq$) Let $\mathcal{R} \subseteq \Phi_f(\mathcal{R})$. Then $in_1^+(in_1^{-1}(\mathcal{R})) \subseteq \mathcal{R} \subseteq \Phi_f(\mathcal{R}) \subseteq \Phi_f(\sim_f) \subseteq \Phi_f(in_1^+(in_1^{-1}(\mathcal{R})) \cup \sim_f)$.
($\supseteq$) Let $\mathcal{R} \subseteq A \times A$ be such that $in_1^+(\mathcal{R}) \subseteq \Phi_f(in_1^+(\mathcal{R}) \cup \sim_f)$. Then, by the Coinduction Principle up-to $\cup\ \nu_{\Phi_f}$ 2.1.8 of Section 2.1, $in_1^+(\mathcal{R}) \subseteq \sim_f$. Hence $\mathcal{R} \subseteq in_1^{-1}(\sim_f)$. □

In Chapter 3, Section 3.3, we will see the categorical generalization of the principle up-to $\cup\ \nu_\Phi$ and the categorical generalization of the proposition above.

Chapter 3

Categorical Foundations: Final Semantics

In this chapter, we discuss coinduction principles and coinductive functions from a categorical standpoint. We try also to formalize rigorously the correspondences between the set-theoretic constructions and their categorical counterparts. More precisely, we present a general theory of *F-coalgebras, F-coalgebra homomorphisms,* and *F-bisimulations.* All can be seen as categorical counterparts/generalizations of corresponding set-theoretic concepts introduced in Chapter 2. The notion of F-bisimulation corresponds to the notion of set-theoretic bisimulation. The notion of *final* F-coalgebra corresponds to that of maximal fixed point of an operator. The *unique* F-coalgebra homomorphism into a final F-coalgebra corresponds to the notion of coiterative function. The theory we spell out here originated with [Acz88], but it was first developed to a considerable extent by Rutten and Turi in [RT93, RT94]. We shall try to push the categorical investigation so as to encompass also categorical counterparts to the variants of coinduction discussed in Chapter 2, and corecursive functions. For the sake of completeness, one should also mention the profound paper [Rut96], where the categorical notions discussed here are viewed as part of a broader duality between the theory of *universal algebras* and that of *universal coalgebras* (see also [RT94]). We shall not consider this issue here.

The chapter is organized as follows. In Section 3.1, we develop a theory of F-bisimulations in a general categorical setting. We build on the work of [Tur96], generalizing many results on F-bisimulations in set-theoretic, metric, and c.p.o.'s based categories (see [Acz88, RT93, Rut96]). For the categorical account we base our work on the useful and perspicuous book by Freyd and Scedrov [FS90]. In Section 3.2, we specialize these results to set-theoretic categories, which will be our main concern in the sequel of this thesis. In Section 3.3, we discuss some categorical variants of the coinduction principle, e.g. a categorical coinduction principle up-to, and a sound coinduction principle for establishing equivalences induced by *corecursive morphisms.* In Section 3.4, we study the relations between set-theoretic coinduction principles and categorical coinduction principles based on F-bisimulations on set-theoretic categories. In particular, we define a procedure, which, given a functor F and an F-coalgebra, provides a monotone set-theoretic operator of relations. This operator induces the set-theoretic coinduction principle corresponding to the categorical coinduction principle based on F-bisimulations. Finally, in Section 3.5 we discuss existence of final F-coalgebras in various categories; and we prove a sharpening of Aczel's Special Final Coalgebra Theorem, which gives a bound on the cardinality of the final coalgebra.

3.1 General Theory of Coalgebras and Bisimulations

In this section, we introduce the category $\mathcal{C}_F$ of *F-coalgebras* and *F-coalgebra homomorphisms*, and the notion of *F-bisimulation* on F-coalgebras, for a given endofunctor F on $\mathcal{C}$. F-bisimulations are relations, which can be endowed with suitable F-coalgebra structures. We build on [Tur96]. The notion of categorical relation is the one introduced in [FS90]. The objective of this section is to place in a general categorical setting many results on F-bisimulations which have been discussed for set-theoretic, metric, and c.p.o.'s based categories (see [Acz88, RT93, RT94, Rut96]). We recall two crucial results appearing in [Tur96]: the *strong extensionality* of *final F-coalgebras*, and the coinductive characterization of the equivalence induced by the *unique* morphism from an F-coalgebra into the *final* F-coalgebra. We also discuss general conditions on the category $\mathcal{C}$ and the functor F, under which:

- pullbacks of F-coalgebra homomorphisms are F-bisimulations;
- kernel pairs of identity morphisms in $\mathcal{C}$ are F-bisimulations;
- the generalization of the set-theoretic union of all F-bisimulations is an F-bisimulation;
- the composition of F-bisimulations is an F-bisimulation;
- the inverse image of an F-bisimulation is an F-bisimulation.

Furthermore, we analyze relations between the following notions:

- kernel pairs of F-coalgebra homomorphisms and F-bisimulation equivalences;
- categorical graphs of F-coalgebra homomorphisms and F-bisimulations.

In this section we use the categorical language of [FS90]. The categorical definitions which we need appear in Appendix A.

Throughout this section, $\mathcal{C}$ will be denote a category, and $F : \mathcal{C} \to \mathcal{C}$ an endofunctor on $\mathcal{C}$.

Definition 3.1.1 (F-coalgebra) *An F-*coalgebra *is a pair (X, α_X), where X is an object of $\mathcal{C}$ and $\alpha_X : X \to F(X)$ is an arrow of $\mathcal{C}$.*

F-coalgebras on a category $\mathcal{C}$ form a category:

Proposition 3.1.2 *Let $\mathcal{C}_F$ be the category whose objects are F-coalgebras and whose arrows between F-coalgebras (X, α_X) and (Y, α_Y), called F-coalgebra homomorphisms or simply F-coalgebra morphisms, are arrows $f : X \to Y$ of the category $\mathcal{C}$ such that the following diagram commutes*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\alpha_X"'] & Y \arrow[d, "\alpha_Y"] \\ F(X) \arrow[r, "F(f)"'] & F(Y) \end{tikzcd}$$

In order to define the notion of F-bisimulation, we need to introduce first the notion of binary *relation* on a category $\mathcal{C}$. A relation is defined in terms of equivalence classes of monic spans (this definition is taken from [FS90], see also [Tur96]):

Definition 3.1.3 *A* monic span $(\mathcal{R}, r_1, r_2)$ *on objects X, Y consists of an object $\mathcal{R}$ in $\mathcal{C}$, and two ordered arrows, $r_1 : \mathcal{R} \to X$ and $r_2 : \mathcal{R} \to Y$, which are jointly monic, i.e., for all $h, h' : Z \to \mathcal{R}$,*

$$r_1 \circ h = r_1 \circ h' \ \wedge \ r_2 \circ h = r_2 \circ h' \implies h = h' \ .$$

Monic spans on objects X and Y can be ordered as follows:

$$\begin{array}{c} (\mathcal{R}, r_1, r_2) \leq (\mathcal{R}', r_1', r_2') \iff \\ \exists f : \mathcal{R} \to \mathcal{R}'. \ \forall i = 1, 2. \ r_i = r_i' \circ f \ . \end{array}$$

Definition 3.1.4 *A binary* relation *on objects X, Y is an equivalence class of* monic spans *w.r.t. the relation $\leq \cap (\leq)^{-1}$.*

In what follows, by abuse of notation, we will denote relations by monic spans in place of equivalence classes of monic spans.

F-bisimulations on an F-coalgebra are relations with a suitable structure of F-coalgebra:

Definition 3.1.5 (F-bisimulation) *A relation $(\mathcal{R}, r_1, r_2)$ on objects X, Y is an F-bisimulation on the F-coalgebras (X, α_X) and (Y, α_Y), if there exists an arrow of $\mathcal{C}$, $\gamma : R \to F(R)$, such that the following diagram commutes:*

$$\begin{tikzcd} X \arrow[d, "\alpha_X"'] & \mathcal{R} \arrow[l, "r_1"'] \arrow[r, "r_2"] \arrow[d, "\gamma"] & Y \arrow[d, "\alpha_Y"] \\ F(X) & F(\mathcal{R}) \arrow[l, "F(r_1)"] \arrow[r, "F(r_2)"'] & F(Y) \end{tikzcd}$$

When the two F-coalgebras (X, α_X) and (Y, α_Y) in the definition above coincide, we will simply refer to F-bisimulations on the F-coalgebra (X, α_X).

A crucial property of the functor F is the preservation of *weak pullbacks* (see Appendix A):

Definition 3.1.6 *The functor F* preserves weak pullbacks *if, for all weak pullbacks (P, p_1, p_2), $(F(P), F(p_1), F(p_2))$ is a weak pullback.*

If F preserves weak pullbacks, then the pullbacks of F-coalgebra morphisms are F-bisimulations:

Lemma 3.1.7 *Suppose that F preserves weak pullbacks. If $f : (X, \alpha_X) \to (Y, \alpha_Y)$ and $g : (Z, \alpha_Z) \to (Y, \alpha_Y)$ are morphisms, then the pullback of f and g in $\mathcal{C}$ is an F-bisimulation on (X, α_X) and (Z, α_Z).*

In what follows, we assume that the category $\mathcal{C}$ has pullbacks.

The following theorem is an immediate consequence of Lemma 3.1.7, and it generalizes the fact that in set-theoretic categories the identity relation is an F-bisimulation. The kernel pair of the identity morphism can be viewed as a generalization of the identity relation.

Theorem 3.1.8 *If F preserves weak pullbacks, then the kernel pair of the identity is an F-bisimulation.*

The following Theorem 3.1.9 express the *Strong Extensionality Property* of *final* F-coalgebras in a general categorical setting. The proof of Theorem 3.1.9 is immediate, using the finality property.

Theorem 3.1.9 (Strong Extensionality of Final Coalgebras) *Any F-bisimulation $(\mathcal{R}, r_1, r_2)$ on the final F-coalgebra is such that $r_1 = r_2$.*

Now we study the relations between F-bisimulation *equivalences* and F-coalgebra morphisms. In order to define equivalence relations we need first to introduce the composition of relations:

Definition 3.1.10 *Let $(\mathcal{R}, r_1, r_2)$ be a relation on X, Y, and let $(\mathcal{R}', r'_1, r'_2)$ be a relation on Y, Z. The relation $(\mathcal{S}, s_1, s_2)$ is a* composition $\mathcal{R} \circ \mathcal{R}'$, *if there exists the pullback (P, p_1, p_2) of r_2 and r'_1 and a cover k such that the following diagram commutes*

$$\begin{tikzcd}
 & & S \arrow[dddll, two heads, "s_1"'] \arrow[dddrr, two heads, "s_2"] & & \\
 & & P \arrow[u, two heads, "k"] \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & & \\
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & & R' \arrow[dl, "r'_1"'] \arrow[dr, "r'_2"] & \\
X & & Y & & Z
\end{tikzcd}$$

Lemma 3.1.11 ([FS90],(1.56)) *Let $\mathcal{C}$ be cartesian category*[1] *with images. For all relations $(\mathcal{R}, r_1, r_2)$ and $(\mathcal{R}', r'_1, r'_2)$, a composition relation $(\mathcal{S}, s_1, s_2)$ can be constructed as follows*

$$
\begin{tikzcd}
& & S \arrow[ddddll, "s_1"'] \arrow[ddddrr, "s_2"] & & \\
& & Q \arrow[u, tail, "i"'] \arrow[ddl, "q_1"'] \arrow[ddr, "q_2"] & & \\
& & P \arrow[u] \arrow[dl, "p_1"] \arrow[dr, "p_2"'] & & \\
& R \arrow[dl, "r_1"] \arrow[dr, "r_2"'] & & R' \arrow[dl, "r'_1"] \arrow[dr, "r'_2"'] & \\
X & & Y & & Z
\end{tikzcd}
$$

where (P, p_1, p_2) is the pullback of r_2 and r'_1, and $i : Q \rightarrowtail S$ is a monic morphism.

Definition 3.1.12 (Equivalence Relation) *An endorelation $(\mathcal{R}, r_1, r_2)$ on an object X is an* equivalence *relation if*

$$
Id_{X \times X} \leq \mathcal{R} \ \wedge \ \mathcal{R}^{-1} \leq \mathcal{R} \ \wedge \ \mathcal{R} \circ \mathcal{R} \leq \mathcal{R} \ ,
$$

where $Id_{X \times X}$ is the identity relation on X, i.e. the relation (X, id_X, id_X), and $\mathcal{R}^{-1}$ denotes the inverse relation of $\mathcal{R}$, i.e. the relation $(\mathcal{R}, r_2, r_1)$.

F-bisimulation equivalences and F-coalgebra morphisms are related by Theorems 3.1.13 and 3.1.14 below.

Theorem 3.1.13 *Let $\mathcal{C}$ be a cartesian category with images. If F preserves weak pullbacks, then the kernel pair of an F-coalgebra morphism is an F-bisimulation equivalence.*

Proof The thesis follows using Lemma 3.1.7, since kernel pairs are equivalence relations. □

The following theorem is a sort of converse of Theorem 3.1.13 above, and it generalizes to categories other than *Set* Proposition 5.8 of [Rut96]. For the notion of *bicartesian regular category* see Appendix A.

Theorem 3.1.14 *Let $(\mathcal{R}, r_1, r_2)$ be an equivalence relation which is an F-bisimulation on the coalgebra (X, α_X), and let $X \xrightarrow{c} C$ be the coequalizer of*

[1] Following [FS90], a *cartesian* category is a category with finite products and equalizers.

$R \underset{r_2}{\overset{r_1}{\rightrightarrows}} X$. *Then* $\exists! \alpha_C : C \to F(C)$ *such that* c *is an* F*-coalgebra morphism between* (X, α_X) *and* (C, α_C).

Proof The existence and uniqueness of $\alpha_C : C \to F(C)$ follow from the definition of coequalizer. $\square$

There at least two possible ways of generalizing the set-theoretic union of all F-bisimulations on an F-coalgebra:

1. In categories based on sets, if F preserves weak pullbacks and there exists the final F-coalgebra, the union of all F-bisimulations coincides with the equivalence induced by the unique morphism $\mathcal{M}$ into the final F-coalgebra. Therefore, we can take as generalization of the union of all F-bisimulations the *kernel pair of* $\mathcal{M}$.

2. If the *generalized coequalizer* of all F-bisimulations exists, we can take the kernel pair of it as another possible generalization of the union of all F-bisimulations.

A natural question which arises is: when is the categorical generalization of the set-theoretic union of all F-bisimulations an F-bisimulation? I.e. when there exists the greatest F-bisimulation? Theorem 3.1.15 and Theorem 3.1.16 provide answers to this question. Theorem 3.1.15 is an immediate consequence of Lemma 3.1.7:

Theorem 3.1.15 *Suppose that* F *has final coalgebra,* (U, α_U)*. If* F *preserves weak pullbacks, then, for any* F*-coalgebra* (X, α_X)*, the kernel pair of the unique morphism* $\mathcal{M} : (X, \alpha_X) \to (U, \alpha_U)$ *is an* F*-bisimulation on* (X, α_X)*.*

Theorem 3.1.16 *If* F *preserves weak pullbacks, and if there exists the generalized coequalizer of all* F*-bisimulations on* (X, α_X)*, then the kernel* (K, k_1, k_2) *of the generalized coequalizer is an* F*-bisimulation on* (X, α_X)*.*

Proof The thesis follows from Lemma 3.1.7, since, by the property of the coequalizer $c : X \to C$, C can be endowed (uniquely) with an F-coalgebra structure, (C, α_C), such that c is a morphism between the coalgebras (X, α_X) and (C, α_C). $\square$

The following theorem generalizes the set-theoretic fact that pairs of elements of an F-coalgebra, which are F-bisimilar, have the same image by the unique morphism into the final F-coalgebra.

Theorem 3.1.17 *Let* $(\mathcal{R}, r_1, r_2)$ *be an* F*-bisimulation on the* F*-coalgebra* (X, α_X)*. If* F *has final* F*-coalgebra,* (U, α_U)*, then the unique morphism* $\mathcal{M} : (X, \alpha_X) \to (U, \alpha_U)$ *is such that* $\mathcal{M} \circ r_1 = \mathcal{M} \circ r_2$*.*

Now we discuss conditions under which the *composition relation* of F-bisimulations is an F-bisimulation.

The following theorem can be easily proved, using Lemma 3.1.7:

Theorem 3.1.18 *Let $\mathcal{C}$ be a cartesian category with images. If F preserves weak pullbacks, then the composition of F-bisimulations is an F-bisimulation.*

Examples of cartesian categories with images, besides Set, are all the categories normally used in the semantics of programming languages, e.g. the category CPO of c.p.o.'s and Scott continuous functions, and the category CMS of complete metric spaces and non distance increasing functions.

In the following definition we generalize the notion of set-theoretic *graph* of a function:

Definition 3.1.19 (Graph, [FS90]) *The* graph *of a morphism $f : X \to Y$ of $\mathcal{C}$ is the relation (X, id_X, f).*

The following theorem is a generalization of [RT94], Proposition 3.8, and of [Rut96], Theorem 2.5.

Theorem 3.1.20 *If $f : (X, \alpha_X) \to (Y, \alpha_Y)$ is an F-coalgebra morphism, then the graph of f is an F-bisimulation. Vice versa, if $f : X \to Y$ is a morphism in $\mathcal{C}$ and its graph is an F-bisimulation on (X, α_X) and (Y, α_Y), then f is an F-coalgebra morphism from (X, α_X) to (Y, α_Y).*

Theorem 3.1.20($\Rightarrow$) has an interesting generalization in *AC regular categories* of [FS90] (see Appendix A). In *AC regular categories* a categorical generalization of the Axiom of Choice holds.

In order to give Theorem 3.1.22 below, which generalizes Theorem 3.1.20($\Rightarrow$), we need the following technical lemma:

Lemma 3.1.21 ([FS90]) *Let $\mathcal{C}$ be an AC regular category. Then, for all morphisms $f : C \to X$, $g : C \to Y$, there exists a monic $i : C' \mapsto C$ such that $f \circ i : C' \to X$ and $g \circ i : C' \to Y$ are jointly monic, and $Img(f) = Img(f \circ i)$ and $Img(g) = Img(g \circ i)$.*

The following theorem is easily proved using Lemma 3.1.21.

Theorem 3.1.22 *Let $\mathcal{C}$ be an AC regular category. If $f : (C, \alpha_C) \to (X, \alpha_X)$ and $g : (C, \alpha_C) \to (Y, \alpha_Y)$ are F-coalgebra morphisms, then there exist $f' : C' \to X$ and $g' : C' \to Y$ such that $Img(f) = Img(f')$, $Img(g) = Img(g')$, and (C', f', g') is an F-bisimulation on (X, α_X) and (Y, α_Y).*

Theorem 3.1.22 above generalizes to categories other than Set Lemma 5.3 of [Rut96].

Finally, we analyze conditions under which the *inverse image* of an F-bisimulation by an F-coalgebra morphism is an F-bisimulation. First of all we need to define the inverse image of a relation:

Definition 3.1.23 (Inverse Image of Relations) *Let $\mathcal{C}$ be a category with products, let $f : X \to Y$ be a morphism in $\mathcal{C}$, and let $(\mathcal{R}, r_1, r_2)$ be an endorelation on Y. Define the* inverse image *of $\mathcal{R}$ by f, denoted by $f^{-1}(R)$, as the inverse image, if it exists, of $< r_1, r_2 >: \mathcal{R} \mapsto Y \times Y$ by the morphism $< f \circ \pi_1, f \circ \pi_2 >: X \times X \to Y \times Y$.*

For the next theorem we need to work in a category with products and in which inverse images of morphisms exist. The natural context in which these conditions are verified is that of a *pre-logos* of [FS90] (see Appendix A).

Theorem 3.1.24 *Let $\mathcal{C}$ be a pre-logos. Let F be an endofunctor on $\mathcal{C}$ which preserves weak pullbacks and such that, for all objects A in $\mathcal{C}$, there exists $i_A : F(A) \times F(A) \to F(A \times A)$ which satisfies the following two conditions*

1. $\forall <u, v>: C \to F(D) \times F(D).\ \forall f : D \to E.\ < F(f \circ \pi_1), F(f \circ \pi_2) > \circ i_D \circ < u, v >=< F(f) \circ \pi_1, F(f) \circ \pi_2 > \circ < u, v >$;
2. $\forall < u, v >: C \to D \times D.\ i_D \circ < F(u), F(v) >= F(< u, v >)$.

If $f : (X, \alpha_X) \to (Y, \alpha_Y)$ is an F-coalgebra morphism and $(\mathcal{R}, r_1, r_2)$ with $\gamma : \mathcal{R} \to F(\mathcal{R})$ is an F-bisimulation on (Y, α_Y), then $f^{-1}(\mathcal{R})$ can be endowed with an F-bisimulation structure.

Proof Let $\alpha_{X\times X} = i_X \circ < \alpha_X \circ \pi_1, \alpha_X \circ \pi_2 >$ and $\alpha_{Y\times Y} = i_Y \circ < \alpha_Y \circ \pi_1, \alpha_Y \circ \pi_2 >$. Then $< f \circ \pi_1, f \circ \pi_2 >$ is an F-coalgebra morphism from $(X \times X, \alpha_{X\times X})$ to $(Y \times Y, \alpha_{Y\times Y})$, since:
$i_Y \circ < \alpha_Y \circ \pi_1, \alpha_Y \circ \pi_2 > \circ < f \circ \pi_1, f \circ \pi_2 >= i_Y \circ < \alpha_Y \circ f \circ \pi_1, \alpha_Y \circ f \circ \pi_2 >= i_Y \circ < F(f) \circ \alpha_X \circ \pi_1, F(f) \circ \alpha_X \circ \pi_2 >= i_Y \circ < F(f) \circ \pi_1, F(f) \circ \pi_2 > \circ < \alpha_X \circ \pi_1, \alpha_X \circ \pi_2 >=$, by condition 1, $i_Y \circ < F(f \circ \pi_1), F(f \circ \pi_2) > \circ i_X \circ < \alpha_X \circ \pi_1, \alpha_X \circ \pi_2 >=$, by condition 2, $F(< f \circ \pi_1, f \circ \pi_2 >) \circ i_X \circ < \alpha_X \circ \pi_1, \alpha_X \circ \pi_2 >$.
Moreover $< r_1, r_2 >$ is an F-coalgebra morphism from $(\mathcal{R}, \gamma)$ to $(Y \times Y, \alpha_{Y\times Y})$, since:
$F(< r_1, r_2 >) \circ \gamma =$, by condition 2, $i_Y \circ < F(r_1), F(r_2) > \circ \gamma = i_Y \circ < F(r_1) \circ \gamma, F(r_2) \circ \gamma >= i_Y \circ < \alpha_Y \circ r_1, \alpha_Y \circ r_2 >= i_Y \circ < \alpha_Y \circ \pi_1, \alpha_Y \circ \pi_2 > \circ < r_1, r_2 >$.
By definition of inverse image, $(f^{-1}(\mathcal{R}), g : f^{-1}(\mathcal{R}) \to \mathcal{R}, f^{-1}(\mathcal{R}) \mapsto X \times X)$, for some g, is the pullback of $< f \circ \pi_1, f \circ \pi_2 >$ and $< r_1, r_2 >$, hence, since $< f \circ \pi_1, f \circ \pi_2 >$ and $< r_1, r_2 >$ are morphisms of F-coalgebras, then by Lemma 3.1.7 $(f^{-1}(\mathcal{R}), g : f^{-1}(\mathcal{R}) \to \mathcal{R}, f^{-1}(\mathcal{R}) \mapsto X \times X)$ is an F-bisimulation on the coalgebras $(\mathcal{R}, \gamma)$ and $(X \times X, \alpha_{X\times X})$. $\square$

3.2 Coalgebras and Bisimulations in Categories of Sets

In this section we specialize the main results of Section 3.1 to the case of categories whose objects are sets or classes of a possibly non-wellfounded universe of sets (see Chapter 2, Section 2.2.1). Many of these results appear in [RT93, RT94, Rut96]. As an application of these categorical results, we will see that the statement of the Antifoundation Axiom FCU is precisely equivalent to the fact that the non-wellfounded universe V_U is *final coalgebra* for the functor $\mathcal{P}(\cdot \cup U)$ on the category of non-wellfounded classes. Correspondingly, the Strong Extensionality Axiom amounts to the strong extensionality of V_U viewed as final coalgebra. A similar result can be stated also for $\overline{X}_1$. It is interesting to point out that the original formulation of the axiom $\overline{X}_1$ of [FH83] is literally the finality of the universe.

The set-theoretic categories which we consider in this section are the following:

Definition 3.2.1

- *Let $Set(U)$ ($Set^*(U)$) be the category whose objects are the (non-)wellfounded sets belonging to a Universe of ZF_0^- (FCU), and whose arrows from a set A to a set B are the functions with domain A and codomain B, tagged with A and B.*
- *Let $Class(U)$ ($Class^*(U)$) be the category whose objects are the classes of (non-)wellfounded sets belonging to a Universe of ZF_0^- (FCU), and whose arrows from a class A to a class B are the functions with domain A and codomain B, tagged with A and B.*
- *Let $\mathcal{HC}_\kappa(U)$ ($(\mathcal{HC}_\kappa)^*(U)$) be the category whose objects are the wellfounded (non wellfounded) sets whose hereditary cardinal is less than κ, and whose arrows between objects A and B are the functions with domain A and codomain B, tagged with A and B.*
- *Let $Card$ ($CARD$) be the category whose objects are the cardinals (including Ord), and whose arrows between objects A and B are the functions with domain A and codomain B, tagged with A and B.*

In the above definition, we stress the fact that arrows in a category are not just set-theoretic functions, but they are triples: a set-theoretic function, with domain and codomain. In principle, only the tag corresponding to the codomain is strictly necessary, since a set-theoretic function does not determine uniquely its codomain.

Throughout this section, let $\mathcal{C}^{\mathcal{S}}$ denote one of the set-theoretic categories defined above, and let F denote an endofunctor on $\mathcal{C}^{\mathcal{S}}$.

Since relations in set-theoretic categories are set-theoretic relations, the notion of F-bisimulation takes the following simpler form:

Definition 3.2.2 *An F-bisimulation on the F-coalgebras (X, α_X) and (Y, α_Y) is a set-theoretic relation $R \subseteq X \times Y$ such that there exists an arrow of $\mathcal{C}$, $\gamma : \mathcal{R} \to F(\mathcal{R})$, making the following diagram commute:*

$$\begin{tikzcd}
X \arrow[d, "\alpha_X"'] & \mathcal{R} \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\gamma"] & Y \arrow[d, "\alpha_Y"] \\
F(X) & F(\mathcal{R}) \arrow[l, "F(\pi_1)"] \arrow[r, "F(\pi_2)"'] & F(Y)
\end{tikzcd}$$

An immediate consequence of Theorem 3.1.8 of Section 3.1 is the following

Theorem 3.2.3 (Identity is an F-bisimulation) *Let (X, α_X) be an F-coalgebra. If F preserves weak pullbacks, then the identity relation on X is an F-bisimulation on (X, α_X).*

From Theorems 3.1.8 and 3.1.18, it follows immediately that

Theorem 3.2.4 *If F preserves weak pullbacks, then the union of all F-bisimulations on the F-coalgebra (X, α_X) is an F-bisimulation equivalence.*

The Strong Extensionality Theorem for final coalgebras takes the form:

Theorem 3.2.5 (Strong Extensionality of Final Coalgebras) *Let (U, α_U) be a final F-coalgebra. Then, for all $u, u' \in U$,*

$$u = u' \iff \exists\, \mathcal{R}\ \ F\text{-bisimulation on } (U, \alpha_U).\ u\, \mathcal{R}\, u'\ .$$

From Theorems 3.1.17 and 3.1.15 of Section 3.1, it follows that the equivalences induced by unique morphisms into final coalgebras can be characterized coinductively as *greatest* F-bisimulations. This yields the categorical version of the coiteration scheme of Chapter 2, Section 2.2.2.

Theorem 3.2.6 *Suppose that F has final F-coalgebra (U, α_U). Let (X, α_X) be an F-coalgebra. If F preserves weak pullbacks, then the equivalence induced by the unique morphism $\mathcal{M} : (X, \alpha_X) \to (U, \alpha_U)$ can be characterized as follows:*

$$\mathcal{M}(x) = \mathcal{M}(x') \iff \exists\, \mathcal{R}\ \ F\text{-bisimulation on } (X, \alpha_X).\ x\, \mathcal{R}\, x'\ .$$

Moreover, the union of all F-bisimulations on (X, α_X) is an F-bisimulation on (X, α_X), denoted by $\sim^F_{(X,\alpha_X)}$.

Finally, we point out that the Antifoundation Axiom FCU is equivalent to the finality of the non-wellfounded universe of sets viewed as coalgebra in the category $Class^*(U)$. The proof of the following proposition is straightforward, using the fact that the non-wellfounded universe of sets $V_U = \mathcal{P}(V_U \cup U)$.

Proposition 3.2.7

$$FCU \iff (V_U, id_{V_U}) \text{ is final } \mathcal{P}(\cdot \cup U)\text{-coalgebra}\ ,$$

where $\mathcal{P}(\cdot \cup U) : Class^*(U) \to Class^*(U)$.

3.3 Variants of the Categorical Coinduction Principle

In this section, we discuss variants of the Coinduction Principle 3.2.6. In particular, we prove the soundness and completeness of a categorical coinduction principle up-to $+fix_F$, where fix_F is any fixed point of the functor F. This principle can be viewed as a sort of counterpart of the set-theoretic coinduction principle up-to $\cup\nu_\Phi$, which arises in the characterization of corecursive functions. As we will see, the categorical coinduction principle up-to $+fix_F$ is related to categorical corecursion. For simplicity, in this section we will work in set-theoretic categories, but all the results can be easily stated in a more general categorical framework.

In order to prove the categorical coinduction principle up-to $+fix_F$, we need first to introduce the notion of F-bisimulation up-to $+fix_F$:

Definition 3.3.1 (**F-bisimulation up-to** $+fix_F$) *Let $F : \mathcal{C}^\mathcal{S} \to \mathcal{C}^\mathcal{S}$, let fix_F denote any fixed point of F, and let $(X + fix_F, [\alpha_1, F(in_2)])$ be an F-coalgebra. An F-bisimulation up-to $+fix_F$ on the F-coalgebra $(X + fix_F, [\alpha_1, F(in_2)])$ is a relation $\mathcal{R} \subseteq (X + fix_F) \times (X + fix_F)$ such that there exists an arrow of $\mathcal{C}^\mathcal{S}$, $\gamma : \mathcal{R} \to F(\mathcal{R} + fix_F)$ making the following diagram commute:*

$$\begin{tikzcd}
X + fix_F \arrow[d, "{[\alpha_1, F(in_2)]}"'] & \mathcal{R} \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\gamma"] & X + fix_F \arrow[d, "{[\alpha_1, F(in_2)]}"] \\
F(X + fix_F) & F(\mathcal{R} + fix_F) \arrow[l, "{F([\pi_1, in_2])}"] \arrow[r, "{F([\pi_2, in_2])}"'] & F(X + fix_F)
\end{tikzcd}$$

The coinduction principle up-to $+fix_F$ arises from the following characterization of the equivalence induced by the unique morphisms into the final F-coalgebra:

Proposition 3.3.2 *Let $F : \mathcal{C}^\mathcal{S} \to \mathcal{C}^\mathcal{S}$ be a functor preserving weak pullbacks, which has final coalgebra (U, α_U). Let fix_F denote any fixed point of F, and let $(X + fix_F, [\alpha_1, F(in_2)])$ be an F-coalgebra. The equivalence $\sim^F_{(X+fix_F,[\alpha_1,F(in_2)])}$, induced by the unique morphism from the F-coalgebra $(X + fix_F, [\alpha_1, F(in_2)])$ to the final F-coalgebra (U, α_U), can be characterized as follows:*

$$\sim^F_{(X+fix_F,[\alpha_1,F(in_2)])} =$$

$$\bigcup\{\mathcal{R} | \mathcal{R}\ \ F\text{-bisimulation up-to} + fix_F \text{ on } (X + fix_F, [\alpha_1, F(in_2)])\}\ .$$

Proof We have to show that

$$\bigcup\{\mathcal{R} | \mathcal{R}\ \ F\text{-bisimulation } on\ (X + fix_F, [\alpha_1, F(in_2)])\}\ =$$

$$\bigcup\{\mathcal{R} | \mathcal{R}\ \ F\text{-bisimulation } up\text{-}to + fix_F\ on\ (X + fix_F, [\alpha_1, F(in_2)])\}\ .$$

($\subseteq$) If $\mathcal{R}$ is an F-bisimulation on $(X + fix_F, [\alpha_1, F(in_2)])$, with $\gamma : \mathcal{R} \to F(\mathcal{R})$, then it is easy to check that $\mathcal{R}$ with $F(in_1) \circ \gamma : \mathcal{R} \to F(\mathcal{R} + fix_F)$, is an F-bisimulation up-to $+fix_F$ on $(X + fix_F, [\alpha_1, F(in_2)])$.
($\supseteq$) If $\mathcal{R}$ is an F-bisimulation up-to $+fix_F$ on $(X + fix_F, [\alpha_1, F(in_2)])$, with $\gamma : \mathcal{R} \to F(\mathcal{R} + fix_F)$, it is easy to check that, by the general Definition 3.1.5,

the relation $(\mathcal{R} + fix_F, [\pi_1, in_2], [\pi_2, in_2])$ is an F-bisimulation on on $(X + fix_F, [\alpha_1, F(in_2)])$ with $[\gamma, F(in_2)] : \mathcal{R} + fix_F \to F(\mathcal{R} + fix_F)$. $\square$

Now we introduce the categorical corecursion scheme for defining morphisms into coinductive data types viewed as final coalgebras. We will use the coinduction principle above to derive a *sound* coinduction principle for establishing equivalences induced by corecursive morphisms.

Definition 3.3.3 (Corecursion) *Let $F : \mathcal{C}^{\mathcal{S}} \to \mathcal{C}^{\mathcal{S}}$ be a functor which has final coalgebra (U, α_U). Let fix_F denote any fixed point of F, and let $(X + fix_F, [\alpha_1, F(in_2)])$ be an F-coalgebra. The* corecursive *morphism $h : X \to U$ is defined by $f \circ in_1$, where f is the unique F-coalgebra morphism from $(X + fix_F, [\alpha_1, F(in_2)])$ to the final coalgebra (U, α_U), i.e.*

$$\begin{tikzcd}
X \arrow[r, "in_1"] & X + fix_F \arrow[r, "f"] \arrow[d, "{[\alpha_1, F(in_2)]}"'] & U \arrow[d, "\alpha_U"] \\
& F(X + fix_F) \arrow[r, "F(f)"'] & F(U)
\end{tikzcd}$$

Using the coinduction principle of Proposition 3.3.2, one can prove the following theorem, which guarantees the soundness of a coinduction principle based on F-bisimulations up-to fix_F for establishing equivalences induced by corecursive morphisms.

Theorem 3.3.4 *Let $F : \mathcal{C}^{\mathcal{S}} \to \mathcal{C}^{\mathcal{S}}$ be a functor preserving weak pullbacks, which has final coalgebra (U, α_U). Let fix_F denote any fixed point of F, let $(X + fix_F, [\alpha_1, F(in_2)])$ be an F-coalgebra, and let h be the corecursive morphism induced by $(X + fix_F, [\alpha_1, F(in_2)])$. Then*

$$\bigcup\{\mathcal{R} \mid \mathcal{R} \subseteq X \times X \ \wedge\ in_1^+(\mathcal{R}) \ F\text{-bisimulation up-to } + fix_F \text{ on } (X + fix_F, [\alpha_1, F(in_2)])\} \ \subseteq \ \sim_h ,$$

where $\sim_h$ is the equivalence induced by the corecursive morphism h.

Proof First of all notice that $\sim_h = in_1^{-1}(\sim^F_{(X+fix_F,[\alpha_1,F(in_2)])})$. If $in_1^+(\mathcal{R})$ is an F-bisimulation up-to $+fix_F$ on $(X + fix_F, [\alpha_1, F(in_2)])$, then by Proposition 3.3.2, $in_1^+(\mathcal{R}) \subseteq \sim^F_{(X+fix_F,[\alpha_1,F(in_2)])}$, and hence $\mathcal{R} \subseteq in_1^{-1}(\sim^F_{(X+fix_F,[\alpha_1,F(in_2)])})$.
$\square$

From Theorem 3.3.4, the soundness of a coinduction principle for establishing equivalences induced by corecursive morphisms follows immediately. This principle can be viewed as the categorical formulation of the set-theoretic principle of Chapter 2, Proposition 2.2.17. However, the completeness of the categorical principle above deserves further study.

Finally, it would be interesting to study categorical counterparts of other set-theoretic coinduction principles of Chapter 2, Section 2.1, e.g. a categorical coinduction principle up-to $\sim^F_{(X,\alpha_X)} \circ\_\circ \sim^F_{(X,\alpha_X)}$. We leave this as an open problem.

3.4 From Categories to Sets

In this section we study the relations between the categorical coinduction principle based on F-bisimulations discussed in Sections 3.1 and 3.2 of this chapter, and the set-theoretic Coinduction Principle 2.1.5 of Chapter 2. We work in set-theoretic categories. However, it would be interesting to extend the results of this section in a more general categorical setting.

In particular, in this section we define an *inductive* procedure for deriving set-theoretic coinduction principles from categorical coinduction principles. Given a functor F and F-coalgebras (X, α) and (Y, β), these induce a coinduction principle based on F-bisimulations, i.e. the greatest F-bisimulation on (X, α) and (Y, β) is the union of all F-bisimulations on (X, α) and (Y, β). What we do is to provide the corresponding set-theoretic coinduction principle, i.e. a monotone operator $\Phi : \mathcal{P}(X \times Y) \rightarrow \mathcal{P}(X \times Y)$ such that: a relation $\mathcal{R} \subseteq X \times Y$ is an F-bisimulation on (X, α) and (Y, β) if and only if $\mathcal{R}$ is a Φ-bisimulation.

It would be extremely interesting to go the other way round, i.e. to be able to provide categorical coinduction principles starting from set-theoretic coinduction principles. But this seems problematic, since F-bisimulations convey more information than set-theoretic bisimulations.

The class $\mathcal{F}$ of *covariant* functors which we consider involves the constructors which are normally used for defining final semantics, i.e. cartesian product, disjoint sum, powerset, and $C \rightarrow (\cdot)$ constructors. This class is defined as follows:

Definition 3.4.1 *Let $\mathcal{F}$ be the class of functors $F : \mathcal{C}^{\mathcal{S}} \rightarrow \mathcal{C}^{\mathcal{S}}$ defined as follows:*

$$(\mathcal{F} \ni) \;\; F(\cdot) ::= Id(\cdot) \mid F_C(\cdot) \mid F(\cdot) \times F(\cdot) \mid F(\cdot) + F(\cdot) \mid \mathcal{P}(F(\cdot)) \mid C \rightarrow (\cdot) \, ,$$

where

- *$Id(\cdot)$ is the identity functor, defined by*

$$\begin{cases} Id(A) = A & \textit{for } A \textit{ object of } \mathcal{C}^{\mathcal{S}} \\ Id(f) = f & \textit{for } f \textit{ arrow of } \mathcal{C}^{\mathcal{S}} \, , \end{cases}$$

- *$F_C(\cdot)$, for C object of $\mathcal{C}^{\mathcal{S}}$, is the constant functor, defined by*

$$\begin{cases} F_C(A) = C & \textit{for } A \textit{ object of } \mathcal{C}^{\mathcal{S}} \\ F_C(f) = id_C & \textit{for } f \textit{ arrow of } \mathcal{C}^{\mathcal{S}} \, , \end{cases}$$

Proposition 3.4.2 *Let $F : \mathcal{C}^{\mathcal{S}} \rightarrow \mathcal{C}^{\mathcal{S}}$ be a functor in $\mathcal{F}$, and let (X, α), (Y, β) be F-coalgebras. Proceeding by induction on the structure of F, we define a monotone operator $\Phi : \mathcal{P}(X \times Y) \rightarrow \mathcal{P}(X \times Y)$ such that*

$$\mathcal{R} \subseteq X \times Y \textit{ is an } F\textit{-bisimulation on } (X, \alpha_X) \textit{ and } (Y, \alpha_Y) \iff \mathcal{R} \subseteq \Phi(\mathcal{R}) \, .$$

(Id) *Let $F(\cdot) = Id(\cdot)$. We define the operator $\Phi : \mathcal{P}(X \times Y) \rightarrow \mathcal{P}(X \times Y)$ as follows:*

$$\Phi(\mathcal{R}) = \{(x, y) \mid (\alpha(x), \beta(y)) \in \mathcal{R}\} \, .$$

(F_C) *Let* $F(\cdot) = F_C(\cdot)$. *We define the operator* $\Phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *as follows:*

$$\Phi(\mathcal{R}) = \{(x, y) \mid \alpha(x) = \beta(y)\} \ .$$

($\times$) *Let* $F(\cdot) = F_1(\cdot) \times F_2(\cdot)$. *The* F*-coalgebras* (X, α) *and* (Y, β) *induce* F_i*-coalgebras* $(X, \pi_i \circ \alpha)$ *and* $(Y, \pi_i \circ \beta)$, *respectively, for* $i = 1, 2$. *Let* $\Phi_i : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *be the monotone operator corresponding to the categorical coinduction principle induced by the* F_i*-coalgebras* $(X, \pi_i \circ \alpha)$ *and* $(Y, \pi_i \circ \beta)$. *Then the operator* $\Phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *is defined as follows:*

$$\Phi(\mathcal{R}) = \Phi_1(\mathcal{R}) \cap \Phi_2(\mathcal{R}) \ .$$

($+$) *Let* $F(\cdot) = F_1(\cdot) + F_2(\cdot)$. *Then the* F*-coalgebras* (X, α) *and* (Y, β) *are of the following shape:* $\alpha = [\alpha_1, \alpha_2] : X_1 + X_2 \to F_1(X) + F_2(X)$, *with* $\alpha_i : X_i \to F_i(X)$, *and* $\beta = [\beta_1, \beta_2] : Y_1 + Y_2 \to F_1(Y) + F_2(Y)$, *with* $\beta_i : Y_i \to F_i(Y)$, *respectively. Let* $\overline{\alpha_i} : X \to F_i(X)$ *be any* F*-coalgebra such that* $\overline{\alpha_i}_{|X_i} = \alpha_i$, *and let* $\overline{\beta_i} : Y \to F_i(Y)$ *be any* F*-coalgebra such that* $\overline{\beta_i}_{|Y_i} = \beta_i$. *Let* $\Phi_i : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *be the monotone operator corresponding to the categorical coinduction principle induced by the* F_i*-coalgebras* $(X, \overline{\alpha_i})$ *and* $(Y, \overline{\beta_i})$. *Then the operator* $\Phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *is defined as follows:*

$$\Phi(\mathcal{R}) = \Phi_1(\mathcal{R} \cap (X_1 \times Y_1)) \cup \Phi_2(\mathcal{R} \cap (X_2 \times Y_2)) \ .$$

($\mathcal{P}$) *Let* $F(\cdot) = \mathcal{P}(F_1(\cdot))$. *The* F*-coalgebras* (X, α) *and* (Y, β) *induce* F_1*-coalgebras* $\alpha_i : X \to F_1(X)$ *for* $i \in I$, *and* $\beta_j : Y \to F_1(Y)$ *for* $j \in J$, *respectively, such that* $\forall x \in X.\ \alpha_i(x) \in \alpha(x)$, *and* $\forall y \in Y.\ \beta_j(y) \in \beta(y)$, *where* I, J *are suitable sets of indices. Let* $\Phi_{ij} : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *be the monotone operator corresponding to the categorical coinduction principle induced by the* F_1*-coalgebras* (X, α_i) *and* (Y, β_j). *The operator* $\Phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *is defined as follows:*

$$\begin{aligned} &\Phi(\mathcal{R}) \ = \\ &\{(x, y) \mid \forall i \in I\ \exists j \in J.\ (x, y) \in \Phi_{ij}(\mathcal{R}) \wedge \forall j \in J\ \exists i \in I.\ (x, y) \in \Phi_{ij}(\mathcal{R})\} \ . \end{aligned}$$

($C \to (\cdot)$) *Let* $F(\cdot) = C \to F_1(\cdot)$. *The* F*-coalgebras* (X, α) *and* (Y, β) *induce* F_1*-coalgebras* $\alpha_c : X \to F_1(X)$, *and* $\beta_c : Y \to F_1(Y)$, *for all* $c \in C$, *respectively. Let* $\Phi_c : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *be the monotone operator corresponding to the categorical coinduction principle induced by the* F_1*-coalgebras* (X, α_c) *and* (Y, β_c). *The operator* $\Phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ *is defined as follows:*

$$\Phi(\mathcal{R}) = \bigcap_{c \in C} \Phi_c(\mathcal{R}) \ .$$

We have discussed the relations between the set-theoretic and the categorical coinduction principle for the class $\mathcal{F}$ of covariant functors. It would be interesting to investigate also *mixed induction-coinduction* principles, and possibly to extend the coalgebraic approach to *mixed covariant-contravariant* functors. This issue should be related with Freyd's work on *algebraically compact categories* ([Fre90, Fre92]). We do not elaborate on this.

3.5 On the Existence Theorems of Final Coalgebras

A convenient general categorical setting for giving *final semantics* to programming languages has to satisfy the following three prerequisites (see Part II).

1. It must be based on a category $\mathcal{C}$, which is rich enough to accommodate the data necessary for giving the operational semantics of programming languages.
2. It must allow for a rich enough class, $\mathcal{F}$, of endofunctors on $\mathcal{C}$, so that the *operational semantics* of the language be representable as an F-coalgebra, for a suitable endofunctor $F \in \mathcal{F}$.
3. Given $F \in \mathcal{F}$, it must allow for a rich enough class of F-coalgebras, for the final F-coalgebra to *exist*; or equivalently, for the induced category $\mathcal{C}_F$ of F-coalgebras to have a terminal object.

Since the seminal book of Peter Aczel [Acz88], various authors have addressed these issues. We recall Aczel and Mendler [AM89], Rutten and Turi [RT93, RT94], Barr [Bar93, Bar94], Aczel [Acz93], Rutten [Rut96]. The literature is quite vast, so we shall only point out some results and make a number of remarks useful for the subsequent development of our work. Some of these results, although simple, do not seem to have been made explicitly before in the literature, to the best of our knowledge. This section is *not self-contained*.

Let us give first a brief historical account of the main results.

In [Acz88], Aczel used as basic category $\mathcal{C}$, the category $Class^*$, whose objects are the subclasses of a universe of sets satisfying the Antifoundation Axiom AFA (X_1), and whose morphisms are the functional classes tagged with domain and codomain. He defined two kinds of endofunctors over $Class^*$, the *standard functors* which preserve *weak pullbacks* and the standard functors which are *uniform on maps*. He proved that both kinds of functors have final coalgebras. He called these theorems the *Final Coalgebra Theorem* and the *Special Final Coalgebra Theorem*, respectively. While the *Special Final Coalgebra Theorem* gives an independent characterization of the final coalgebra as the maximal fixed point of the set theoretical operator underlying the functor F, the *Final Coalgebra Theorem* is more abstract and characterizes the final coalgebra as a quotient.

Aczel and Mendler, in [AM89], reduced the hypotheses in the *Final Coalgebra Theorem* to the fact that the functor is set-based.

Rutten and Turi [RT93] proved *Final Coalgebra Theorem*'s for ω-continuous functors in the category CMS, of complete metric spaces and non distance increasing functions, and the category $CPO_\perp$, of complete partial orders and strict Scott-continuous functions (see also [SP82]).

Barr in [Bar93, Bar94] proved a *Final Coalgebra Theorem* for *κ-accessible functors* in the category *Set*, provided κ is regular and such that $\lambda < \kappa \Longrightarrow 2^\lambda \leq \kappa$. This result was clarified and substantially enhanced by Rutten in [Rut96].

Finally, Turi in [Tur96] carried out a categorical investigation of the notion of "uniform on maps" and the *Special Final Coalgebra Theorem* in $Class^*$.

We recall first some crucial definitions:

Definition 3.5.1 *Let $\mathcal{C}$ be a category of classes of possibly non-wellfounded sets, and let $F : \mathcal{C} \longrightarrow \mathcal{C}$ be a functor.*

- F *is* inclusion preserving *if*

$$\forall A, B.\ A \subseteq B \Longrightarrow\ (F(A) \subseteq F(B)\ \wedge\ F(\iota_{A,B}) = \iota_{F(A),F(B)})\ ,$$

  where $\iota_{A,B} : A \longrightarrow B$ is the inclusion map from A to B.

- F *is* set based *if, for each class A and each $a \in F(A)$, there exist a set $A_0 \subseteq A$ and $a_0 \in F(A_0)$ such that $a = F(\iota_{A_0,A})(a_0)$.*

- F *is* standard *if it is inclusion preserving and set-based.*

- F *is* uniform on maps *if, for all A and $f : A \longrightarrow V$, there exists a mapping $\phi_A : F(A) \longrightarrow V_{X_A}$ such that $F(f) = \widehat{f} \circ \phi_A$.*
  We recall that V_{X_A} is the greatest solution of the equation $Y = \mathcal{P}(Y \cup X_A)$, where X_A is a class of atoms in one-to-one correspondence with the class A, and $\widehat{f} : V_{X_A} \to V$ is the function defined by

$$\widehat{f}(B) = \{f(a) \mid x_a \in B \cap X_A\} \cup \{\widehat{f}(B') \mid B' \in B \cap V_{X_A}\}\ .$$

The question arises naturally as to what are the relations between the results mentioned above. To put it briefly this is the situation.

We will see below that endofunctors in $Class^*$, which are inclusion preserving and uniform on maps, are also set-based, hence the Special Final Coalgebra Theorem does not add anything to the Final Coalgebra Theorem, as far as existence.

Clearly the rôle of AFA , is prominent only in the *Special Final Coalgebra Theorem*. One can easily see in fact that the proof of *Final Coalgebra Theorem* works just as well in any category *Class* determined by the subclasses of a universe of sets, irrespectively of which foundation/anti-foundation axiom it satisfies, if any. Actually, the relative consistency proof of an anti-foundation axiom such as that of X_1 in [FH83], (or that of AFA) is an absolute proof of existence of a final $\mathcal{P}(\ )$-coalgebra. In a sense, it is the "master" proof of all Final Coalgebra Theorems. The practical value of the *Special Final Coalgebra Theorem* is precisely that it is a *universal* Final Coalgebra Theorem. Once we have proved that such a final coalgebra exists, we do not need to redo that proof for all functors.

We can derive the existence of a final coalgebra for functors which are uniform on maps, and thus can be fully described set-theoretically, from the existence of a final coalgebra for $\mathcal{P}(\ )$. For

Barr-Rutten's existence theorem can be viewed as a generalization of the Final Coalgebra Theorem, in that it generalizes the notion of set-based functor to other cardinalities besides Ord; thus giving more information on the cardinality of the final coalgebra.

In this thesis we have decided to use categories of *hypersets* such as $Class^*(U)$ and $\mathcal{HC}^*_\kappa(U)$ (see Chapter 3) and to use a sharpened version of the "Special Final Coalgebra Theorem" (see below), which allows for a fine control on the cardinality of the final coalgebra.

Here are some of the considerations that led us to this decision.

The goal of Final Semantics is to provide rigorous formal semantics and principled foundations to various techniques for reasoning on programs, while keeping the mathematical overhead as low as possible. In line with this inspiring idea, we tried to use, or at least explore to what extent one can use, as basic categories, natural purely set-theoretic categories based on $Class$. We feel in fact that one should try to express the constructions in the most direct way, trying to eliminate all unnecessary encodings, which can obscure the nature of the interpretation function. In view of the applications, one should also try to avoid "sweeping under the rug" many details for the sake of categorical elegance.

Hence, contrary to the position advocated by Barr, such "exotica as non-wellfounded set theory" ([Bar93]), is welcome in Final Semantics, in that it allows to avoid using elaborate indirect encodings or cumbersome quotients. Using hypersets one operates, once and for all, a unique quotient operation at the outset in the consistency proof. Of course, all constructions up-to-bijective maps, such as those used in final semantics, which can be carried out in a non-wellfounded universe can be carried out also in a wellfounded universe. It would be nice to state a precise conservative extension result here.

We have a full spectrum of set theoretical categories to take as basic categories for final semantics. At one end we have the most succinct, $Card$ ($CARD$). At the other end we have the most "exotic" $Class^*(U)$. The other categories mentioned in Section 3.2 lie in between. We are in favour of using urelements (atoms) explicitly, rather than using them as a *façon de parler* and sweeping under the carpet some encoding. This latter attitude makes rigorous proofs more difficult.

Often it is convenient to have a sharp control on the cardinality of the final coalgebra. So it is important to consider also smaller categories and not only super-large categories. Interesting categories in this respect are the categories $\mathcal{HC}^*_\kappa$ and $\mathcal{HC}^*_\kappa(U)$.

Using the existence theorems in the literature, one can claim that all categories above, ranging from $Card$ to $Class^*(U)$, can be used for providing final semantics. We think that the settings provided by $Class^*(U)$ and $\mathcal{HC}^*_\kappa(U)$ are the most transparent.

In discussing functors the following proposition is important.

Proposition 3.5.2 *Let $F : Class^*(U) \longrightarrow Class^*(U)$ be a functor which is inclusion preserving and uniform on maps, then*

$$F(A) = \bigcup_{a \in \mathcal{P}(A)} F(a).$$

Hence F is set-based.

Proof Since F is inclusion preserving, we have immediately that it is also monotone. Moreover, since it is also uniform on maps, we have that, for all $f : A \longrightarrow B$, irrespectively from the codomain, $F(f) = \widehat{f} \circ \phi_A$.
So, assume $b \in F(a)$ and put $a = \{v \mid x_v \in TC(\phi_A(b)) \cap X_A\}$. Take $\pi : A \longrightarrow a$ to be a projection. Now, since $\pi_{|a} = id_{|a}$, we have that $F(\pi)(b) = \widehat{\pi}(\phi_A(b)) = \widehat{id_A}(\phi_A(b)) = F(id)(b) = b$. $\square$

Although more general, the *Final Coalgebra Theorem* makes use of a general colimit construction, which does not yield a straightforward definition of the final F-coalgebra. The final F-coalgebra comes in the form of a cumbersome quotient. On the other hand, the *Special Final Coalgebra Theorem*, exploiting hypersets, has the virtue of defining the final F-coalgebra as the maximal fixed point of the set theoretic operator underlying the functor F. In final semantics, the advantage of having an independent definition of the final F-coalgebra is well worth the limitation to functors which are uniform on maps and inclusion preserving. One can in fact even go as far as claiming (of course informally) that all functors which arise from operational semantics are uniform on maps.

Sometimes, the Special Final Coalgebra Theorem can be used also to compute final coalgebras for functors which are not uniform on maps (e.g. the identity functor), or not inclusion preserving, as can be seen from the following trivial proposition

Proposition 3.5.3 *Let $t : F \Longrightarrow G$ be a natural transformation between the functors $F, G : Class^*(U) \longrightarrow Class^*(U)$, and let F be uniform on maps and inclusion preserving. Let (U_F, id_{U_F}) be the final F-coalgebra. If, for all A, t_A is an isomorphism, then (U_F, t_{U_F}) is a final G-coalgebra.*

The above proposition immediately provides a final coalgebra for the identity functor, one just needs to calculate, say, the final coalgebra of the standard, uniform on maps, functor $USC(A) = \{\{a\} \mid a \in A\}$.

Finally, we give the most important result of this section, i.e. a sharpening, in the way of cardinality, of the Special Final Coalgebra Theorem. This result, together with its restriction to the category $\mathcal{HC}^*_\kappa(U)$, will be the theorem that we shall use throughout this thesis for asserting the existence of final coalgebras. First of all we need the following definition:

Definition 3.5.4 *Let $\mathcal{C} \in \{Class^*(U), \mathcal{HC}^*_\kappa(U)\}$. $F : \mathcal{C} \to \mathcal{C}$ is* κ-boundedly uniform on maps *if F is uniform on maps and moreover, for all A and $a \in F(A)$,*

$$| \bigcap \{C \subseteq (\mathcal{P}(C) \cup U \cup X_A) \mid (a \in C) \ \wedge \ (\phi_A(a) \subseteq C) \ \wedge \ (x_b \in C \Longrightarrow \phi_A(b) \subseteq C)\}| < \kappa \qquad (3.1)$$

Theorem 3.5.5 *Let $F : Class^*(U) \longrightarrow Class^*(U)$ be a functor that is inclusion preserving and κ-boundedly uniform on maps. Let J_F be the maximal fixed point of the class-operator underlying the functor F. Then (J_F, id_{J_F}), is a final F-coalgebra and moreover $J_F \subseteq HC_\kappa(U)$.*

Proof The proof of the first part is essentially a rephrasing of the proof of Aczel's *Special Final Coalgebra Theorem.* To show that $J_F \subseteq HC_\kappa(U)$, we shall apply Proposition 2.2.11 of Chapter 2.
Since $F(J_F) = J_F$, we have that $id_{J_F} = \widehat{id_{J_F}} \circ \phi_{J_F}$. Let $U_0 = U \setminus X_{J_F}$. We put $TC(\phi^+_{J_F}(J_F)) \setminus U_0 = A$ and we define $f : A \longrightarrow \mathcal{P}(A)$ as follows:

$$f(a) = \begin{cases} \phi_{J_F}(j) & if \ a = x_j \\ a & otherwise\ . \end{cases}$$

By Axiom FCU we have that there exists a unique function $h : A \longrightarrow V$ such that, for all $a \in A$, $h(a) = \{h(b) \mid b \in f(a)\} \ \cup \ (f(a) \cap U_0)$. Let $g_1 : X_{J_F} \longrightarrow J_F$ be such that $g_1(x_j) = j$, and put $g = id_{(TC(J_F)\setminus U_0)} \cup g_1$. One can easily check that

$$g(a) = \begin{cases} j & if \ a = x_j \\ \{g(b) \mid b \in a\} \ \cup \ (a \cap U_0) & otherwise\ . \end{cases} .$$

Using the above definitions and the fact that $id_{J_F} = \widehat{id_{J_F}} \circ \phi_{J_F}$, one can check straightforwardly that also g satisfies the property that, for all $a \in A$, $g(a) = \{g(b) \mid b \in f(a)\} \ \cup \ (f(a) \cap U_0)$. Hence $g = h$. Condition 3.1 amounts to the fact that, for all $a \in A$, $| TC_f(a) | < \kappa$; hence, using Proposition 2.2.11, we have that $J_F \subseteq h^+(A) \subseteq HC_\kappa(U)$. $\square$

Notice that the condition 3.1 in Definition 3.5.4 above is satisfied for all A and $a \in F(A)$, if $|TC(\phi_A(a))| < \kappa$ and κ is regular.

A careful analysis of what has been used in the proof of the previous theorem allows to prove the following

Theorem 3.5.6 *i) Let $F : \mathcal{HC}^*_\kappa(U) \longrightarrow \mathcal{HC}^*_\kappa(U)$ be a functor which is inclusion preserving and κ-boundedly uniform on maps. Let J_F be the maximal fixed point of the operator underlying the functor F. Then (J_F, id_{J_F}), is a final F-coalgebra, and moreover $J_F \subseteq HC_\kappa$.*
ii) If F is the restriction of a functor $F' : Class^*(U) \longrightarrow Class^*(U)$, which is inclusion preserving and κ-boundedly uniform on maps, then the final F-coalgebra coincides with the final F'-coalgebra.

Chapter 4

Towards Logics for Circular Objects

In the previous chapters we have discussed infinite and circular objects from a purely semantical (set-theoretical or categorical) perspective, using the language of ordinary mathematics. However it is convenient, if not indispensable for applications to software validation, to be able to place such reasonings within logical systems, amenable to formal proof check.

Various formal systems for establishing properties of infinite and circular objects have been proposed in the literature. One can mention:
- the various Logics for Computable Functions based on Domain theory, which make use of *partial elements*, see e.g. [Plo85];
- the various framework theories such as Talcott's IOCC [Tal90], based on Feferman's theory of operations and classes [Fef90], or Paulson's Isabelle [Pau93], based on Higher Order Logic, where the theory of maximal fixed points can be explicitly formalized;
- the various systems of Intuitionistic Type Theories such as Constable's NUPRL, based on Martin-Löf Type Theory [Mar84], where one can introduce special expressions and proof rules for dealing with infinite objects [MPC86].

An extremely interesting technique for dealing with coinductive types in Intuitionistic Type Theories was introduced by Coquand in his seminal paper, [Coq94]. This technique, originally developed for predicative systems, was later extended by Giménez to impredicative systems, [Gim95]. In a sense, one can say that Coquand defines a system which can make sense of *infinitely regressive proofs*, by means of the *guarded induction principle*. This approach is particularly appealing, because in such an intuitionistic setting, where proofs are first-class objects of discourse, proofs by coinduction are viewed as infinite objects on a par with any other co-inductively defined object, such as a stream, or a coinductive function.

In this chapter we explore the possibility of axiomatizing à la Gentzen the greatest bisimulation relations on *syntactical* F-coalgebras of terms for denoting objects of coinductive types. We work out in detail only two examples: that of terms for denoting *infinite regular binary trees* and that of *regular non-deterministic processes*. *Regular* objects are objects which have only a *finite* number of non-isomorphic subobjects. To put it differently, we axiomatize the equivalence induced by the final semantics.

Alternative axiomatizations of the notion of strong bisimulation have been proposed by Milner in [Mil84].

Our work is probably related to [Coq94]. It is inspired by [BH97], where a coinductive axiomatization of the type (in)equality for a simple first order language of regular recursive types is provided. The types considered in [BH97] are terms for denoting regular binary trees. They can be endowed with a coalgebra structure for a suitable functor F^{BT}. In our first example we essentially reformulate the system of [BH97], by presenting a system $\mathcal{S}^{BT}$ à la Gentzen, which axiomatizes coinductively the largest F^{BT}-bisimulation relation on binary trees. We give proofs of correctness and completeness of this system alternative to those in [BH97]. The correctness is proved by coinduction, i.e. by showing that the relation axiomatized by $\mathcal{S}^{BT}$ is a bisimulation. The completeness proof exploits the fact that the objects that we consider are regular.

We conjecture that these results for binary trees can be generalized to a suitable class of functors and syntactical coalgebras for denoting *regular objects*. To this end we present in this chapter also an axiomatization of the largest bisimulation over a coalgebra of terms for simple non-deterministic processes.

4.1 Axiomatizing Binary Trees

In this section we discuss a coinductive axiomatization à la Gentzen of the bisimulation equivalence on *binary tree terms* viewed as a coalgebra for the functor $F^{BT} : Class^*(U) \to Class^*(U)$ defined by

$$F^{BT}(X) = (X \times X) + C \ ,$$

where C is a finite non empty set of constants.

Definition 4.1.1 • *The* binary tree terms *(trees for short) are defined as follows:*

$$(T_{BT} \ni)\ p ::= \ c \ \mid \ x \ \mid < p_1, p_2 > \mid \ \mu x. < p_1, p_2 > \ ,$$

where $c \in C$, $x \in TVar$, for $TVar$ infinite set of tree terms variables.

Let T^0_{BT} denote the set of closed tree terms.

- *The set of closed tree terms can be endowed with an F^{BT}-coalgebra structure as follows. Let $\alpha_{T^0_{BT}} : T^0_{BT} \to F^{BT}(T^0_{BT})$ be the function defined by*

$$\alpha_{T^0_{BT}}(p) = \begin{cases} c & \textit{if } p = c \in C \\ (p_1, p_2) & \textit{if } p =< p_1, p_2 > \\ (p_1[p/x], p_2[p/x]) & \textit{if } p = \mu x. < p_1, p_2 > \ . \end{cases}$$

The intended denotations of the above terms are regular binary trees. The equivalence induced by this intended interpretation, which is also the final semantics, is the largest F^{BT}-bisimulation on the F^{BT}-coalgebra $(T^0_{BT}, \alpha_{T^0_{BT}})$. An F^{BT}-bisimulation on $(T^0_{BT}, \alpha_{T^0_{BT}})$ can be characterized as follows:

Definition 4.1.2 *An F^{BT}-bisimulation on the F^{BT}-coalgebra $(T^0_{BT}, \alpha_{T^0_{BT}})$ is a relation $\mathcal{R} \subseteq T^0_{BT} \times T^0_{BT}$ such that:*

$p \,\mathcal{R}\, q \implies$

- $p = c = q \;\;\vee$
- $(p = < p_1, p_2 > \;\wedge\; q = < q_1, q_2 > \;\wedge\; p_1 \,\mathcal{R}\, q_1 \;\wedge\; p_2 \,\mathcal{R}\, q_2) \;\;\vee$
- $(p = < p_1, p_2 > \;\wedge\; q = \mu x. < q_1, q_2 > \;\wedge\; p_1 \,\mathcal{R}\, q_1[q/x] \;\wedge\; p_2 \,\mathcal{R}\, q_2[q/x]) \;\;\vee$
- $(p = \mu x. < p_1, p_2 > \;\wedge\; q = \mu x. < q_1, q_2 > \;\wedge\; p_1[p/x] \,\mathcal{R}\, q_1[q/x] \;\wedge\; p_2[p/x] \,\mathcal{R}\, q_2[q/x]) \;\;\vee$
- $(p = \mu x. < p_1, p_2 > \;\wedge\; q = < q_1, q_2 > \;\wedge\; p_1[p/x] \,\mathcal{R}\, q_1 \;\wedge\; p_2[p/x] \,\mathcal{R}\, q_2)$.

We denote by $\sim^{BT}$ the greatest bisimulation on $T^0_{BT} \times T^0_{BT}$.

Now we introduce the system à la Gentzen $\mathcal{S}_{BT}$ for axiomatizing the relation $\sim \subseteq T_{BT} \times T_{BT}$. We will show that $\sim$ restricted to closed terms is exactly the greatest F^{BT}-bisimulation on the F^{BT}-coalgebra of closed binary tree terms.

Definition 4.1.3 ($\mathcal{S}_{BT}$) *The system $\mathcal{S}_{BT}$ axiomatizes judgements of the shape $p \sim q$, where $p, q \in T_{BT}$. Let $\Gamma, \Gamma' \in Env_{BT}$ be finite sets of judgements of the shape $p_i \sim q_i$, for $p_i, q_i \in T_{BT}$. The system $\mathcal{S}_{BT}$ consists of the following rules:*

$$\Gamma \vdash_{BT} p \sim p \quad (refl) \qquad \frac{\Gamma \vdash_{BT} p_1 \sim p_2}{\Gamma \vdash_{BT} p_2 \sim p_1} \quad (symm)$$

$$\frac{\Gamma \vdash_{BT} p_1 \sim p_2 \quad \Gamma \vdash_{BT} p_2 \sim p_3}{\Gamma \vdash_{BT} p_1 \sim p_3} \quad (trans)$$

$$\Gamma, p \sim p', \Gamma' \vdash_{BT} p \sim p' \quad (hyp) \qquad\qquad \Gamma \vdash_{BT} \mu x.p \sim p[\mu x.p/x] \quad (\mu)$$

$$\frac{\{\Gamma, < p_1, p_2 > \sim < p'_1, p'_2 > \vdash_{BT} p_i \sim p'_i\}_{i=1,2}}{\Gamma \vdash_{BT} < p_1, p_2 > \sim < p'_1, p'_2 >} \quad (pair)$$

Notice the "coinductive" nature of the rule $(pair)$: for establishing the equivalence $\sim$ between terms of the shape $< p_1, p_2 >$ and $< p'_1, p'_2 >$, we can assume the judgement that we want to prove, i.e. $< p_1, p_2 > \sim < p'_1, p'_2 >$, in the premises of the rule. The system $\mathcal{S}_{BT}$ is the version for equality of the system defined in [BH97] for subtyping relation. We prove now correctness and completeness of the system $\mathcal{S}_{BT}$. The proofs are alternative to those in [BH97]. In particular, we prove that $\vdash_{BT} p \sim q$ if and only if the pair $< p, q >$ belongs to an F^{BT}-bisimulation on the F^{BT}-coalgebra $(T^0_{BT}, \alpha_{T^0_{BT}})$.

It is easy to check that a *Weakening Lemma* holds for the system $\mathcal{S}_{BT}$:

Lemma 4.1.4 (Weakening) *If* $\Gamma \vdash_{BT} p \sim q$*, then, for all* Γ'*, also* $\Gamma, \Gamma' \vdash_{BT} p \sim q$ *and* $\Gamma', \Gamma \vdash_{BT} p \sim q$.

In order to prove correctness of $\mathcal{S}_{BT}$ we need to introduce the notion of *closed sequent*:

Definition 4.1.5 *A sequent* $\Gamma \vdash_{BT} p \sim q$*, where* $\Gamma \in Env_{BT}$ *is the list of judgements* $p_1 \sim q_1, \ldots, p_n \sim q_n$*, for* $n \geq 0$*, is* closed *in* $\mathcal{S}_{BT}$*, when it is derivable in* $\mathcal{S}_{BT}$*, and moreover there exist derivations in* $\mathcal{S}_{BT}$ *of the sequents* $p_1 \sim q_1, \ldots, p_{i-1} \sim q_{i-1} \vdash_{BT} p_i \sim q_i$*, for all* $i = 1, \ldots, n$.

Lemma 4.1.6 *If* $\Gamma \vdash_{BT} p \sim q$ *and* $\Gamma' \vdash_{BT} p' \sim q'$ *are closed, then also* $\Gamma, \Gamma' \vdash_{BT} p \sim q$ *and* $\Gamma, \Gamma' \vdash_{BT} p' \sim q'$ *are closed.*

Theorem 4.1.7 (Correctness of $\mathcal{S}_{BT}$) *For all* $p, q \in T^0_{BT}$,

$$\vdash_{BT} p \sim q \implies p \sim^{BT} q \ .$$

Proof We show that the following relation is a bisimulation on $T^0_{BT} \times T^0_{BT}$: $\mathcal{R} = \{(p, q) \in T^0_{BT} \times T^0_{BT} \mid \exists \Gamma.\ \Gamma \vdash_{BT} p \sim q \text{ closed in } \mathcal{S}_{BT}\}$.
We prove this by induction on the sum of the lengths of the derivations τ and τ_i's, where τ denotes the derivation of $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p \sim q$ and τ_i denotes the derivation of $p_1 \sim q_1, \ldots, p_{i-1} \sim q_{i-1} \vdash_{BT} p_i \sim q_i$, for $i = 1, \ldots, n$.
Base Case: $\forall i.\ l(\tau_i) = 0 \wedge l(\tau) = 1$. The only rule applied in τ is $(refl)$ or (μ). The thesis follows since the identity relation is included in $\mathcal{R}$.
Induction Step: we proceed by analyzing the last rule applied in τ. If the last rule is $(refl)$ or (μ), then again the thesis follows from the fact that the identity relation is included in $\mathcal{R}$. If the last rule is $(symm)$, then the thesis is immediate by induction hypothesis, using the fact that $\mathcal{R}$ is symmetric. If the last rule is $(trans)$, i.e. $\dfrac{\Gamma \vdash_{BT} p \sim p' \quad \Gamma \vdash_{BT} p' \sim q}{\Gamma \vdash_{BT} p \sim q}$, then there are various cases, according to the structure of p, p', q. If $p' = c$, then by induction hypothesis also $p, q = c$. If $p = < p_1, p_2 >$, $p' = < p'_1, p'_2 >$, and $q = < q_1, q_2 >$, then, by induction hypothesis, there exist $\Gamma_1, \Gamma_2, \Gamma'_1, \Gamma'_2$ such that $\Gamma_1 \vdash_{BT} p_1 \sim p'_1$, $\Gamma_2 \vdash_{BT} p_2 \sim p'_2$, $\Gamma'_1 \vdash_{BT} p'_1 \sim q_1$, and $\Gamma'_2 \vdash_{BT} p'_2 \sim q_2$ are closed in $\mathcal{S}_{BT}$. Then the thesis follows from Lemma 4.1.6, using rule $(trans)$. If $p = < p_1, p_2 >$, $p' = \mu x. < p'_1, p'_2 >$, and $q = < q_1, q_2 >$, then, by induction hypothesis, there exist $\Gamma_1, \Gamma_2, \Gamma'_1, \Gamma'_2$ such that $\Gamma_1 \vdash_{BT} p_1 \sim p'_1[p'/x]$, $\Gamma_2 \vdash_{BT} p_2 \sim p'_2[p'/x]$, $\Gamma'_1 \vdash_{BT} p'_1[p'/x] \sim q_1$, and $\Gamma'_2 \vdash_{BT} p'_2[p'/x] \sim q_2$ are closed in $\mathcal{S}_{BT}$. Then the thesis follows again from Lemma 4.1.6, using rule $(trans)$. The remaining cases are dealt with similarly. If the last rule in τ is (hyp), i.e. $\tau : \quad \Gamma, p \sim p' \vdash_{BT} p \sim p'$, then, since τ is closed, then also $\Gamma \vdash_{BT} p \sim p'$ is closed in $\mathcal{S}_{BT}$; then the thesis follows by induction hypothesis. Finally, if the last rule of τ is $(pair)$, then the thesis follows immediately, since, if the conclusion of an application of the rule $(pair)$ is closed, then also the premises are closed. $\square$

In order to show the completeness of the system $\mathcal{S}_{BT}$, we need to exploit the regularity of binary trees terms. Namely, we introduce the notion of set of subterms of a given term.

Definition 4.1.8 *Let $p \in T^0_{BT}$. The set of subterms of p, $sub(p)$, is defined as follows:*

- *if $p = c$, then $sub(p) = \{c\}$;*
- *if $p =< p_1, p_2 >$, then $sub(p) = \{p\} \cup sub(p_1) \cup sub(p_2)$;*
- *if $p = \mu x. < p_1, p_2 >$, then $sub(p) = \{p\} \cup \{q[p/x] \mid q \in sub(< p_1, p_2 >)\}$.*

Lemma 4.1.9 *For all $p \in T^0_{BT}$, the set $sub(p)$ is finite.*

Now we are in the position of stating the Completeness Theorem for the system $\mathcal{S}_{BT}$. The proof of this theorem consists in showing that, if two closed binary tree terms p, q are bisimilar, then, since they have only a finite number of subterms, we can build a derivation of $\vdash_{BT} p \sim q$ in a top-down fashion.

Theorem 4.1.10 (Completeness of $\mathcal{S}_{BT}$) *For all $p, q \in T^0_{BT}$,*

$$p \sim^{BT} q \implies \vdash_{BT} p \sim q \quad .$$

Proof We prove that, if $p \sim^{BT} q$, then for all $p_1, \ldots, p_n, p' \in sub(p)$, $q_1, \ldots, q_n, q' \in sub(q)$ such that $\forall i = 1, \ldots, n.\ p_i \sim^{BT} q_i$, and $p' \sim^{BT} q'$, there exists a derivation of $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p' \sim q'$. Suppose by contradiction that $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p' \sim q'$ is not derivable. Then we show that there exists an infinite sequence of distinct pairs of processes (p_i, q_i) such that $p_i \sim^{BT} q_i$ and $p_i \in sub(p)$, $q_i \in sub(q)$, which is clearly impossible because $sub(p)$ and $sub(q)$ are finite.. In fact, if $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p' \sim q'$ is not derivable, then we show that a sequent of the following shape is not derivable: $p_1 \sim q_1, \ldots, p_n \sim q_n, p_{n+1} \sim q_{n+1} \vdash_{BT} p'' \sim q''$, for some $p_{n+1}, p'' \in sub(p)$, $q_{n+1}, q'' \in sub(q)$, such that $p_{n+1} \sim^{BT} q_{n+1}$, $p'' \sim^{BT} q''$, and the hypothesis $p_{n+1} \sim q_{n+1}$ is new, in the sense that it does not appear among $p_1 \sim q_1, \ldots, p_n \sim q_n$. This latter fact is proved by case analysis, according to the structure of p' and q'. We analyze only the case $p' = \mu x. < p'_1, p'_2 >$ and $q' =< q'_1, q'_2 >$, the remaining cases being dealt with similarly. Let $\Gamma = p_1 \sim q_1, \ldots, p_n \sim q_n$. Suppose by contradiction that the sequents $\{\Gamma, < p'_1[p'/x], p'_2[p'/x] > \sim q' \vdash_{BT} p'_j[p'/x] \sim q'_j\}_{j=1,2}$ are derivable. Then the following would be a proof of the sequent $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p' \sim q'$ (using rules $(\mu), (pair), (trans)$):

$$\frac{\{\Gamma, < p'_1[p'/x], p'_2[p'/x] > \sim q' \vdash_{BT} p'_j[p'/x] \sim q'_j\}_{j=1,2}}{\Gamma \vdash_{BT} < p'_1[p'/x], p'_2[p'/x] > \sim q'} \ (pair)$$

$$\frac{\Gamma \vdash_{BT} p' \sim < p'_1[p'/x], p'_2[p'/x] > \quad \Gamma \vdash_{BT} < p'_1[p'/x], p'_2[p'/x] > \sim q'}{\Gamma \vdash_{BT} p' \sim q'} \ (trans)$$

where the pair $< p'_1[p'/x], p'_2[p'/x] > \sim q'$ is new in the righhand premises of the instance of the rule $(pair)$, otherwise we would already have a proof of the sequent $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{BT} p' \sim q'$, and the lefthand premise in the instance of the rule $(trans)$ is an instance of the axiom (μ). Then for some j, $\Gamma, < p'_1[p'/x], p'_2[p'/x] > \sim q' \vdash_{BT} p'_j[p'/x] \sim q'_j$ is not derivable, and

Γ does not contain already the hypothesis $< p'_1[p'/x], p'_2[p'/x] >\sim q'$, since otherwise $\Gamma \vdash_{BT} < p'_1[p'/x], p'_2[p'/x] >\sim q'$ would be an instance of the axiom (hyp). Hence $(< p'_1[p'/x], p'_2[p'/x] >, < q'_1, q'_2 >)$ is a new pair such that $< p'_1[p'/x], p'_2[p'/x] >\sim^{BT} < q'_1, q'_2 >$, and $p'_1[p'/x] \in sub(p')$, $< q'_1, q'_2 >\in sub(q')$. $\square$

4.2 Axiomatizing non-Deterministic Processes

In this section, we consider a language for simple *non-deterministic process terms*, and we present a system à la Gentzen for axiomatizing *coinductively* the (strong) bisimulation equivalence on it. These process terms can be viewed as a coalgebra for the functor $F^{\Sigma} : Class^*(U) \to Class^*(U)$ defined as follows:

$$F^{\Sigma}(X) = \mathcal{P}_{fin}(X) \ .$$

The language of simple non-deterministic processes which we consider is a fragment of CCS with only one atomic action. The terms of this language and its operational semantics are defined as follows:

Definition 4.2.1 • *The simple non-deterministic process terms (processes for short) are defined as follows:*

$$(T_{\Sigma} \ni) \ p ::= \ nil \ \mid \ \{p\} \ \mid \ p + p \ \mid \ x \ \mid \ \mu x.p \ ,$$

where $x \in TVar$, for $TVar$ infinite set of process terms variables, and $\mu x.p$ is guarded, i.e. any occurrence of the variable x in p occurs in the scope of a $\{ \ \}$-operator.
Let T^0_{Σ} denote the set of closed processes, and let $\Sigma_{i \in I} p_i$ be an abbreviation for $(p_1 + \ldots) + p_n$, for $I = \{1, \ldots, n\}$.

- *Let $\to \subseteq T^0_{\Sigma} \times T^0_{\Sigma}$ be the least relation closed under the following rules:*

$$\{p\} \to p \quad \{\} \qquad\qquad \frac{p[\mu x.p/x] \to p'}{\mu x.p \to p'} \ \mu$$

$$\frac{p_1 \to p'_1}{p_1 + p_2 \to p'_1} \ +_l \qquad\qquad \frac{p_2 \to p'_2}{p_1 + p_2 \to p'_2} \ +_r$$

The notion of equivalence on $T^0_{\Sigma} \times T^0_{\Sigma}$ that we will axiomatize is the *strong bisimulation equivalence* (see Chapter 5, Section 5.2, Definition 5.2.6). The strong equivalence on T^0_{Σ} will be denoted by $\sim^{\Sigma}$.

The set of closed processes can be endowed with a structure of F^{Σ}-coalgebra as follows, in such a way that the greatest F^{Σ}-bisimulation is the strong bisimulation equivalence:

Definition 4.2.2 *Let $\alpha_{T^0_{\Sigma}} : T^0_{\Sigma} \to F^{\Sigma}(T^0_{\Sigma})$ be the function defined by*

$$\alpha_{T^0_{\Sigma}}(p) = \{p_1 \mid p \to p_1\} \ .$$

Before introducing the system $\mathcal{S}_\Sigma$, we define the notion of *canonical form* of a process, and we show that all processes have strongly equivalent canonical forms. The notion of canonical form will be useful for proving the completeness of the system $\mathcal{S}_\Sigma$. For technical reasons, we give the definition of canonical form for terms defined by the following grammar:

$$(\overline{T}_\Sigma \ni)\ p ::= \ nil \ \mid \ \{p\} \ \mid \ p+p \ \mid \ \{x\} \ \mid \ \mu x.p \ ,$$

where $\mu x.p$ is guarded.

Definition 4.2.3 (Canonical Form) *Let $p \in \overline{T}_\Sigma$. Let $can(p)$ be defined as follows:*

- *if $p = nil$, then $can(p) = nil$;*
- *if $p = \{p_1\}$, then $can(p) = \{can(p_1)\}$;*
- *if $p = p_1 + p_2$, then*

$$can(p) = \begin{cases} can(p_2) & \textit{if } can(p_1) = nil \\ can(p_1) & \textit{if } can(p_2) = nil \\ can(p_1) + can(p_2) & \textit{otherwise;} \end{cases}$$

- *if $p = \{x\}$, then $can(p) = \{x\}$;*
- *if $p = \mu x.p_1$, then $can(p) = can(p_1)[p/x]$.*

In order to prove Lemma 4.2.5 below, i.e. that a process term is strongly equivalent to its canonical form, we need to extend the notion of bisimulation also to open terms:

Definition 4.2.4 *Let $p, p' \in T_\Sigma$ be such that $fv(p,q) \subseteq \{x_1, \ldots, x_n\}$. We define $p \sim^\Sigma p'$ if and only if, for all $p_1, \ldots, p_n \in (T^\Sigma)^0$, $p[p_1/x_1, \ldots, p_n/x_n] \sim^\Sigma p'[p_1/x_1, \ldots, p_n/x_n]$.*

Lemma 4.2.5 *Let $p \in \overline{T}_\Sigma$. Then $p \sim^\Sigma can(p)$.*

Proof

- We proceed by induction on the structure of $p \in \overline{T}_\Sigma$.
  Base Cases: if $p = nil$ or $p = \{x\}$, then they are already in canonical form.
  Induction Step: if $p = p_1 + p_2$ or $p = \{p_1\}$, then the thesis follows immediately applying the induction hypothesis. If $p = \mu x.p_1$, then by induction hypothesis $p_1 \sim^\Sigma can(p_1)$, and hence, in particular, $p_1[p/x] \sim^\Sigma can(p_1)[p/x]$. Since $p \sim^\Sigma p_1[p/x]$, then, by transitivity, $p \sim^\Sigma can(p_1)[p/x]$.
- By induction on the structure of $p \in \overline{T}_\Sigma$. The only non trivial case is that of $p = \mu x.p_1$. By induction hypothesis, $\vdash_\Sigma p_1 \sim can(p_1)$. Then, using the fact that, $\forall p_1, p_2, p \in \overline{T}_\Sigma$, $\Gamma \vdash_\Sigma p_1 \sim p_2 \ \Rightarrow \ \Gamma \vdash_\Sigma p_1[p/x] \sim p_2[p/x]$ (which can be proved by induction on the derivations), we get $\vdash_\Sigma p_1[p/x] \sim can(p_1)[p/x]$. Hence using rules (μ) and $(trans)$, $\vdash_\Sigma \mu x.p_1 \sim can(p_1)[p/x]$.

- One can easily prove by induction on the structure of $p \in \overline{T}_\Sigma$, $fv(p) \subseteq \{x_1, \ldots, x_n\}$, that, for all $p_1, \ldots, p_n \in T^0_\Sigma$,
$sub(can(p[p_1/x_1, \ldots, p_n/x_n])) = sub(p[p_1/x_1, \ldots, p_n/x_n])$.

□

Notice that, since only guarded recursion is allowed, up to reordering of parentheses in sums of processes, canonical forms have the following shape:

Lemma 4.2.6 *Let $p \in \overline{T}_\Sigma$. Then $can(p)$ is either nil or $\Sigma_{i\in I} p_i$, where, for all $i \in I$, $p_i = \{p'_i\}$, with $p'_i \in T_\Sigma$.*

Now we introduce the system $\mathcal{S}_\Sigma$ for axiomatizing the relation $\sim \subseteq T_\Sigma \times T_\Sigma$. We will show that the relation $\sim$ restricted to closed process terms amounts exactly to the strong equivalence $\sim^\Sigma$.

Definition 4.2.7 ($\mathcal{S}_\Sigma$) *The system $\mathcal{S}_\Sigma$ axiomatizes judgements of the shape $p \sim q$, where $p, q \in T_\Sigma$. Let $\Gamma \in Env_\Sigma$ be a finite list of judgements of the shape $p_i \sim q_i$, for $p_i, q_i \in T_\Sigma$. The system $\mathcal{S}_\Sigma$ consists of the following rules:*

$$\Gamma \vdash_\Sigma p \sim p \quad (refl) \qquad \frac{\Gamma \vdash_\Sigma p_1 \sim p_2}{\Gamma \vdash_\Sigma p_2 \sim p_1} \; symm$$

$$\frac{\Gamma \vdash_\Sigma p_1 \sim p_2 \quad \Gamma \vdash_\Sigma p_2 \sim p_3}{\Gamma \vdash_\Sigma p_1 \sim p_3} \; (trans)$$

$$\Gamma, p \sim p', \Gamma' \vdash_\Sigma p \sim p' \quad (hyp) \qquad \Gamma \vdash_\Sigma \mu x.p \sim p[\mu x.p/x] \quad (\mu)$$

$$\frac{\Gamma \vdash_\Sigma p \sim p'}{\Gamma \vdash_\Sigma \{p\} \sim \{p'\}} \; (\{\ \})$$

$$\frac{\{\Gamma, \Sigma_{i\in I}\{p_i\} \sim \Sigma_{i\in I}\{p'_i\} \vdash_\Sigma p_i \sim p'_i\}_{i\in I,\ I\ finite}}{\Gamma \vdash_\Sigma \Sigma_{i\in I}\{p_i\} \sim \Sigma_{i\in I}\{p'_i\}} \; (\Sigma)$$

$$\Gamma \vdash_\Sigma p + nil \sim p \quad (nil) \qquad \Gamma \vdash_\Sigma p + p \sim p \quad (abs)$$

$$\Gamma \vdash_\Sigma p_1 + p_2 \sim p_2 + p_1 \quad (+comm) \qquad \Gamma \vdash_\Sigma (p_1 + p_2) + p_3 \sim p_1 + (p_2 + p_3) \quad (+ass)$$

The rule Σ in the system $\mathcal{S}_\Sigma$ above is coinductive, since we allow the conclusion to appear as hypothesis in the premises of the rules.

The proof of the correctness of the system $\mathcal{S}_\Sigma$ is similar to the proof of correctness of the system $\mathcal{S}_{BT}$. In particular, we introduce the notion of *closed judgement* for the system $\mathcal{S}_\Sigma$:

Definition 4.2.8 *A sequent $\Gamma \vdash_\Sigma p \sim q$, where $\Gamma \in Env_\Sigma$ is the list $p_1 \sim q_1, \ldots, p_n \sim q_n$, for $n \geq 0$, is* closed *in $\mathcal{S}_\Sigma$, when it is derivable in $\mathcal{S}_\Sigma$, and moreover there exist derivations in $\mathcal{S}_\Sigma$ of $p_1 \sim q_1, \ldots, p_{i-1} \sim q_{i-1} \vdash_\Sigma p_i \sim q_i$, for all $i = 1, \ldots, n$.*

Theorem 4.2.9 (Correctness of $\mathcal{S}_\Sigma$) *For all $p, q \in T^0_\Sigma$,*

$$\vdash_\Sigma p \sim q \implies p \sim^\Sigma q \ .$$

Proof We show that the following relation is a strong bisimulation:
$\mathcal{R}= \{(p,q) \in T^0_\Sigma \times T^0_\Sigma \mid \exists\Gamma.\ \Gamma \vdash_\Sigma p \sim q$ is closed in $\mathcal{S}_\Sigma\}$.
We prove this by induction on the sum of the length of the derivations τ and τ_i's, where τ denotes the derivation of $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_\Sigma p \sim q$ and τ_i denotes the derivation of $p_1 \sim q_1, \ldots, p_{i-1} \sim q_{i-1} \vdash_\Sigma p_i \sim q_i$, for $i = 1, \ldots, n$.
Base Case: $\forall i.\ l(\tau_i) = 0 \ \wedge \ l(\tau) = 1$. The only rule applied in τ is $(refl)$, (μ), (nil), (abs), $(+comm)$ or $(+ass)$. The thesis follows since the identity relation is included in $\mathcal{R}$.
Induction Step: we proceed by analyzing the last rule applied in τ. If the last rule is $(\{\ \})$, then the thesis is immediate. If the last rule is $(refl)$, (μ), (nil), (abs), $(+comm)$ or $(+ass)$, then again the thesis follows from the fact that the identity relation is included in $\mathcal{R}$. If the last rule is $(symm)$ or (hyp), then the thesis is immediate by induction hypothesis. If the last rule is $(trans)$, i.e. $\dfrac{\Gamma \vdash_\Sigma p \sim r \quad \Gamma \vdash_\Sigma r \sim q}{\Gamma \vdash_\Sigma p \sim q}$, then, if $\exists p'.\ p \to p'$, then by induction hypothesis $\exists r', q'.\ r \to r' \ \wedge \ q \to q'$, and $\exists \Gamma', \Gamma''$ such that $\Gamma' \vdash_\Sigma p' \sim r'$ and $\Gamma'' \vdash_\Sigma r' \sim q'$ are closed. Hence, by rule $(trans)$, also $\Gamma', \Gamma'' \vdash_\Sigma p' \sim q'$ is closed. Finally, if the last rule of τ is Σ, then the thesis follows immediately, since, if the conclusion of an application of the rule Σ is a closed sequent, then also the premises are closed sequents. $\square$

The proof of completeness of the system $\mathcal{S}_\Sigma$ exploits the fact that the terms in T_Σ are regular, i.e. they have only a finite number of subterms.

Definition 4.2.10 *Let $p \in T_\Sigma$. The set of subprocesses of p, $sub(p)$, is defined as follows:*

- *if $p \equiv x$, then $sub(p) = \{x\}$;*
- *if $p \equiv nil$, then $sub(p) = \{nil\}$;*
- *if $p \equiv \{p_1\}$, then $sub(p) = \{p\} \cup sub(p_1)$;*
- *if $p \equiv p_1 + p_2$, then $sub(p) = \{p\} \cup sub(p_1) \cup sub(p_2)$;*
- *if $p \equiv \mu x.p$, then $sub(p) = \{p\} \cup \{q[p/x] \mid q \in sub(p)\}$.*

Lemma 4.2.11 *For all $p \in T_\Sigma$, the set $sub(p)$ is finite.*

The following lemma is proved by induction on the structure of p:

Lemma 4.2.12 *Let $p \in \overline{T}_\Sigma$. Then*

1. *$\vdash_\Sigma p \sim can(p)$ is derivable in $\mathcal{S}_\Sigma$.*
2. *$sub(can(p)) = sub(p)$.*

Theorem 4.2.13 (Completeness of $\mathcal{S}_\Sigma$) *For all $p, q \in T^0_\Sigma$,*

$$p \sim^\Sigma q \implies \vdash_\Sigma p \sim q \quad .$$

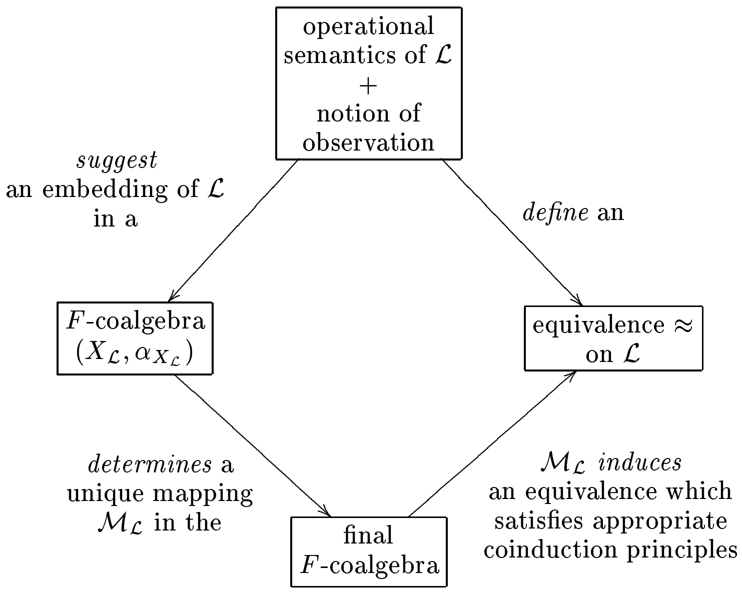
Proof As in the proof of Theorem 4.1.10, we prove that, if $p \sim^{\Sigma} q$, then for all $p_1, \ldots, p_n, p' \in sub(p)$, $q_1, \ldots, q_n, q' \in sub(q)$ such that $\forall i = 1, \ldots, n.\ p_i \sim^{\Sigma} q_i$, and $p' \sim^{\Sigma} q'$, there exists a derivation of $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{\Sigma} p' \sim q'$. Suppose by contradiction that the sequent $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{\Sigma} p' \sim q'$ is not derivable. Then we show that there exist an infinite sequence of distinct pairs of processes (p_i, q_i) such that $p_i \sim^{\Sigma} q_i$ and $p_i \in sub(p)$, $q_i \in sub(q)$. In fact, if $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{\Sigma} p' \sim q'$ is not derivable, then we show that a sequent of the following shape is not derivable: $p_1 \sim q_1, \ldots, p_n \sim q_n, p_{n+1} \sim q_{n+1} \vdash_{\Sigma} p'' \sim q''$, for some $p_{n+1}, p'' \in sub(p)$, $q_{n+1}, q'' \in sub(q)$, such that $p_{n+1} \sim^{\Sigma} q_{n+1}$, $p'' \sim^{\Sigma} q''$, and the hypothesis $p_{n+1} \sim q_{n+1}$ is new, in the sense that it does not appear among $p_1 \sim q_1, \ldots, p_n \sim q_n$. Suppose $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{\Sigma} p' \sim q'$ is not derivable. Then, by Lemmata 4.2.5 and 4.2.12, we can assume that p' and q' are canonical forms, i.e. $p' = \Sigma_{i \in I}\{p'_i\}$ and $q' = \Sigma_{j \in J}\{q'_j\}$. W.l.o.g., we consider $I = J$. The hypothesis $\Sigma_{i \in I}\{p'_i\} \sim \Sigma_{i \in I}\{q'_i\}$ does not appear among the hypotheses $p_1 \sim q_1, \ldots, p_n \sim q_n$, since otherwise the sequent $p_1 \sim q_1, \ldots, p_n \sim q_n \vdash_{\Sigma} p' \sim q'$ would be an instance of the axiom (hyp). But then there exists $i \in I$ such that the sequent $p_1 \sim q_1, \ldots, p_n \sim q_n, \Sigma_{i \in I}\{p'_i\} \sim \Sigma_{i \in I}\{q'_i\} \vdash_{\Sigma} p' \sim q'$ is not derivable, with $(\Sigma_{i \in I}\{p'_i\}, \Sigma_{i \in I}\{q'_i\})$ a new pair such that $\Sigma_{i \in I}\{p'_i\} \sim^{\Sigma} \Sigma_{i \in I}\{q'_i\}$, and $\Sigma_{i \in I}\{p'_i\} \in sub(p')$, $\Sigma_{i \in I}\{q'_i\} \in sub(q')$. $\square$

Finally, we point out the interesting issue of analyzing the relations between the systems à la Gentzen $\mathcal{S}_{BT}$ and $\mathcal{S}_{\Sigma}$, discussed in this chapter, with process algebraic equational characterizations of the corresponding bisimulation equivalences, in the line of Milner's equational theories.

Part II

Final Semantics for Programming Languages

The *final semantics paradigm* consists in representing the operational semantics of a programming language as a *syntactical F-coalgebra*, for a suitable functor F, in such a way that the appropriate *behavioural equivalence* (*observational equivalence*) can be characterized as the equivalence induced on terms of the language by the *unique morphism* from the syntactical coalgebra to the *final F-coalgebra*. If the functor F is "well-behaved" (see Chapter 3, Theorem 3.2.6), then the equivalence induced by the final semantics has a *coinductive characterization*. Hence, in particular, the final semantics paradigm validates *coinductive proofs techniques* for establishing behavioural equivalences. In some cases, it can even suggest different and new forms of coinduction principles, by representing the operational semantics of a language as coalgebra for different functors (see e.g. Chapter 6, Section 6.2.2, and Chapter 7, Section 7.2). The final semantics paradigm can be summarized by the following diagram. Let $\mathcal{L}$ be a language:



The above diagram is intended to "commute". It must be stressed that the structure of the functor F and the nature of $X_{\mathcal{L}}$, the carrier of the coalgebra, in which $\mathcal{L}$ is embedded, are by no means determined uniquely. The crucial decision when giving final semantics lies here.

The final semantics paradigm was originally introduced for CCS-like languages by Aczel in [Acz88] (see also [Acz93, RT93, RT94, Tur96]). In this part of the thesis we explore to what extent the final semantics paradigm can be applied to various notions of equivalences for various classes of languages.

We feel that this range of programming equivalences provides a sophisticated benchmark for a semantical methodology. If final semantics can explain the characteristics of these programming languages, then it can be claimed to be sufficiently general. To this end, in this part, we start by discussing various kinds of equivalences on *process algebras*, many of which have not yet received a final description. These languages are essentially "first order" and so they can be neatly explained using *monotone* functors. In order to see the applicability of final semantics to languages with higher order features and binding operators, we discuss in detail *imperative concurrent languages* with higher order assignment and communication, *functional languages*, and *calculi for mobile processes.* We shall see that it is possible to explain, using *syntactically* based *covariant* functors, features that seem to require *contravariant* functors or sophisticated mathematical settings, such as categories with embedding-projection pairs as morphisms.

As remarked in the Introduction, the final approach to semantics is conceptually appealing because it is quite natural, in that it is *driven by the operational semantics*, as pointed out by Aczel in [Acz93].

Therefore, methodologically it is important to keep the mathematical overhead as low as possible. Consequently, we decide to use, throughout our investigation of final semantics, set-theoretical categories. Consistently with this attitude, we use non-wellfounded sets abiding by an antifoundation axiom, such as X_1 of [FH83], or AFA of [Acz88]. In our view, non-wellfounded sets are not those "exotica" as authors have claimed (see e.g. [Bar93]), but they are convenient mathematical objects, which allow to factor out once and for all complex encodings or cumbersome equivalence relations.

Non-wellfounded sets suggest terse mental images. And, after all, they arise as soon as we take the membership relation to be sufficiently universal.

Many of the final constructions that we give can be worked out also in other categories. But this is not always the case.

There is a very important technical reason for considering naive set-theoretical frameworks rather than more enriched frameworks, such as c.p.o's or metric spaces. There exist in fact observational equivalences, such as weak equivalence on CCS, which are intrinsically *infinitely branching*, and *not even compactly branching*. Processes under weak equivalence are discontinuous, and hence apparently they escape a description in terms of c.p.o.'s or metric spaces (see Chapter 5, Section 5.2). We shall therefore work in $Class^*$ (or in subcategories of it, such as $\mathcal{HC}_\kappa$).

It is interesting to point out that our set-theoretic treatment gives rise to mathematically intriguing issues, when we compare it to more traditional semantical treatments using c.p.o.'s ([GS90]) or metric spaces ([BZ82, AR89]). This is particularly true when dealing with higher order languages, which normally seem to necessitate *contravariant* functors (see e.g. Chapter 6).

We consider a wide spectrum of languages, which comprises *process algebra languages*, *imperative concurrent languages*, *functional languages*, *π-calculus*, and various equivalences thereon.

In this part of the thesis we will see many examples of coinduction proofs, which make use of the final semantics that we introduce, in many different language contexts. Validating coinductive proof techniques is in fact one of the most significant applications of final semantics.

Furthermore, the final paradigm allows also to phrase coinductive proofs in a more principled and uniform way, using the notion of F-coalgebra. In the sequel, we refer to the coinductive proof technique phrased in terms of coalgebras as proof technique *by finality*. It shall consists of establishing a property from the existence of a unique morphism into the final F-coalgebra (see e.g. Chapter 8, Section 8.4.1 for an application of this technique).

The issue of the *compositionality* of final semantics deserves a brief introduction.

Compositionality is one of the crucial properties of *initial semantics* and *denotational semantics*. Often referred to as *referential transparency*, or commutativity of the interpretation function w.r.t substitution. Compositionality ultimately amounts to the fact that the interpretation function induces a *congruence* relation. In particular, we take this as the definition and we say that a semantics $\mathcal{M}_{\mathcal{L}} : \mathcal{L} \to \mathcal{D}$ is *compositional*, if, for all t, t' terms of $\mathcal{L}$,

$$\mathcal{M}_{\mathcal{L}}(t) = \mathcal{M}_{\mathcal{L}}(t') \implies \forall C[\ ].\ \mathcal{M}_{\mathcal{L}}(C[t]) = \mathcal{M}_{\mathcal{L}}(C[t'])\ ,$$

where $C[\ ]$ is a context of $\mathcal{L}$.

Final semantics in general are not compositional. Final semantics induce bisimulations (see Chapter 3), and as Rutten and Turi in [RT94] have brilliantly explained, this is precise *dual* to congruence.

Not being necessarily compositional has both positive and negative sides to it. On one hand, we can capture by finality also equivalences which are not congruences, e.g. weak equivalences (see Chapter 5 and Chapter 8). But, on the other hand, when it is the case, establishing compositionality of final semantics can be difficult and subtle. In this part of the thesis, we shall investigate this issue very carefully and we shall discuss various techniques for proving compositionality of final semantics.

In general, there are two dual approaches to compositionality of the semantics $\mathcal{M}_{\mathcal{L}}$, which we call *a posteriori* and *a priori*, respectively.

In the *a posteriori* approach, one shows directly that the equivalence $\approx$ induced by the final semantics is a *congruence* with respect to the syntactical operators, i.e. $\approx$ is an equivalence relation such that

$$s \approx s' \implies \forall C[\ ].\ C[s] \approx C[s']\ .$$

In the case of process algebras, this is often done by coinduction, so as in the case of imperative concurrent languages (see Chapter 6, Section 6.2.1). In the case of functional languages, showing that $\approx$ is a congruence is extremely difficult. We will present both semantically and syntactically based techniques for showing this (see Chapter 6, Section 6.2.1).

The *a priori* approach consists in defining *a priori* operators by finality coinductively and independently, on the semantical domain, in the line of [Acz88], Chapter 8, and [Acz93], and then showing that

$$\mathcal{M}_{\mathcal{L}}(op_n(s_1, \ldots, s_n)) = \widetilde{op}_n(\mathcal{M}_{\mathcal{L}}(s_1), \ldots, \mathcal{M}_{\mathcal{L}}(s_n))\ ,$$

for all $s_1, \ldots, s_n$, where $\widetilde{op}_n$ is the semantical operator corresponding to the n-ary syntactical operator op_n (see Chapter 6, Section 6.3.3). Showing the above equality is a situation where the coalgebraic framework allows a more direct treatment of bisimulation. Another situation where the appropriate bisimulations can be neatly phrased in the coalgebraic setting is discussed in Chapter 8, Section 8.5.

The a priori approach is rather problematic when dealing with higher order constructors, e.g. in λ-calculus or in π-calculus. It is viable for process algebras, and imperative concurrent languages.

Also in the a posteriori approach one can define a posteriori semantical operators, once the equivalence $\approx$ has been proved to be a congruence. One can do this immediately on the image of the interpretation function $\mathcal{M}_{\mathcal{L}}$. The extension of these operators to all the points of the final coalgebra can be problematic, especially in the case of higher order constructors, such as λ-abstraction or μ-recursion operator. We shall not elaborate on this. In [Rut92], Rutten has introduced the technique of *processes as terms*, which allows to carry out this extension for the case of CCS-like languages. In any case, in the case studies which we will consider, at least on the image of the interpretation function, semantical operators defined *a posteriori* coincide with the semantical operators defined *a priori*. This is the key result for addressing the issue of the relations between final and initial semantics. In [Tur96], Turi has studied this from a general categorical viewpoint, working out the example of process algebras with strong bisimulation equivalence.

This part of the thesis is organized as follows.

In Chapter 5, we discuss the final semantics for *process algebras*. The novelty of this presentation lies in the generality and uniformity of the definition of the final semantics. We consider a general process algebra language, and we define, coinductively, a general notion of branching equivalence, which we call *B-equivalence*, and a general notion of linear equivalence, which we call *L-equivalence*. As we will show, many branching-like equivalences are special instances of B-equivalences, e.g. the strong bisimulation, the branching bisimulation of [GW89], Milner's weak bisimulation and weak congruence, the dynamic bisimulation of [MS92]. Particular instances of L-equivalences are: the partial trace equivalence, the completed trace equivalence, the failure equivalence, etc.. We give final semantics descriptions for the B-equivalence and the L-equivalence, respectively, hence capturing uniformly, at the same time, many notions of branching-like equivalences and linear-like equivalences, respectively. In the case of branching-like equivalences, our approach generalizes [Acz93], where final semantics for weak equivalence and weak congruence on CCS were given, using the notion of *τ-prefix* labelled transition system, in order to provide a coinductive characterization of the weak congruence.

Chapter 6 is devoted to *imperative concurrent languages* in the style of de Bakker (see [BV96]). We try to capture the essence of an imperative concurrent language with higher order features, by introducing the general syntax, and the general format of the rules in the transition system. The configurations in the transition system are pairs of *global syntactical states* and statements, and the rules are specified in a sort of $GSOS$ format for imperative languages. As far as we know, these languages cannot be captured by any of the general formats introduced in the literature, in particular they are not instances of the $tyft/tyxt$ format with global state operator introduced in [GV92]. We extend the final semantics paradigm to imperative concurrent languages, providing two alternative final descriptions, to which correspond different, logically independent coinduction principles. We discuss two relevant examples of imperative concurrent languages with higher order features: the language $\mathcal{L}_{pas_2}$ with second order assignment, and the language $\mathcal{L}_{co_2}$ with second order communication. These languages are introduced and studied from the metric perspective in [BB93, BB9?]. $\mathcal{L}_{pas_2}$ is studied from the set-theoretic final perspective in [Len96]. We discuss the *a priori* approach for showing the compositionality of the final semantics for $\mathcal{L}_{pas_2}$ and $\mathcal{L}_{co_2}$, by defining semantical operators corresponding to the syntactical operators of the languages. Then we work out in detail the comparison between our final semantics and the metric denotational semantics of [BB93]. This section expands [Len96]. We prove that the equivalences induced by the two semantics coincide. This can be viewed as a sort of *full abstraction* of the metric semantics w.r.t. the syntactical final semantics. This result is very interesting, since not only we gain new proof principles for reasoning on the metric denotational semantics, but we succeed also in characterizing an equivalence induced by a *contravariant* functor, using solely *covariant* tools.

In Chapter 7, we investigate final descriptions of various λ-*theories* of the *untyped* λ-*calculus*, many of which are induced by reduction strategies. λ-calculus is the paradigm of *functional languages*, and its denotational semantics is usually expressed via *contravariant* functors in categories of complete partial orders (see [Wad76, CDZ87, HR92, EHR92, AO93, HL9?]). Following [HL95], in Chapter 7, we present two alternative (uniform) final descriptions of λ-theories, corresponding to the functors F and G, respectively. The functors F and G induce two conceptually independent coinduction principles. The first turns out to be the one corresponding to *applicative bisimulation* (see e.g. [AO93]), and the latter appears to be a new coinduction principle, which exploits the alternation of $\forall$ and $\exists$ quantifiers. A large part of Chapter 7 is devoted to the investigation of the general conditions under which final descriptions capture exactly the λ-theory we want to model. This is a rather difficult general problem. For the functor F, this amounts to showing that the λ-theory coincides with the corresponding *applicative equivalence*. We discuss in detail various techniques for showing this equality, expanding and generalizing [AO93, How96, Pit96, Len97, Len97a]. Our analysis is rather elaborate, and, as we will see, it provides further interesting proof principles (see e.g. Section 7.2). These discussions have also a significant bearing on *intersection type theory* of [BCD83, CDHL82], and on logical descriptions of domains in the style of Abramsky, [Abr91] (see Section 7.6.4).

Finally, in Chapter 8, we discuss final semantics for the π-calculus, a process algebra which models systems that can dynamically change the topology of the channels. We generalize the standard techniques of Chapter 5 for process algebras so as to accommodate the mechanism of *name creation* and the behaviour of the binding operators peculiar to the π-calculus. As a preliminary step, we give a *higher order* presentation of the π-calculus using as metalanguage LF, a *logical framework* based on typed λ-calculus. Such a presentation highlights the nature of the binding operators and elucidates the rôle of free and bound channels. We provide final descriptions for many bisimulation equivalences and congruences, arising both from *strong* and *weak* operational semantics, expanding the work in [HLMP98].

Chapter 5

Process Algebras

In this chapter we discuss final semantics for *process algebras* endowed with various bisimulation equivalences. That is we use the machinery introduced in Part I of the thesis, in order to characterize various equivalences on languages for processes. For the sake of completeness and as an introduction to the more sophisticated techniques of the following chapters, we will present the final semantics characterization both for *branching-like* and *linear-like bisimulation equivalences* for process algebra languages. These languages provide an important class of examples. One can even go as far as to say that the motivation for introducing final semantics was that of putting on general principled ground the operational equivalences on Milner's $SCCS$ (see Aczel's seminal work [Acz88]). The final semantics for these languages has been investigated by various authors. We recall a few fundamental pieces of work. Aczel studied strong equivalence for $SCCS$ in [Acz88], and weak equivalence and congruence for Milner's $SCCS$ and Hoare's CSP in [Acz93]. Rutten, in [Rut92], studied final semantics for languages whose operational semantics is in $tyxt/tyft$ format. Rutten and Turi, in [RT93, RT94], further investigate final semantics for CCS endowed with branching and trace equivalences. Turi, in [Tur96], studied final semantics for languages whose operational semantics is in $GSOS$ format.

In this chapter, we study both *branching-like* and *linear-like bisimulation equivalences* for process algebra languages, in a *uniform way*. In particular, in Section 5.1, we present the syntax of a generic process algebra language, and we recall the strong operational semantics of it, expressed in $GSOS$ format ([BIM88]). In Sections 5.2 and 5.3, we provide a general definition of branching-like bisimulations, which we call B*-bisimulations*, and a general definition of linear-like bisimulations, i.e. L*-bisimulations*. We call B-equivalence (L-equivalence) the greatest B-bisimulation (L-bisimulation). A B-bisimulation (L-bisimulation) is induced by a *transition relation*, which determines an operational semantics, and by a notion of *observation*. The notions of B-bisimulation and L-bisimulation generalize many important notions of branching and linear bisimulation (and equivalences). The following branching-like bisimulations and equivalences are particular instances of B-bisimulations:

- the *strong bisimulation*,
- the *branching bisimulation* of [GW89],
- the *weak bisimulation*,
- the *weak congruence*,
- the *dynamic bisimulation* of [MS92],
- ...

The following linear-like equivalences can be viewed as particular instances of greatest L-bisimulations:

- *partial trace equivalence*,
- *completed trace equivalence*,
- *failure equivalence*,
- ...

We give *final semantics* for a general process algebra language, inducing the B-equivalence and the L-equivalence, respectively. Thus we get a *uniform* definition of the final semantics for a large class of branching-like and linear-like equivalences. In our development, we shall try to use the simplest mathematical constructions. However, since in general B-bisimulations and L-bisimulations arise from *infinitely branching* transition relations, we are forced to model these notions of bisimulations via a functor which involves the $\mathcal{P}^0_{<|Proc_{PA}|^+}(\cdot)$ powerset constructor, where $|Proc^0_{PA}|$ denotes the cardinality of the syntactical set of closed process terms of the process algebra language.

Finally, in Section 5.4, we discuss the *compositionality* of the final semantics w.r.t. the μ-recursion operator.

5.1 Syntax and Operational Semantics

In this section we introduce the syntax and the operational semantics of our general paradigm process algebra language, $\mathcal{L}_{PA}$.

Definition 5.1.1 *Let A be a set of* atomic actions *such that the action $\tau \in A$. The set $Proc_{PA}$ of processes of the language $\mathcal{L}_{PA}$ is defined by*

$$(Proc_{PA} \ni)\ p \ ::=\ nil \ \mid x \ \mid\ a.p \ \mid\ op_n(p, \ldots, p) \ \mid\ \mu x.p\ ,$$

*where $x \in PVar$, $a \in A$, $a.(\cdot)$ is the a-*prefixing operator*, op_n is an n-ary syntactical operator, μ is the* recursion operator*, and all terms of the shape $\mu x.p$ are* guarded*, i.e. any occurrence of the variable x in p is in the scope of a prefixing operator.*
Let $Proc^0_{PA}$ denote the set of closed processes.

An operational semantics for $\mathcal{L}_{PA}$ is expressed in terms of a (generalized) *transition relation* $\rightarrow \subseteq \mathcal{A} \times Proc^0_{PA} \times (Proc^0_{PA})^k$, for $\mathcal{A}$ a suitable set of atomic actions, and $k \geq 1$. In the sequel we will consider various notions of transition relations on $\mathcal{L}_{PA}$, and hence various operational semantics, and we provide final descriptions for many equivalences, arising from these operational semantics and from various notions of observations.

Since all the transition relations which we will consider are derivable from the *strong transition relation*, we define now the *strong operational semantics* of $\mathcal{L}_{PA}$. This is presented in a *GSOS*-like format. First of all, we need to introduce the notion of *k-ary context.* k-ary contexts are contexts whose holes are labelled with indices from 1 to k, more precisely:

Definition 5.1.2 *Let $k \geq 1$. k-ary contexts are defined as follows:*

$$C_k[\cdot,\dots,\cdot] \ ::= \ nil \ | \ x \ | \ [\cdot]_i \ | \ a.C_k[\cdot,\dots,\cdot] \ | \ \mu x.C_k[\cdot,\dots,\cdot] \ | \ op_n(C_k[\cdot,\dots,\cdot],\dots,C_k[\cdot,\dots,\cdot]) \ ,$$

where $1 \leq i \leq n$.

For $k = 1$, we get the standard notion of unary contexts, which will be simply denoted by $C[\ ]$.

The strong operational semantics of $\mathcal{L}_{PA}$ is defined as follows:

Definition 5.1.3 (Strong Operational Semantics) *The* strong transition relation $\rightarrow_s \subseteq Proc^0_{PA} \times A \times Proc^0_{PA}$ *is defined by the rules (at), (μ)*

$$a.p \xrightarrow{a}_s p \quad (at)$$

$$\frac{p[\mu x.p/x] \xrightarrow{a}_s p_1}{\mu x.p \xrightarrow{a}_s p_1} \quad (\mu)$$

and by a finite number of rules of the following shape

$$\frac{\{\ p_i \xrightarrow{a_{ij}}_s q_{ij}\ \}^{1\leq i\leq n}_{1\leq j\leq l_i} \quad \{p_i \not\xrightarrow{b_{ij}}_s\}^{1\leq i\leq n}_{1\leq j\leq m_i}}{op_n(p_1,\dots,p_n) \xrightarrow{a}_s C_k[p_1,\dots,p_n,q_{11},\dots,q_{1l_1},\dots,q_{n1},\dots,q_{nl_n}]} \quad (op)$$

where $l_i, m_i \geq 0$, for all $i = 1,\dots,n$, $k \geq n + \Sigma^n_{i=1} l_i$, and $p \not\xrightarrow{a}_s$ is a given judgement satisfying the following property

$$p \not\xrightarrow{a}_s \ \Longrightarrow \ \not\exists q.\ p \xrightarrow{a}_s q \ .$$

The judgement $p \not\xrightarrow{a}_s$ in the definition above is a generalization of the judgement "p successfully terminates".

5.2 Final Semantics for $\mathcal{L}_{PA}$: Branching-like Equivalences

Branching-like equivalences can be characterized coinductively, in a uniform way, using the following general notion of B-bisimulation. A B-bisimulation is determined by a transition relation and by an *observation procedure*, i.e. an observation context:

Definition 5.2.1 (B**-bisimulation)** *Let* $\rightarrow \subseteq \mathcal{A} \times Proc^0_{PA} \times (Proc^0_{PA})^k$ *be a transition relation. A* B-bisimulation *is a symmetric relation* $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *such that*

$$p\mathcal{R}q \implies$$
$$\forall \alpha.\, \forall p_1, \ldots, p_k.\ p \xrightarrow{\alpha} (p_1, \ldots, p_k) \implies \exists (q_1, \ldots, q_k).\ q \xrightarrow{\alpha} (q_1, \ldots, q_k) \ \wedge$$
$$\forall i = 1, \ldots, k.\ C_i[p_i]\mathcal{R}C_i[q_i],$$

where $C_i[\cdot]$ *is the context in which we* observe *the behaviour of processes.*

Let us call B-equivalence *the greatest* B*-bisimulation.*

The notion of B-bisimulation is quite general, since, as we will see below, it captures many classical equivalences on $\mathcal{L}_{PA}$, like Milner's *strong bisimulation*, Milner's *weak bisimulation and congruence* ([Mil89]), van Glabbeek and Weijland branching bisimulation ([GW89]), and *progressing bisimulation* of [MS92]. B-bisimulations arise from transition relations which, in general, are *not finitely branching*, and *not even compactly branching*. In fact, as we will see below, weak equivalence and congruence, branching equivalence, and progressing equivalence are not finitely (compactly) branching already on a very simple fragment of CCS. Hence, categories based on c.p.o's or on c.m.s's do not seem flexible enough in order to give final semantics to the general notion of B-equivalence. However, we can use any of the *purely set-theoretic* categories introduced in Chapter 3, Section 3.2. E.g. we can work in $Class^*(U)$, where the class U of Urelementen is sufficiently large to contain the (generalized) atomic actions in $\mathcal{A}$.

The functor necessary to express the operational semantics determined by the transition $\rightarrow \subseteq \mathcal{A} \times Proc^0_{PA} \times (Proc^0_{PA})^k$ and by the observation procedure is defined by:

Definition 5.2.2 *Let* $F^B : Class^*(U) \rightarrow Class^*(U)$ *be the functor defined by:*

$$F^B(X) = \mathcal{P}_{<|Proc^0_{PA}|^+}(\mathcal{A} \times X^k)$$
$$F^B(f) = (id_A \times f^k)^+,$$

where $\mathcal{P}_{<|Proc^0_{PA}|^+}(\cdot)$ *denotes the set of all subsets whose cardinality is less than* $|Proc^0_{PA}|^+$*.*

The transition $\rightarrow \subseteq \mathcal{A} \times Proc^0_{PA} \times (Proc^0_{PA})^k$ and the observation procedure determine immediately a notion of F^B-coalgebra:

Definition 5.2.3 *Let $\alpha^B_{Proc^0_{PA}} : Proc^0_{PA} \to F^B(Proc^0_{PA})$ be the function defined by*

$$\alpha^B_{Proc^0_{PA}}(p) = \{(\alpha, (C_1[p_1], \ldots, C_k[p_k])) \mid p \xrightarrow{\alpha} (p_1, \ldots, p_k)\ \} .$$

Applying Theorem 3.5.5 of Chapter 3, Section 3.5 to the functor F^B, we immediately get

Lemma 5.2.4 *The functor F^B has final coalgebra, (U^B, α_{U^B}).*

The notion of B-equivalence is captured by the final semantics determined by the F^B-coalgebra $(Proc^0_{PA}, \alpha^B_{Proc^0_{PA}})$, in the sense given by the following

Proposition 5.2.5 *Let $\mathcal{M}^B_{PA} : (Proc^0_{PA}, \alpha^B_{Proc^0_{PA}}) \to (U^B, \alpha_{U^B})$ be the final semantics defined by*

$$\mathcal{M}^B_{PA}(p) = \{(\alpha, (\mathcal{M}^B_{PA}(C_1[p_1]), \ldots, \mathcal{M}^B_{PA}(C_k[p_k])) \mid p \xrightarrow{a} (p_1, \ldots, p_{j_i})\ \} .$$

The equivalence induced by $\mathcal{M}^B_{PA}$ on $\mathcal{L}_{PA}$ is the B-equivalence.

Here we recall a list of classical notions of equivalences on $\mathcal{L}_{PA}$, whose coinductive characterization is subsumed by the general definition of B-bisimulation.

Definition 5.2.6 (Strong Bisimulation, [Mil89]) *A symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ is a* strong bisimulation *if, for all p, q,*

$p\mathcal{R}p' \implies$
$\forall p_1,\ \forall a.\ (\ p \xrightarrow{a}_s p_1 \implies \exists q_1.\ q \xrightarrow{a}_s q_1 \ \wedge\ p_1\mathcal{R}q_1)\ .$
The greatest strong bisimulation, $\approx^s$, is called strong equivalence.

Definition 5.2.7 (Branching Bisimulation, [GW89]) *A symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ is a* branching bisimulation *if, for all p, q,*

$p\mathcal{R}q \implies$

- $\forall p_1.\ (\ p \xrightarrow{\tau}_s p_1 \implies \exists q_1.\ q \xrightarrow{\tau}_s q_1 \ \wedge\ p_1\mathcal{R}q_1)\ .$
- $\forall p_1,\ \forall a \neq \tau.\ (\ p \xrightarrow{a}_s p_1 \implies \exists q_1', q_2', q_1.\ q \Rightarrow q_1' \xrightarrow{a}_s q_2' \Rightarrow q_1 \ \wedge\ p\mathcal{R}q_1' \ \wedge\ p_1\mathcal{R}q_2' \ \wedge\ p_1\mathcal{R}q_1)\ ,$

where $\Rightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}_s$.
The greatest branching bisimulation, $\approx^b$, is called branching equivalence.

Definition 5.2.8 (Weak Bisimulation, [Mil89]) *A symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ is a* weak bisimulation *if, for all p, q,*

$p\mathcal{R}q \implies$
$\forall p_1,\ \forall a.\ (\ p \xrightarrow{a}_s p_1 \implies \exists q_1.\ q \stackrel{a}{\Rightarrow} q_1 \ \wedge\ p_1\mathcal{R}q_1)$,
where $\stackrel{a}{\Rightarrow}$ *denotes* $\Rightarrow$ *(i.e. the reflexive and transitive closure of* $\xrightarrow{\tau}_s$ *), if* $a = \tau$*, and* $\Rightarrow \circ \xrightarrow{a} \circ \Rightarrow$*, otherwise.*
The greatest branching bisimulation, $\approx^w$*, is called* weak equivalence.

Definition 5.2.9 (Weak Congruence, [Mil89]) *The relation* $\approx^{wc} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *is defined by*

$p \approx^{wc} q$ *if and only if*
$\forall p_1,\ a.\ p \xrightarrow{a}_s p_1 \implies \exists q_1.\ q \xrightarrow{a}_s q_1 \ \wedge\ p_1 \approx^w p_1'$.

Definition 5.2.10 (Progressing Bisimulation, [MS92]) *A symmetric relation* $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *is a* progressing bisimulation *if, for all* p, q,

$p\mathcal{R}q \implies$
$\forall p_1,\ \forall a.\ (\ p \xrightarrow{a}_s p_1 \implies \exists q_1, q_1', q_2'.\ q \Rightarrow q_1' \xrightarrow{a}_s q_2' \Rightarrow q_1 \ \wedge\ p_1\mathcal{R}q_1)$,
where $\Rightarrow$ *denotes the reflexive and transitive closure of* $\xrightarrow{\tau}_s$.
The greatest branching bisimulation, $\approx^p$*, is called* progressing equivalence.

Clearly the strong equivalence is a B-equivalence, just take the strong transition relation $\rightarrow_s$ as transition relation, and the empty context $[\cdot]$ as observation context.

Now we show how branching equivalence, weak equivalence, weak congruence, and progressing equivalence are B-equivalences. On this purpose we need to introduce appropriate transition relations. First of all we need the following instrumental definition:

Definition 5.2.11

- *Let* $\xrightarrow[\tau^*(n\tau)\tau^*]{a} \subseteq Proc^0_{PA} \times Proc^0_{PA}$*, for* $a \in A \setminus \{\tau\}$*, be the composition* $\Rightarrow \circ \xrightarrow{a}_s \circ \Rightarrow$.
- *Let* $\xrightarrow[\tau^*(n\tau)]{a} \subseteq Proc^0_{PA} \times (Proc^0_{PA})^2$*, for* $a \in A \setminus \{\tau\}$*, be the transition relation defined as follows:*
  $p \xrightarrow[\tau^*(n\tau)]{a} (p_1, p_2)$ *if and only if* $p \Rightarrow p_1 \xrightarrow{a}_s p_2$.
- *Let* $\xrightarrow[\tau^+]{\tau}$ *be the transitive closure of* $\xrightarrow{\tau}_s$.
- *Let* $\xrightarrow[\tau^+(n\tau)\tau^*]{a}$ *, for* $a \in A \setminus \{\tau\}$*, be defined by* $\xrightarrow[\tau^+]{\tau} \circ \xrightarrow{a}_s \circ \Rightarrow$.

Definition 5.2.12 (Branching Transition) *The* branching transition $\rightarrow_b \subseteq A \times Proc^0_{PA} \times (Proc^0_{PA})^2$ *is defined as follows:*

$p \xrightarrow[b]{a} (p_1, p_2)$ *if and only if*

- *either* $a = \tau$, $p \xrightarrow[\tau^*]{\tau} p_1$, *and* $p_2 = nil$,
- *or* $a \neq \tau$ *and* $p \xrightarrow[\tau^*(n\tau)]{a} (p_1, p_2)$.

Notice that, in the definition above, due to the nature of the branching bisimulation, we require $\rightarrow_b$ to relate a process p to a pair of processes (p_1, p_2). Hence, in case of τ actions, we are forced to introduce an "inactive" process p_2, e.g. we can always take it equal to nil. Another choice would be add to the syntax of $\mathcal{L}_{PA}$ a new neutral constant, and take p_2 equal to it, or else, we could modify the definition of B-bisimulation, by considering more than one transition relation inducing a B-bisimulation. But this would make the treatment unnecessary complex.

Definition 5.2.13 (Weak Transition) *Let* $\rightarrow_w \subseteq A \times Proc^0_{PA} \times Proc^0_{PA}$ *be the* weak transition *defined as follows:*

$p \xrightarrow[w]{a} p_1$ *if and only if*

- *either* $a = \tau$, *and* $p \Rightarrow p_1$,
- *or* $a \neq \tau$ *and* $p \xrightarrow[\tau^*(n\tau)\tau^*]{a} p_1$.

Definition 5.2.14 (Weak Congruence Transition) *Let* $\rightarrow_{wc} \subseteq A \times Proc^0_{PA} \times Proc^0_{PA}$ *be the* weak congruence transition *defined as follows:*

$p \xrightarrow[wc]{a} p_1$ *if and only if*

- *either* $a = \tau$, *and* $p \xrightarrow[\tau^+]{\tau} p_1$,
- *or* $a \neq \tau$ *and* $p \xrightarrow[\tau^+(n\tau)\tau^*]{a} p_1$.

Definition 5.2.15 (Progressing Transition) *Let* $\rightarrow_p \subseteq A \times Proc^0_{PA} \times Proc^0_{PA}$ *be the* progressing transition *defined by* $\Rightarrow \circ \rightarrow_s \circ \Rightarrow$.

The following easy lemmata give coinductive characterizations of the equivalences defined above. All these equivalences can be viewed as greatest fixed points of B-bisimulations.

Lemma 5.2.16 (Coinductive Char. of the Branching Eq.) *The branching equivalence is the greatest symmetric relation* $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *such that, for all* $p, q \in Proc^0_{PA}$,

$$p\mathcal{R}q \implies$$
$$\forall p_1,\ p_2,\ a.\ p \xrightarrow[b]{a} (p_1, p_2) \implies \exists q_1,\ q_2.\ q \xrightarrow[b]{a} (q_1, q_2) \ \wedge\ p_1\mathcal{R}q_1 \ \wedge\ p_2\mathcal{R}q_2 \ .$$

The following two lemmata are due to Aczel (see [Acz93]). In the case of the weak congruence the observation context is of the shape $\tau.[\cdot]$. This corresponds to the τ-prefix labelled transition systems of [Acz93].

Lemma 5.2.17 (Coinductive Char. of the Weak Eq.) *The weak equivalence is the greatest symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ such that, for all $p, q \in Proc^0_{PA}$,*

$$p\mathcal{R}q \implies$$
$$\forall p_1,\ a.\ \ p \xrightarrow[w]{a} p_1 \implies \exists q_1.\ \ q \xrightarrow[w]{a} q_1 \ \ \wedge\ \ p_1 \mathcal{R} q_1\ .$$

Lemma 5.2.18 (Coinductive Char. of the Weak Cong.) *The weak congruence is the greatest symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ such that, for all $p, q \in Proc^0_{PA}$,*

$$p\mathcal{R}q \implies$$
$$\forall p_1,\ a.\ \ p \xrightarrow[wc]{a} p_1 \implies \exists q_1.\ \ q \xrightarrow[wc]{a} q_1 \ \ \wedge\ \ \tau p_1 \mathcal{R} \tau q_1\ .$$

Lemma 5.2.19 (Coinductive Char. of the Progressing Eq.) *The progressing equivalence is the greatest symmetric relation $\mathcal{R} \subseteq Proc^0_{PA} \times Proc^0_{PA}$ such that, for all $p, q \in Proc^0_{PA}$,*

$$p\mathcal{R}q \implies$$
$$\forall p_1,\ a.\ \ p \xrightarrow[p]{a} p_1 \implies \exists q_1.\ \ q \xrightarrow[p]{a} q_1 \ \ \wedge\ \ p_1 \mathcal{R} q_1\ .$$

Finally, we point out that branching equivalence, weak equivalence, weak congruence, and progressing equivalence are not compactly branching already on a very simple process algebra language, which contains the non-deterministic choice operator and the parallel operator.

Lemma 5.2.20 *Weak equivalence, weak congruence, and branching equivalence are not compactly branching on the following fragment of CCS:*

$$(Proc_{CCS^-} \ni)\ p ::= \ nil \ \mid\ a.p \ \mid\ p+p \ \mid\ p||p \ \mid\ \mu x.p\ .$$

Proof The process $p = \mu x.((\tau x || a.nil) + b.nil)$ is such that

$$p \xrightarrow[\tau^+]{\tau} p || \underbrace{a.nil || \ldots || a.nil}_{k} \xrightarrow{b} \underbrace{a.nil || \ldots || a.nil}_{k}\ ,$$

for any $k \geq 1$. Hence $p \xrightarrow[\tau^+(n\tau)]{b} p_k$, for $k \geq 0$, where $p_k = \underbrace{a.nil || \ldots || a.nil}_{k}$, i.e., after a "$b$", the process p can do only a *finite* number of a's. Hence the sequence $\{p_k\}_{k \geq 0}$ has not limit. $\square$

Remark 5.2.21 (Simulation-like Equivalences) *In order to give final semantics to simulation-like equivalences (see e.g. van Glabbeek spectrum, [Gla90]), we need a refined notion of categorical F-bisimulation, i.e. that of* ordered categorical bisimulation *in a order enriched categorical setting (see [RT93, Fio96]). We shall not develop this.*

5.3 Final Semantics for $\mathcal{L}_{PA}$: Linear-like Equivalences

Linear equivalences do not have an immediate coinductive characterization as relations on processes. However, they can receive a coinductive chraracterization, when viewed as relations on $\mathcal{P}(Proc^0_{PA})$. The linear-like equivalences which we model via the final semantics can be characterized coinductively, in a uniform way, using the general notion of L-bisimulation on $\mathcal{P}(Proc^0_{PA})$ defined below. The structure of an L-bisimulation is essentially the same as that of a B-bisimulation. The only difference is that we need to extend transition relations to sets of processes. Since the examples of L-bisimulations which we consider are all determined by binary transition relations, and by the empty observation context, we take the general notion of L-bisimulation to be determined just by a binary transition relation and the empty observation context. Of course, this definition can be generalized by considering generalized transition relations, which relate a set to tuple of sets, and generic observation contexts.

Before giving the definition of L-bisimulation, we need to introduce some notation. Let $\mathcal{A}$ be a set of labels, and let $\rightarrow \subseteq \mathcal{A} \times Proc^0_{PA} \times Proc^0_{PA}$ be a transition relation. We extend $\rightarrow$ to $\mathcal{A} \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ as follows: let $P, P_1 \in \mathcal{P}(Proc^0_{PA})$, $P \xrightarrow{\alpha} P_1$ if and only if there exists $p \in P$ such that $p \xrightarrow{\alpha} p_1$, for some $p_1 \in Proc^0_{PA}$, and $P_1 = (P \setminus \{p\}) \cup \{p_1\}$. With $P \not\rightarrow$ we denote that no $p \in P$ makes any $\rightarrow$-transition.

Definition 5.3.1 (L-bisimulation) *Let $\rightarrow \subseteq \mathcal{A} \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ be a transition relation. An L-bisimulation is a symmetric relation on $\mathcal{R} \subseteq \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ such that*

$$P\mathcal{R}Q \implies$$
$$\forall P_1.\ P \xrightarrow{\alpha} P_1 \implies \exists Q_1.\ Q \xrightarrow{\alpha} Q_1 \ \wedge\ P_1\mathcal{R}Q_1.$$

The notion of L-bisimulation, as we will see below, captures many classical linear-like equivalences on $\mathcal{L}_{PA}$, such as *partial trace equivalence*, *completed trace equivalence*, *failure equivalence*, and other equivalences, see e.g. those of van Glabbeek's Spectrum (see [Gla90]).

The functor necessary to express the operational semantics determined by the transition $\rightarrow \subseteq \mathcal{A} \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ and the empty observation context is defined by:

Definition 5.3.2 *The functor $F^L : Class^*(U) \rightarrow Class^*(U)$ is defined by:*

$$F^L(X) = \mathcal{P}_{<|Proc^0_{PA}|^+}(\mathcal{A} \times X)$$
$$F^L(f) = (id_{\mathcal{A}} \times f)^+.$$

The transition $\rightarrow \subseteq \mathcal{A} \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ (with the empty observation context) determines immediately a notion of F^L-coalgebra:

Definition 5.3.3 *Let $\alpha^L_{\mathcal{P}(Proc^0_{PA})} : \mathcal{P}(Proc^0_{PA}) \rightarrow F^L(\mathcal{P}(Proc^0_{PA}))$ be the function defined by*

$$\alpha^L_{\mathcal{P}(Proc^0_{PA})}(P) = \{(\alpha, P_1) \mid \ P \xrightarrow{a} P_1 \ \} \ .$$

Applying Theorem 3.5.5 of Chapter 3, Section 3.5 to the functor F^L, we immediately get

Lemma 5.3.4 *The functor F^L has final coalgebra, (U^L, α_{U^L}).*

Proposition 5.3.5 *Let $\mathcal{M}^L_{PA} : (\mathcal{P}(Proc^0_{PA}), \alpha^L_{\mathcal{P}(Proc^0_{PA})}) \rightarrow (U^L, \alpha_{U^L})$ be the final semantics defined by*

$$\mathcal{M}^L_{PA}(P) = \{(\alpha, \mathcal{M}^L_{PA}(P_1)) \mid \ P \xrightarrow{\alpha} P_1 \ \} \ .$$

The equivalence induced by $\mathcal{M}^L_{PA}$ on $\mathcal{L}_{PA}$ is the L-equivalence.

Remark 5.3.6 *In order to give final semantics to $\mathcal{L}_{PA}$ with linear-like equivalences, we could use functors different from F^L. By way of example, we mention a functor, inspired by [Bre97]. Define $F : Class^*(U) \rightarrow Class^*(U)$ by:*

$$F(X) = \Sigma_{J \in \mathcal{P}_{ne}(\mathcal{A})}(J \rightarrow X) \ ,$$

where $\mathcal{P}_{ne}(\cdot)$ denotes the powerset of nonempty sets.
A notion of linear-like bisimulation induces an F-coalgebra on $\mathcal{P}(Proc^0_{PA})$ which is defined exactly as the F^L-coalgebra of Definition 5.3.3. In fact, it is immediate to see that, the F^L-coalgebra of Definition 5.3.3 associates to a set of processes P the empty function, if $P \not\rightarrow$, otherwise, if $P \rightarrow$, the F^L-coalgebra returns a function from the set J of initial actions to the continuation sets of processes.

Similar functors were first introduced and analyzed in [Bre97] for the metric setting. The interest of these functors lies in the fact that the powerset of the space of the streams on actions is a fixed point and final coalgebra. This is shown in [Bre97] in the metric setting, for finitely branching and image finite processes.
The functors F^L and F are related in a precise categorical sense to each other, and to the functor on semilattices used in [Tur96] for modeling trace equivalence. We do not elaborate on this.

Now we list some definitions of linear-like equivalences, which receive a coinductive characterization by suitably instantiating the general notion of L-bisimulation.

Let A^* be the set of streams on A, and let $\Longrightarrow$ denote the reflexive and transitive closure of $\xrightarrow{\ \ }_s$; $p \overset{\sigma}{\Longrightarrow} p_k$, for $\sigma \in A^*$, is an abbreviation for $p \xrightarrow{a_1}_s p_1 \xrightarrow{a_2}_s \ldots \xrightarrow{a_k}_s p_k$, where $\sigma = a_1 \ldots a_k$.

Definition 5.3.7 (Partial Trace Equivalence) *The* partial trace equivalence $\sim^p \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *is defined by:*

$$p \sim^p q \textit{ if and only if}$$

$$\forall \sigma \in A^*,\ \sigma \textit{ finite},\ \exists p_1.\ p \overset{\sigma}{\Longrightarrow} p_1 \iff \exists q_1.\ q \overset{\sigma}{\Longrightarrow} q_1\ .$$

Definition 5.3.8 (Completed Trace Equivalence) *The* completed trace equivalence $\sim^c \subseteq Proc^0_{PA} \times Proc^0_{PA}$ *is defined by:*

$$p \sim^c q \textit{ if and only if}$$

$$\forall \sigma \in A^*,\ p \overset{\sigma}{\Longrightarrow} nil \iff q \overset{\sigma}{\Longrightarrow} nil\ .$$

Definition 5.3.9 (Failure Equivalence, [Gla90]) *The function*
$\iota : Proc^0_{PA} \to \mathcal{P}_{<|A|^+}(A)$ *that,*
for each process p, gives the set of initial actions of p, is defined by
$\iota(p) = \{a \mid \exists p_1.\ p \xrightarrow{a} p_1\ \}$.
A pair $(\sigma, X) \in A^* \times \mathcal{P}_{<|A|^+}(A)$ is a failure pair *of $p \in Proc^0_{PA}$, if there exists $p' \in Proc^0_{PA}$ such that $p \overset{\sigma}{\Longrightarrow} p'$, and $\iota(p') \cap X = \emptyset$.*
The failure equivalence, *$\sim^f$, is defined by:*

$$p \sim^f q \textit{ if and only if } p \textit{ and } q \textit{ have the same set of failure pairs.}$$

All the equivalences above can be immediately extended on $\mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ as follows: for $P, P' \in \mathcal{P}(Proc^0_{PA})$, and for $\sim \in \{\sim^p, \sim^c, \sim^f\}$, we define

$$P \sim Q \text{ if and only if}$$

$$\forall p \in P\ \exists q \in Q.\ p \sim q\ \wedge\ \forall q \in Q\ \exists p \in P.\ p \sim q.$$

We can naturally give a coinductive characterization of the linear-like equivalences above, when these are viewed as relations on $\mathcal{P}(Proc^0_{PA})$. First of all we need to introduce appropriate notions of transition relations. The following definition is instrumental:

Definition 5.3.10 *Let $P, P' \in \mathcal{P}(Proc^0_{PA})$ and let $a \in A$:*

- *let $P \xrightarrow{a\downarrow} P'$ denote the fact that there exists $p \in P$ such that $p \xrightarrow{a}_s nil$ and $P' = (P \setminus \{p\}) \cup \{nil\}$;*

- *let* $P \xrightarrow{a \not\downarrow} P'$ *denote the fact that there exists* $p \in P$ *and* $p' \neq nil$ *such that* $p \xrightarrow{a}_s p'$ *and* $P' = (P \setminus \{p\}) \cup \{p'\}$;
- *let* $\iota(P) = \bigcup_{p \in P} \iota(p)$.

Definition 5.3.11 (Partial Trace Transition) *The* partial trace transition $\rightarrow_p \subseteq A \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ *is defined as follows:*

$$P \xrightarrow{a}_p P_1 \text{ if and only if } P \xrightarrow{a}_s P_1 \ .$$

Definition 5.3.12 (Completed Trace Transition) *Let* $\mathcal{A}_c = \{a \downarrow \mid a \in A\} \cup \{a \not\downarrow \mid a \in A\}$. *The* completed trace transition $\rightarrow_c \subseteq \mathcal{A}_c \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ *is defined as follows:*

$P \xrightarrow{\alpha}_c P_1$ *if and only if*

- *either* $P \xrightarrow{a\downarrow} P_1$ *and* $\alpha = a \downarrow$,
- *or* $P \xrightarrow{a \not\downarrow} P_1$ *and* $\alpha = a \not\downarrow$.

Definition 5.3.13 (Failure Transition) *Let* $\mathcal{A}_f = \{(a, P) \mid a \in A \ \wedge \ P \in \mathcal{P}(Proc^0_{PA})\}$. *The* failure transition $\rightarrow_f \subseteq \mathcal{A}_f \times \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ *is defined as follows:*

$$P \xrightarrow{\alpha}_f P_1 \text{ if and only if } P \xrightarrow{a}_s P_1 \ \wedge \ \alpha = (a, \iota(P)) \ .$$

Lemma 5.3.14 (Coinductive Char. of the Partial Trace Eq.) *The partial trace equivalence is the greatest symmetric relation* $\mathcal{R} \subseteq \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ *such that:*

$P \mathcal{R} Q \implies$

- $P \not\rightarrow_p \implies Q \not\rightarrow_p$;
- $\forall P_1 \in \mathcal{P}(Proc^0_{PA}) \ \forall a \in A. \ P \xrightarrow{a}_p P_1 \implies \exists Q_1. \ Q \xrightarrow{a}_p Q_1 \ \wedge \ P_1 \mathcal{R} Q_1$.

Lemma 5.3.15 (Coinductive Char. of the Completed Trace Eq.) *The completed trace equivalence is the greatest symmetric relation* $\mathcal{R} \subseteq \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ *such that:*

$P \mathcal{R} Q \implies$

- $P \not\rightarrow \implies Q \not\rightarrow$;

- $\forall P_1 \in \mathcal{P}(Proc^0_{PA})\ \forall \alpha \in \mathcal{A}_c.\ P \xrightarrow{\alpha}_c P_1 \implies \exists Q_1.\ Q \xrightarrow{\alpha}_c Q_1 \wedge P_1 \mathcal{R} Q_1$.

Lemma 5.3.16 (Coinductive Char. of the Failure Eq.) *The failure equivalence is the greatest symmetric relation $\mathcal{R} \subseteq \mathcal{P}(Proc^0_{PA}) \times \mathcal{P}(Proc^0_{PA})$ such that:*

$P\mathcal{R}Q \implies$

- $P \not\rightarrow \implies Q \not\rightarrow$;
- $\forall P_1 \in \mathcal{P}(Proc^0_{PA})\ \forall \alpha \in \mathcal{A}_f.\ P \xrightarrow{\alpha}_f P_1 \implies \exists Q_1.\ Q \xrightarrow{\alpha}_f Q_1 \wedge P_1 \mathcal{R} Q_1$.

As mentioned earlier, similar coinductive characterizations can be given for many other congruences, such as those of van Glabbeek's spectrum ([Gla90]).

5.4 Compositionality of Final Semantics

So far we have defined final semantics only on closed processes. What it means for final semantics to be compositional is therefore straightforward, if we consider only contexts made out of syntactical operators op_n's and prefixing operator. Moreover, in the case in which the final semantics induces a congruence w.r.t. the syntactical operators op_n's and the prefixing operator, we can even define "a posteriori" semantical operators on the codomain of the final semantics $\mathcal{M}_{PA}$ as follows. Let op_n be a n-ary syntactical operator, for $p_1, \ldots, p_n \in Proc^0_{PA}$, we define

$$\widetilde{op}_n(\mathcal{M}_{PA}(p_1), \ldots, \mathcal{M}_{PA}(p_n)) = \mathcal{M}_{PA}(op_n(p_1, \ldots, p_n)) .$$

A deeper analysis is required for expressing compositionality of the final semantics w.r.t. the recursion operator. We need first to extend the definition of the final semantics also to *open* processes. There are at least two possible choices. For simplicity, we discuss only the case of the strong equivalence, the other cases can be dealt with similarly.

1. Let $Proc^g_{PA}$ denote the set of *guarded* processes. We introduce *syntactical* environments $\rho : PVar \to Proc^g_{PA}$, and, for all $x \in PVar$, we add rules of the following shape to the transition system of Definition 5.1.3:

$$\frac{\rho(x) \xrightarrow{a}_s p_1}{x \xrightarrow{a}_s p_1} \ .$$

   Given an environment ρ, it is immediate to extend the strong equivalence $\approx^s$ to open terms.

Then, for each environment ρ, we naturally endow $Proc_{PA}$ with an F^s-coalgebra structure, for $F^s : Class^*(U) \rightarrow Class^*(U)$ the functor defined by $F^s(X) = \mathcal{P}_{<|Proc_{PA}|^+}(A \times X)$. Namely, let $(Proc_{PA}, \alpha^s_{Proc_{PA},\rho})$ be the F^s-coalgebra defined by:

$$\alpha^s_{Proc_{PA},\rho}(p) = \{(a, p_1) \mid p \xrightarrow[s]{a} p_1 \} .$$

If we consider the final semantics $\mathcal{M}^s_\rho : (Proc_{PA}, \alpha^s_{Proc_{PA},\rho}) \rightarrow (U^s, \alpha_{U^s})$, then one can easily check that, for all ρ, $\mathcal{M}^s_\rho$ coincides on closed processes with $\mathcal{M}^s$. This follows from the fact that, if $j : (Proc^0_{PA}, \alpha^s_{Proc^0_{PA}}) \rightarrow (Proc_{PA}, \alpha^s_{Proc_{PA},\rho})$ is the inclusion F^s-coalgebra map, then $\mathcal{M}^s_\rho \circ j$ is a coalgebra map from $(Proc^0_{PA}, \alpha^s_{Proc^0_{PA}})$ to the final coalgebra (U^s, α_{U^s}). Hence, by finality, $\mathcal{M}^s_\rho \circ j = \mathcal{M}^s$.
Now, the compositionality of the final semantics w.r.t. the recursion operator can be expressed as follows: let $p, q \in Proc_{PA}$,

$$\forall \rho, \rho'. \ (\rho \approx^s \rho' \ \Rightarrow \ \mathcal{M}^s_\rho(p) = \mathcal{M}^s_{\rho'}(q)) \implies \mathcal{M}^s_\rho(\mu x.p) = \mathcal{M}^s_{\rho'}(\mu x.q) \ ,$$

where $\rho \approx^s \rho'$ denotes the fact that $\forall x.\ \rho(x) \approx^s \rho'(x)$. Notice that we allow ρ and ρ' to be *strongly equivalent* and not exactly the same environment, since the environments are syntactical. One can easily check that a sufficient condition which guarantees the compositionality of $\mathcal{M}^s$ w.r.t. the recursion operator amounts to the fact that $\approx^s$ is a *congruence* w.r.t. the recursion operator, in the following sense:

$$\forall r, r' \in Proc^0_{PA}. \ (r \approx^s r' \ \Rightarrow \ p[r/x] \approx^s q[r/x]) \implies \mu x.p \approx^s \mu x.q \ .$$

2. We introduce a new constant in the process algebra syntax, $*$, and new action symbols, corresponding to the variables in $PVar$, i.e. $\mathcal{A} = A \cup \{a_x \mid x \in PVar\}$. We add to the transition system $\mathcal{T}_{PA}$ the following rules for the variables: $x \xrightarrow{a_x} *$. The notion of strong bisimulation can be naturally extended on all the set of processes so defined, $Proc^*_{PA}$, by simply requiring that the new constant $*$ only match with itself. Call $\approx^{s^*}$ the notion of equivalence so obtained. Now we can define a final semantics on all $Proc^*_{PA}$, without introducing any notion of environment. Let $F^{s^*} : Class^*(U) \rightarrow Class^*(U)$ be the functor defined by: $F^{s^*}(X) = \mathcal{P}_{<|Proc^*_{PA}|^+}(\mathcal{A} \times (X + \{*\}))$. Let $(Proc^*_{PA}, \alpha^{s^*}_{Proc^*_{PA}})$ be the F^{s^*}-coalgebra defined as follows

$$\alpha^{s^*}_{Proc^*_{PA}}(p) = \{(a, p_1) \mid p \xrightarrow{a} p_1 \} .$$

One can easily check that the equivalence induced by the final semantics $\mathcal{M}^{s^*}_{PA^*} : (Proc^*_{PA}, \alpha^{s^*}_{Proc^*_{PA}}) \rightarrow (U^{s^*}, \alpha_{U^{s^*}})$ is exactly $\approx^{s^*}$, and that $\approx^{s^*}$ is conservative w.r.t. $\approx^s$ on closed processes, in the sense that: if $p, q \in (Proc_{PA})^0$,

$$\mathcal{M}^{s^*}_{PA^*}(p) = \mathcal{M}^{s^*}_{PA^*}(q) \implies \mathcal{M}^s_{PA}(p) = \mathcal{M}^s_{PA}(q) \ .$$

Now the compositionality of $\mathcal{M}^{s^*}_{PA^*}$ w.r.t. the recursion operator can be expressed simply as follows:

$$\mathcal{M}^{s^*}_{PA^*}(p) = \mathcal{M}^{s^*}_{PA^*}(q) \implies \mathcal{M}^{s^*}_{PA^*}(\mu x.p) = \mathcal{M}^{s^*}_{PA^*}(\mu x.q) \ .$$

The implication above follows immediately if $\approx^{s^*}$ is a congruence w.r.t. the recursion operator, i.e.:

$$p \approx^{s^*} q \implies \mu x.p \approx^{s^*} \mu x.q \ .$$

Finally, we point out that the compositionality of process algebra languages in $GSOS$ format with strong bisimulation is discussed in [Tur96], where a categorical method for proving compositionality of final semantics is presented. In particular, a general condition on the operational semantics is provided, which guarantees that the final semantics coincides with the initial semantics. This categorical approach applies in particular to operational semantics in $GSOS$ format, but it is not clear whether it can be extended to more complex cases, like the case of imperative concurrent languages such as those discussed in Chapter 6, or the case of functional languages, in particular the λ-calculus discussed in Chapter 7, or the case of π-calculus discussed in Chapter 8.

Chapter 6

Higher Order Imperative Concurrent Languages

In this chapter we discuss final semantics for imperative concurrent languages with particular attention to the higher order features that such languages often exhibit, e.g. *second order assignment* or *second order communication*. We shall present them in the style of de Bakker (see [BV96]). Usually, they are modeled by means of *recursive* domains of metric processes (see e.g. [BB93, BV96, BB9?]). We will show that a *syntactical* final approach to semantics making use solely of sets can be applied successfully also to this class of concurrent languages featuring higher order constructs.

First of all, in Section 6.1, we present the syntax and the operational semantics of a generic higher order concurrent language, which we call $\mathcal{L}_{?_2}$. The operational semantics is given in terms of transition systems whose configurations are pairs of *global syntactical states* and statements. The equivalences we focus on, and we try to describe via the final semantics, do not come from a *SOS* system in tyft/tyxt-like format (see [GV92] for a description of this format). And in particular, it cannot be captured using the *global state operator* of [GV92]. If this were the case, techniques similar to those of the preceding chapter could be used.

In Section 6.2, we will proceed then to give the final semantics to $\mathcal{L}_{?_2}$, and to discuss the *compositionality* of the final semantics. We present two alternative final descriptions of $\mathcal{L}_{?_2}$, and we discuss the corresponding coinduction principles which arise. In Section 6.3, the language $\mathcal{L}_{?_2}$ will be specialized to two examples of concurrent languages with higher order features:

1. The language $\mathcal{L}_{pas_2}$, with *second order assignment*. This language is obtained by adding a parallel operator to the language with second order assignment $\mathcal{L}_{as_2}$, introduced and studied by de Bakker and van Breugel in [BB93] from the metric perspective. $\mathcal{L}_{pas_2}$ is studied from the final semantics perspective in [Len96].

2. The language $\mathcal{L}_{co_2}$ with *second order communication.* This language is introduced and studied in [BB93], and further investigated in [BB9?] from the metric perspective.

In Section 6.3.3, we illustrate the *a priori* approach for showing the compositionality of the final semantics for these two languages. In Section 6.3.4, we discuss alternative coinductive characterizations of the equivalence induced by the final semantics. Finally, in Section 6.3.5, we prove that, for $\mathcal{L}_{pas_2}$, the equivalence induced by the final semantics coincides with the equivalence induced by the metric semantics. The proof of this fact is technically rather involved, and it requires a fine analysis of derivations in the transition system. This proof is an expansion of the proof which appears in [Len96]. This result can be viewed as a *full abstraction* of the metric initial semantics w.r.t. the syntax-based purely set-theoretic final semantics. The interest of it lies in the fact that we can characterize an equivalence defined using a *contravariant* functor (in the metric spaces) using solely "covariant" tools. We conjecture that the same result holds also for the language $\mathcal{L}_{co_2}$.

6.1 Syntax and Operational Semantics

Let $\mathcal{L}_{?_2}$ be a generic imperative concurrent language with *individual* (first order) *variables* in $(v \in)IVar$, *procedure* (higher order) *variables* in $(x \in)PVar$.

Definition 6.1.1 *The set of statements of $\mathcal{L}_{?_2}$ is defined as follows:*

$$(Stat_{?_2} \ni) \; s ::= \; E \;\mid\; s_{at} \;\mid\; x \;\mid\; op_n(s, \ldots, s) \;,$$

where

- *E denotes the* empty statement*;*
- *s_{at} denotes an* atomic statement*; atomic statements include first order assignment, i.e. statements of the shape $v := e$, where $v \in IVar$, $IVar$ is a set of individual variables, and e is an expression;*
- *$x \in PVar$;*
- *op_n denotes a syntactical operator of arity n.*

The operational semantics of $\mathcal{L}_{?_2}$ is given by a transition system whose configurations are pairs of *syntactical states* and statements. Syntactical states are *global* and keep track of the value of both individual variables and procedure variables. We assume a set of basic values Val, which are assignable to variables in $IVar$. *Extended syntactical states* account also for higher order communication. In particular, we assume there exists a set $Chan$ of channel's names for the higher order communication.

Definition 6.1.2 *The set of* syntactical states $SynState_{?_2}$ *is defined by*

$$(\sigma \in)\ SynState_{?_2} = (IVar \to Val) \times (PVar \to Stat_{?_2})\ .$$

The set of extended syntactical states $SynState^{ext}_{?_2}$ *is defined by*

$$(\eta \in)\ SynState^{ext}_{?_2} = SynState_{?_2} \cup (Chan \times Stat_{?_2}) \cup (\overline{Chan} \times PVar)\ ,$$

where $(c \in) Chan$ *is a set of channel's names, and* $\overline{Chan} = \{\overline{c} \mid c \in Chan\}$.

In order to discuss the transition system $\mathcal{T}_{?_2}$ for $\mathcal{L}_{?_2}$, we need to introduce the notion of k*-ary context* for the language $\mathcal{L}_{?_2}$.

Definition 6.1.3 *Let* $k \geq 1$. k*-ary contexts are defined as follows:*

$$C_k[\ \cdot, \ldots, \cdot]\ ::=\ [\ \cdot]_i\ \mid\ E \mid\ x\ \mid\ s_{at}\ \mid\ op_n(C_k[\ \cdot, \ldots, \cdot], \ldots, C_k[\ \cdot, \ldots, \cdot])\ .$$

where $1 \leq i \leq n$.

For $k = 1$, we get the standard notion of unary contexts, which will be simply denoted by $C[\ ]$.

In the following definition we present the general format of the rules in the transition system $\mathcal{T}_{?_2}$. This is a sort of $GSOS$ format for concurrent languages with global states. We remark that these languages cannot be cast into the $tyxt/tyft$ format with the global operator defined in [GV92]. The equivalence induced by this is not the one normally associated to them, following de Bakker's ([BV96]).

Definition 6.1.4 *The rules in the transition system* $\mathcal{T}_{?_2}$ *are of the shape*

$$(\sigma, s_{at}) \to (f(\sigma, s_{at}), E)\quad (at)\ ,$$

$$(\sigma, x) \to (\sigma, \sigma(x))\quad (var)\ ,$$

where f *is a function from* $SynState_{?_2} \times Stat_{?_2}$ *to* $SynState^{ext}_{?_2}$,

and of the shape (*op*)

$$\frac{\{(\sigma, s_j) \to (\eta_j, s'_j)\}_{j \in J} \qquad \{s_i \not\to\ \}_{i \in I}}{(\sigma, op_n(s_1, \ldots, s_n)) \to (g(\sigma, \eta_{j_1}, \ldots, \eta_{j_l}, s_1, \ldots, s_n), C_n[s_{i_1}, \ldots, s_{i_k}, s'_{j_1}, \ldots, s'_{j_l}])}$$

where $J = \{j_1, \ldots, j_l\}$, $I \subseteq \{i_1, \ldots, i_k\}$, $I, J \subseteq \{1, \ldots, n\}$, $n \geq 1$, $1 \leq l \leq n$, $0 \leq k \leq n$, g *is a function from* $SynState_{?_2} \times (SynState^{ext}_{?_2})^l \times (Stat_{?_2})^n$ *to* $SynState^{ext}_{?_2}$, *and the judgement* $s \not\to$ *is defined by* $\not\exists \sigma, \eta, s'.\ (\sigma, s) \to (\eta, s')$.

For each operator op_n *there can be zero, one or more rules of the form* (*op*).

The presence of negative premisses in the rules (op) above is not problematic. In fact, one can define a function $f : Stat_{?_2} \to \{0,1\}$ such that $\exists \sigma, \eta, s'.\ (\sigma, s) \to (\eta, s') \iff f(s) = 0$, by induction on statements as follows
$f(s_{at}) = 0$
$f(x) = 0$
$f(E) = 1$
$f(op(s_1, \ldots, s_n)) = 0$, if there exists a rule (op) such that $\forall j \in J.\ f(s_j) = 0$ and $\forall i \in I.\ f(s_i) = 1$
$f(op(s_1, \ldots, s_n)) = 1$, otherwise.

As we will see below, interesting transition systems are those in which the functions g's, appearing in the rule (op), preserve the contextual closure of the equivalence on statements modeled by the final semantics. In fact, if this is the case, one can easily prove that the final semantics is compositional.

6.2 Final Semantics

Final semantics for $\mathcal{L}_{?_2}$ can be given using the following functor:

Definition 6.2.1 *Let $F_{?_2} : Class^*(U) \to Class^*(U)$ be the endofunctor defined by*

$$\begin{aligned} F_{?_2}(X) = [SynState_{?_2} \to \mathcal{P}_{ne,<2^{\aleph_0}}((((IVar \to Val) \times (PVar \to X)) \cup \\ (Chan \times X) \ \cup\ (\overline{Chan} \times PVar)) \times X)] + \{E\} \\ F_{?_2}(f) = [id_{SynState_{?_2}} \to \mathcal{P}_{ne,<2^{\aleph_0}}((id_{IVar \to Val} \times (id_{PVar} \to f)) \cup \\ (id_{Chan} \times f) \ \cup\ id_{\overline{Chan} \times PVar}) \times f)] + id_{\{E\}}\ , \end{aligned}$$

where $\mathcal{P}_{ne,<2^{\aleph_0}}$ denotes the set of all non-empty subsets with cardinality less than $2^{\aleph_0}$, and $+$ denotes the disjoint sum.

In the definition of the functor $F_{?_2}$, we consider the powerset $\mathcal{P}_{ne,<2^{\aleph_0}}$, in order to allow possibly infinitely branching transition systems. But, for the two example languages which we will consider, the powerset of finite sets will be sufficient, since both transition systems are finitely branching.

The transition system $\mathcal{T}_{?_2}$ induces the following $F_{?_2}$-coalgebra:

Definition 6.2.2 *Let $(Stat_{?_2},\ \alpha_{Stat_{?_2}})$ be the $F_{?_2}$-coalgebra defined by*

$$\alpha_{Stat_{?_2}}(s) = \begin{cases} \{(\sigma, \{(\eta_1, s_1) \mid (\sigma, s) \to (\eta_1, s_1)\}) \mid \sigma \in SynState_{?_2}\} & \text{if } s \to \\ E & \text{if } s \not\to \end{cases}$$

where $s \to$ means that, for every σ, there is a transition starting from (σ, s).

Notice that, in the definition above, for all s, $\alpha_{Stat_{?_2}}(s)$ defines a function.

Proposition 6.2.3 *The functor $F_{?_2}$ has final coalgebra $(U_{?_2}, \alpha_{U_{?_2}})$. The final semantics $\mathcal{M}_{?_2} : (Stat_{?_2},\ \alpha_{Stat_{?_2}}) \to (U_{?_2}, \alpha_{U_{?_2}})$ is defined by:*

$$\mathcal{M}_{?_2}(s) = \begin{cases} \{(\sigma,\ \{((\pi_1(\sigma_1),\ \{(x,\ \mathcal{M}_{?_2}(\pi_2(\sigma_1)(x))) \mid x \in PVar\}), \\ \qquad \mathcal{M}_{?_2}(s_1)) \mid (\sigma,\ s) \to (\sigma_1,\ s_1)\} \cup \\ \quad \{((c, \mathcal{M}_{?_2}(s_2)), \mathcal{M}_{?_2}(s_1)) \mid (\sigma, s) \to ((c, s_2), s_1)\} \cup \\ \qquad \{((\overline{c}, x), \mathcal{M}_{?_2}(s_1)) \mid (\sigma, s) \to ((\overline{c}, x), s_1)\}) \mid \\ \qquad \qquad \sigma \in SynState_{?_2}\} & \text{if } s \to \\ E & \text{if } s \not\to . \end{cases}$$

Moreover,

$$\mathcal{M}_{?_2}(s) = \mathcal{M}_{?_2}(s') \iff s \approx_{?_2} s' ,$$

where $\approx_{?_2}$ is the union of all $F_{?_2}$-bisimulations on the coalgebra $(Stat_{?_2},\ \alpha_{Stat_{?_2}})$.

It is easy to prove the following set-theoretic coinductive characterization of the equivalence $\approx_{?_2}$.

Lemma 6.2.4 *$\approx_{?_2}$ is the greatest fixed point of the operator $\Phi_{?_2} : \mathcal{P}(Stat_{?_2} \times Stat_{?_2}) \to \mathcal{P}(Stat_{?_2} \times Stat_{?_2})$ defined as follows:*

$\Phi_{?_2}(\mathcal{R}) = \{(s, s') \mid \forall\sigma.(((\sigma, s) \to (\eta_1, s_1) \Rightarrow \exists\eta_1', s_1'.((\sigma, s') \to (\eta_1', s_1') \wedge \eta_1\mathcal{R}\eta_1' \wedge s_1\mathcal{R}s_1')) \wedge ((\sigma, s') \to (\eta_1', s_1') \Rightarrow \exists\eta_1, s_1.((\sigma, s) \to (\eta_1, s_1) \wedge \eta_1\mathcal{R}\eta_1' \wedge s_1\mathcal{R}s_1')))\}$,

where $\eta_1\mathcal{R}\eta_1'$ is a shorthand for:
$(\eta_1, \eta_1' \in SynState_{?_2} \wedge \pi_1(\eta_1) = \pi_1(\eta_1') \wedge \forall x \in PVar.\ (\pi_2(\eta_1)(x)\mathcal{R}\pi_2(\eta_1')(x))) \vee$
$(\eta_1, \eta_1' \in Chan \times Stat_{?_2} \wedge \pi_1(\eta_1) = \pi_1(\eta_1') \wedge \pi_2(\eta_1)\mathcal{R}\pi_2(\eta_1')) \vee$
$(\eta_1, \eta_1' \in \overline{Chan} \times PVar \wedge \eta_1 = \eta_1')$.

6.2.1 Compositionality of the Final Semantics

We discuss now compositionality of the final semantics. First of all we need to consider the *contextual closure* of the equivalence $\approx_{?_2}$.

Definition 6.2.5 *Let $\widehat{\approx}_{?_2} \subseteq Stat_{?_2} \times Stat_{?_2}$ be the* contextual closure *of $\approx_{?_2}$, i.e.:*

$$\widehat{\approx}_{?_2} = \{(C_k[s_1, \ldots, s_k], C_k[s_1', \ldots, s_k'],) \mid C_k[\cdot, \ldots, \cdot] \text{ context} \wedge \forall i = 1, \ldots, k.\ s_i \approx_{?_2} s_i'\} .$$

Definition 6.2.6 *Let $i, j, h \geq 0$, $i + j + h \geq 1$. A function $f : (SynState_{?_2})^i \times (SynState_{?_2}^{ext})^j \times (Stat_{?_2})^h \to SynState_{?_2}$ is $\widehat{\approx}_{?_2}$-preserving if, for all $\sigma_p, \sigma_p' \in SynState_{?_2}$, $1 \leq p \leq i$, for all $\eta_q, \eta_q' \in SynState_{?_2}^{ext}$, $1 \leq q \leq j$, for all $s_r, s_r' \in Stat_{?_2}$, $1 \leq r \leq h$,*

$$(\sigma_p \widehat{\approx}_{?_2} \sigma_p' \wedge \eta_q \widehat{\approx}_{?_2} \eta_q' \wedge s_r \widehat{\approx}_{?_2} s_r' \implies f(\sigma_1, \ldots, \sigma_i, \eta_1, \ldots, \eta_j, s_1, \ldots, s_h) \widehat{\approx}_{?_2} f(\sigma_1', \ldots, \sigma_i', \eta_1', \ldots, \eta_j', s_1', \ldots, s_h')).$$

Compositionality of the final semantics for suitable transition systems can be proved now using the *a posteriori* approach:

Theorem 6.2.7 *If $\mathcal{T}_{?_2}$ is a transition system such that the functions g's appearing in the rules (op) are $\widehat{\approx}_{?_2}$-preserving, then $\widehat{\approx}_{?_2} = \approx_{?_2}$.*

Proof We show that $\widehat{\approx}_{?_2}$ is a $\Phi_{?_2}$-bisimulation by induction on contexts, i.e. we show that

$$(C_k[s_1,\ldots,s_k], C_k[s'_1,\ldots,s'_k]) \in \widehat{\approx}_{?_2} \implies$$

$$\forall\sigma.\ ((\sigma, C_k[s_1,\ldots,s_k]) \to (\eta_1, s_1) \implies$$
$$((\sigma, C_k[s'_1,\ldots,s'_k]) \to (\eta'_1, s'_1) \ \wedge\ \eta_1 \widehat{\approx}_{?_2} \eta'_1 \ \wedge\ s_1 \widehat{\approx}_{?_2} s'_1)) \text{ and}$$
$$\forall\sigma.\ ((\sigma, C_k[s'_1,\ldots,s'_k]) \to (\eta'_1, s'_1) \implies$$
$$((\sigma, C_k[s_1,\ldots,s_k]) \to (\eta_1, s_1) \ \wedge\ \eta'_1 \widehat{\approx}_{?_2} \eta_1 \ \wedge\ s'_1 \widehat{\approx}_{?_2} s_1)).$$

Let $(C_k[s_1,\ldots,s_k], C_k[s'_1,\ldots,s'_k]) \in \widehat{\approx}_{?_2}$. If the context $C_k[\cdot,\ldots,\cdot] = s_{at}$, or $C_k[\cdot,\ldots,\cdot] = E$, or $C_k[\cdot,\ldots,\cdot] \in PVar$, then the thesis follows from the reflexivity of $\approx_{?_2}$. For $C_k[\cdot,\ldots,\cdot] = [\cdot]_i$ and $s_i \approx_{?_2} s'_i$, the thesis is immediate. The only non trivial case is that of
$(op_n(C^1_k[s_1,\ldots,s_k],\ldots,C^n_k[s_1,\ldots,s_k]), op_n(C^1_k[s'_1,\ldots,s'_k],\ldots,C^n_k[s'_1,\ldots,s'_k])) \in \widehat{\approx}_{?_2}$, for $s_i \approx_{?_2} s'_i$, $i = 1,\ldots,k$.
If $(\sigma, op_n(C^1_k[s_1,\ldots,s_k],\ldots,C^n_k[s_1,\ldots,s_k])) \to$
$(g(\sigma,\eta_{j_1},\ldots,\eta_{j_l},C^1_k[s_1,\ldots,s_k],\ldots,C^n_k[s_1,\ldots,s_k]), C'_n[s_{i_1},\ldots,s_{i_k},\overline{s_{j_1}},\ldots,\overline{s_{j_l}}])$,
and $(\sigma, C^j_k[s_1,\ldots,s_k]) \to (\eta_j,\overline{s_j})$, for $j \in J = \{j_1,\ldots,j_l\}$, $(\sigma, C^i_k[s_1,\ldots,s_k]) \not\to$, for $i \in I \subseteq \{1,\ldots,n\}$, then, by induction hypothesis, $(\sigma, C^j_k[s'_1,\ldots,s'_k]) \to (\eta'_j,\overline{s'_j})$, with $\eta_j \widehat{\approx}_{?_2} \eta'_j$ and $\overline{s_j} \widehat{\approx}_{?_2} \overline{s'_j}$, for all $j \in J$, and $(\sigma, C^i_k[s'_1,\ldots,s'_k]) \not\to$, for all $i \in I$. Hence
$(\sigma, op_n(C^1_k[s'_1,\ldots,s'_k],\ldots,C^n_k[s'_1,\ldots,s'_k])) \to$
$(g(\sigma',\eta'_{j_1},\ldots,\eta'_{j_l},C^1_k[s'_1,\ldots,s'_k],\ldots,C^n_k[s'_1,\ldots,s'_k]), C'_n[s'_{i_1},\ldots,s'_{i_k},\overline{s'_{j_1}},\ldots,\overline{s'_{j_l}}])$,
where
$op_n(C^1_k[s_1,\ldots,s_k],\ldots,C^n_k[s_1,\ldots,s_k]) \widehat{\approx}_{?_2} op_n(C^1_k[s'_1,\ldots,s'_k],\ldots,C^n_k[s'_1,\ldots,s'_k])$,
and, since g is $\widehat{\approx}_{?_2}$-preserving,
$g(\sigma,\eta_{j_1},\ldots,\eta_{j_l},C^1_k[s_1,\ldots,s_k],\ldots,C^n_k[s_1,\ldots,s_k]) \widehat{\approx}_{?_2}$
$g(\sigma,\eta'_{j_1},\ldots,\eta'_{j_l},C^1_k[s'_1,\ldots,s'_k],\ldots,C^n_k[s'_1,\ldots,s'_k])$. □

Corollary 6.2.8 *If $\mathcal{T}_{?_2}$ is a transition system such that the functions g's appearing in the rules (op) are $\widehat{\approx}_{?_2}$-preserving, then $\approx_{?_2}$ is a congruence w.r.t. the syntactical operators of the language.*

Corollary 6.2.9 *Let $\mathcal{T}_{?_2}$ be a transition system such that the functions g's appearing in the rules (op) are $\widehat{\approx}_{?_2}$-preserving. Then the final semantics $\mathcal{M}_{?_2}$ is compositional, i.e., for all $s, s' \in Stat_{?_2}$,*

$$\mathcal{M}_{?_2}(s) = \mathcal{M}_{?_2}(s') \implies \forall C[\ ].\ \mathcal{M}_{?_2}(C[s]) = \mathcal{M}_{?_2}(C[s]')\ .$$

One could consider a stronger notion of compositionality than just congruence w.r.t. the operators. Namely, one could ask that, for all $s, s' \in Stat_{?_2}$, for all $\sigma, \sigma' \in Synstate_{?_2}$,

$$\mathcal{M}_{?_2}(s)(\sigma) = \mathcal{M}_{?_2}(s')(\sigma') \implies \forall C[\;].\ \mathcal{M}_{?_2}(C[s])(\sigma) = \mathcal{M}_{?_2}(C[s]')(\sigma')\ .$$

We shall not elaborate on this in general. In Section 6.3.4, we shall discuss this for the special cases of $\mathcal{L}_{pas_2}$ and $\mathcal{L}_{co_2}$.

6.2.2 An Alternative Final Description

A final description for $\mathcal{L}_{pas_2}$, alternative to the one given via the functor $F_{?_2}$ can be obtained by embedding function spaces of the shape $A \to \mathcal{P}(B)$ into spaces of relations $\mathcal{P}(A \times B)$. Applying this idea to the functor $F_{?_2}$, we obtain a new functor $H_{?_2}$. Hence we obtain another coinduction principle, independent from that corresponding to the functor $F_{?_2}$. In Section 6.3.4, we will show that for the languages $\mathcal{L}_{pas_2}$ and $\mathcal{L}_{co_2}$ the equivalence induced by the functor $H_{?_2}$ coincides with the equivalence $\approx_{?_2}$ induced by $F_{?_2}$. Hence we will obtain another coinduction principle for reasoning on $\approx_{pas_2}$ and $\approx_{co_2}$, respectively.

Definition 6.2.10 *Let $H_{?_2} : Class^*(U) \to Class^*(U)$ be the endofunctor defined by*

$$H_{?_2}(X) = \mathcal{P}_{ne}(SynState_{?_2} \times (IVar \to Val) \times (PVar \to X) \times X) + \{E\}$$
$$H_{?_2}(f) = \mathcal{P}_{ne}(id_{SynState_{?_2} \times (IVar \to Val)} \times (id_{PVar} \to f) \times f) + id_{\{E\}}.$$

Theorem 6.2.11 *The unique map from the $H_{?_2}$-coalgebra $(Stat_{?_2}, \alpha_{Stat_{?_2}})$ to the final $H_{?_2}$-coalgebra induces the equivalence $\approx^H_{?_2}$. $\approx^H_{?_2}$ is the greatest fixed point of the operator $\Phi'_{?_2} : \mathcal{P}(Stat_{?_2} \times Stat_{?_2}) \to \mathcal{P}(Stat_{?_2} \times Stat_{?_2})$ defined as follows:*

$$\Phi'_{?_2}(\mathcal{R}) = \{(s, s') \mid \forall \sigma.(((\sigma, s) \to (\eta_1, s_1) \;\Rightarrow\; \exists \sigma', \eta'_1, s'_1.(\sigma \mathcal{R} \sigma' \;\wedge\; (\sigma', s') \to (\eta'_1, s'_1) \;\wedge\; \eta_1 \mathcal{R} \eta'_1 \;\wedge\; s_1 \mathcal{R} s'_1)) \;\wedge\; ((\sigma, s') \to (\eta'_1, s'_1) \;\Rightarrow\; \exists \sigma', \eta_1, s_1.(\sigma' \mathcal{R} \sigma \;\wedge\; (\sigma', s) \to (\eta_1, s_1) \;\wedge\; \eta_1 \mathcal{R} \eta'_1 \;\wedge\; s_1 \mathcal{R} s'_1)))\}.$$

6.3 Examples

In this section we discuss two specializations of the general conditions carried out in the previous sections. We introduce

- $\mathcal{L}_{pas_2}$: an imperative concurrent language with higher order assignment;
- $\mathcal{L}_{co_2}$: an imperative concurrent language with higher order communication.

6.3.1 An Imperative Concurrent Language with Higher Order Assignment: $\mathcal{L}_{pas_2}$

The imperative concurrent language $\mathcal{L}_{pas_2}$ features a significant higher order construct: *second order assignment.* This language is obtained by adding a parallel operator to the language with second order assignment $\mathcal{L}_{as_2}$, introduced and studied by de Bakker and van Breugel in [BB93].

Definition 6.3.1 *The set of statements* $(s \in) Stat_{pas_2}$ *of the language* $\mathcal{L}_{pas_2}$ *is defined by*

$$s ::= \ E \ \mid \ v := e \ \mid \ s; s \ \mid \ s + s \ \mid \ s \,||\, s \ \mid \ x \ \mid \ x := s \ ,$$

where $v \in IVar$ *and* $x \in PVar$.

In the definition above, the atomic statements are the statements of the shape $v := e$. Notice that statements of the shape $x := s$ are not considered atomic, but compound, since $x :=$ is viewed as a unary operator.

Let $v \in IVar$, $x \in PVar$. Let the states $\sigma[\alpha/v] \in SynState_{pas_2}$ and $\sigma[s/x] \in SynState_{pas_2}$ be short for $(\pi_1(\sigma)[\alpha/v], \pi_2(\sigma))$ and $(\pi_1(\sigma), \pi_2(\sigma)[s/x])$, respectively. The transition system $\mathcal{T}_{pas_2}$ is defined as follows:

Definition 6.3.2 ($\mathcal{T}_{pas_2}$) *The transition relation* $\rightarrow$ *is the smallest subset of* $(SynState_{pas_2} \times Stat_{pas_2}) \times (SynState_{pas_2} \times Stat_{pas_2})$, *satisfying*

$(\sigma,\ v := e) \rightarrow (\sigma[\mathcal{V}(e)(\sigma)/v],\ E)\quad (:=_1)$,

where $\mathcal{V} : Expression \times Synstate_{pas_2} \rightarrow Val$ *is a function which associates values to expressions according to a syntactical state;*

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s; s') \rightarrow (\sigma_1,\ s_1; s')}\ (;_1) \qquad \frac{s \not\rightarrow \quad (\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s; s') \rightarrow (\sigma_1',\ s_1')}\ (;_2)$$

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s + s') \rightarrow (\sigma_1,\ s_1)}\ (+_1) \qquad \frac{(\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s + s') \rightarrow (\sigma_1',\ s_1')}\ (+_2)$$

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s \,||\, s') \rightarrow (\sigma_1,\ s_1 \,||\, s')}\ (||_1) \qquad \frac{(\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s \,||\, s') \rightarrow (\sigma_1',\ s \,||\, s_1')}\ (||_2)$$

$(\sigma,\ x) \rightarrow (\sigma,\ \sigma(x))\quad (var)$

$(\sigma,\ x := s) \rightarrow (\sigma[s/x],\ E)\quad (:=_2)$,

where $s \not\rightarrow$ *means that, for every* σ, *there is no transition starting from* $(\sigma,\ s)$.

The transition system $\mathcal{T}_{pas_2}$ is an instance of the general transition system $\mathcal{T}_{?_2}$, just take the function f in the rule $(:=_1)$ to be defined by
$f_{:=_1}(\sigma, v := e) = \sigma[\mathcal{V}(e)(\sigma)/v]$, and
the functions g's in the rules for the operators ;, $+$, $||$, and $:=_x$ to be defined by
$g_{op}(\sigma, \sigma_1, s, s') = \sigma_1$, for $op \in \{;, +, ||\}$
$g_{:=_x}(\sigma, s) = \sigma[s/x]$.

6.3.2 An Imperative Concurrent Language with Higher Order Communication: $\mathcal{L}_{co_2}$

The imperative concurrent language $\mathcal{L}_{co_2}$ features a significant higher order construct: *second order communication.* This language is introduced and studied by de Bakker and van Breugel in [BB93].

Definition 6.3.3 *The set of statements* $(s \in) Stat_{co_2}$ *of the language* $\mathcal{L}_{co_2}$ *is defined by*

$$s ::= E \mid v := e \mid s;s \mid s+s \mid s \,||\, s \mid x \mid c!s \mid \overline{c}?x \;,$$

where $v \in IVar$, $x \in PVar$, $c \in Chan$, *and* $\overline{c} \in \overline{Chan}$.

The atomic statements in the definition above are the statements of the shape $v := e$ and $\overline{c}?x$.

The transition system $\mathcal{T}_{co_2}$ is defined as follows:

Definition 6.3.4 ($\mathcal{T}_{co_2}$) *The transition relation* $\rightarrow$ *is the smallest subset of* $(SynState_{co_2} \times Stat_{co_2}) \times (SynState^{ext}_{co_2} \times Stat_{co_2})$, *satisfying*

$(\sigma,\ v := e) \rightarrow (\sigma[\mathcal{V}(e)(\sigma)/v],\ E)\quad (:=_1)$,

where $\mathcal{V} : Expression \times Synstate_{co_2} \rightarrow Val$ *is a function which associates values to expressions according to a syntactical state;*

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s;s') \rightarrow (\sigma_1,\ s_1;s')}\ (;_1) \qquad \frac{s \not\rightarrow \quad (\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s;s') \rightarrow (\sigma_1',\ s_1')}\ (;_2)$$

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s+s') \rightarrow (\sigma_1,\ s_1)}\ (+_1) \qquad \frac{(\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s+s') \rightarrow (\sigma_1',\ s_1')}\ (+_2)$$

$$\frac{(\sigma,\ s) \rightarrow (\sigma_1,\ s_1)}{(\sigma,\ s \,||\, s') \rightarrow (\sigma_1,\ s_1 \,||\, s')}\ (||_1) \qquad \frac{(\sigma,\ s') \rightarrow (\sigma_1',\ s_1')}{(\sigma,\ s \,||\, s') \rightarrow (\sigma_1',\ s \,||\, s_1')}\ (||_2)$$

$$\frac{(\sigma,\ s) \rightarrow ((\overline{c}, x),\ s_1) \quad (\sigma,\ s) \rightarrow ((c, s_2'),\ s_1')}{(\sigma,\ s \,||\, s') \rightarrow (\sigma[s_2'/x],\ s_1 \,||\, s_1')}\ (com)$$

$(\sigma,\ x) \rightarrow (\sigma,\ \sigma(x))\quad (var)$

$(\sigma,\ c!s) \rightarrow ((c,s),\ E)\quad (!) \qquad (\sigma,\ \overline{c}?x) \rightarrow ((\overline{c},x),\ E)\quad (?)$,

where $s \not\rightarrow$ *means that, for every* σ, *there is no transition starting from* $(\sigma,\ s)$.

The transition system $\mathcal{T}_{co_2}$ is an instance of the general transition system $\mathcal{T}_{?_2}$, just take the function f in the rule (?) to be defined by
$f_?(\sigma, \overline{c}?x) = (\overline{c}, x)$,
the function g in the rule (!) to be defined by
$g_{!_c}(\sigma, s) = (c, s)$
the function g in the rule (com) to be defined by
$g_{com}(\sigma, (\overline{c}, x), (c, s_2'), s, s') = \sigma[s_2'/x]$, and
the functions g's in the rules for ;, +, || to be defined by
$g_{op}(\sigma, \sigma_1, s, s') = \sigma_1$, for $op \in \{;, +, ||\}$.

6.3.3 Compositionality of the Final Semantics: "a Priori" Approach

We recall that the *a priori* approach for showing compositionality of final semantics consists in defining *a priori* operators coinductively and independently, on the semantical domain, and then showing that
$\widetilde{op}_n(\mathcal{M}_{?_2}(s_1), \ldots, \mathcal{M}_{?_2}(s_n)) = \mathcal{M}_{?_2}(op_n(s_1, \ldots, s_n))$, for all $s_1, \ldots, s_n$,
where $\widetilde{op}_n$ is the semantical operator corresponding to the n-ary syntactical operator op_n.

In this section, we illustrate the a priori approach for the languages $\mathcal{L}_{pas_2}$ and $\mathcal{L}_{co_2}$. One can easily check that, on the image of the final semantics, the semantical operators defined *a posteriori* for these languages coincide with the semantical operators defined *a priori.* Notice how this construction relies heavily on the coalgebraic framework.

Proposition 6.3.5 (Semantical Operators for $\mathcal{L}_{pas_2}$) *The following semantical operators are well defined:*

i) $\widetilde{;} : U_{pas_2} \times U_{pas_2} \to U_{pas_2}$

$$p \,\widetilde{;}\, q = \begin{cases} E & \textit{if } p = E \wedge q = E \\ q & \textit{if } p = E \wedge q \neq E \\ \{(\sigma,\ \{(\sigma',\ p_1 \,\widetilde{;}\, q) \mid \\ \quad (\sigma', p_1) \in p(\sigma)\}) \mid \sigma \in SynState_{pas_2}\} & \textit{otherwise.} \end{cases}$$

ii) $\widetilde{+} : U_{pas_2} \times U_{pas_2} \to U_{pas_2}$

$$p \,\widetilde{+}\, q = \begin{cases} p & \textit{if } q = E \\ q & \textit{if } p = E \\ p \cup q & \textit{otherwise.} \end{cases}$$

iii) $\widetilde{||} : U_{pas_2} \times U_{pas_2} \to U_{pas_2}$

$$p \,\widetilde{||}\, q = \begin{cases} E & \textit{if } p = E \wedge q = E \\ p & \textit{if } p \neq E \wedge q = E \\ q & \textit{if } p = E \wedge q \neq E \\ \{(\sigma,\ \{(\sigma',\ p_1 \,\widetilde{||}\, q) \mid (\sigma', p_1) \in p(\sigma)\} \cup \\ \qquad \{(\sigma',\ p \,\widetilde{||}\, q_1) \mid (\sigma', q_1) \in q(\sigma)\}) \mid \\ \qquad\qquad \sigma \in SynState_{pas_2}\} & \textit{otherwise.} \end{cases}$$

iv) $\widetilde{:=} : PVar \to (U_{pas_2} \to U_{pas_2})$ *defined by*
$\widetilde{:=}(x) = \widetilde{:=_x}$, *where*
$\widetilde{:=_x} : U_{pas_2} \to U_{pas_2}$ *is defined by*
$\widetilde{:=_x}\ p = \{(\sigma, \{((\pi_1(\sigma), \{(y, \mathcal{M}_{pas_2}(\pi_2(\sigma)(y))) \mid y \in PVar \ \wedge \ y \neq x\} \cup \{(x, p)\}), E)\}) \mid \sigma \in SynState_{pas_2}\}$.

Proof We have only to check that the operators $\widetilde{;}$ and $\widetilde{||}$ are well defined. Let $\widehat{;}$ be the unique coalgebra morphism from the F_{pas_2}-coalgebra $(U_{pas_2} \times$

$U_{pas_2},\ \beta_;)$ to the final F_{pas_2}-coalgebra $(U_{pas_2},\ id)$, where $\beta_; : U_{pas_2} \times U_{pas_2} \to F(U_{pas_2} \times U_{pas_2})$ is defined by

$$\beta_;(p,q) = \begin{cases} E & \text{if } p = E \wedge q = E \\ \{(\sigma,\ \{(\sigma'_E, (E, q_1)) \mid (\sigma', q_1) \in q(\sigma)\}) \mid \\ \qquad\qquad\qquad \sigma \in SynState_{pas_2}\} & \text{if } p = E \wedge q \neq E \\ \{(\sigma,\ \{(\sigma'_E, (p_1, q)) \mid (\sigma', p_1) \in p(\sigma)\}) \mid \\ \qquad\qquad\qquad \sigma \in SynState_{pas_2}\} & \text{otherwise,} \end{cases}$$

where $\sigma'_E \in (IVar \to Val) \times (PVar \to U_{pas_2} \times U_{pas_2})$, $\sigma'_E(v) = \sigma'(v)$, for all $v \in IVar$, and $\sigma'_E(x) = (E, \sigma'(x))$, for all $x \in PVar$.
Then $\widetilde{;}$ coincides with $\widehat{;}$, since, by finality of (U_{pas_2}, id), the coalgebra morphism $f_; : (U_{pas_2}, id) \to (U_{pas_2}, id)$, defined by $f_;(p) = E \,\widehat{;}\, p$, must be the identity on U_{pas_2}.
The operator $\widetilde{||}$ is dealt with similarly. $\square$

We are in the position of showing the compositionality of $\mathcal{M}_{pas_2}$. The proof of the following theorem is another application of the technique "by finality", which allows to factor out complexity of coinductive proofs.

Theorem 6.3.6 (Compositionality of $\mathcal{M}_{pas_2}$) *The final semantics $\mathcal{M}_{pas_2}$ is compositional.*

Proof We have to show that:

$$\begin{aligned} \mathcal{M}_{pas_2}(s) \,\widetilde{;}\, \mathcal{M}_{pas_2}(s') &= \mathcal{M}_{pas_2}(s; s') \\ \mathcal{M}_{pas_2}(s) \,\widetilde{+}\, \mathcal{M}_{pas_2}(s') &= \mathcal{M}_{pas_2}(s + s') \\ \mathcal{M}_{pas_2}(s) \,\widetilde{||}\, \mathcal{M}_{pas_2}(s') &= \mathcal{M}_{pas_2}(s \mid\mid s') \\ \widetilde{:=}_x(\mathcal{M}_{pas_2}(s)) &= \mathcal{M}_{pas_2}(x := s). \end{aligned}$$

Let $op \in \{:, +, ||\}$. Define functions $h_{op} : Stat_{pas_2} \times Stat_{pas_2} \to Stat_{pas_2}$ as follows:

$$h_{op}(s, s') = s \ op \ s' \ .$$

Then it is easy to check that the functions $\mathcal{M}_{pas_2} \circ h_{op} : (Stat_{pas_2} \times Stat_{pas_2}) \to U_{pas_2}$ and $\widetilde{op} \circ (\mathcal{M}_{pas_2} \times \mathcal{M}_{pas_2}) : (Stat_{pas_2} \times Stat_{pas_2}) \to U_{pas_2}$ coincides, since they are both morphisms from suitables F_{pas_2}-coalgebras on $(Stat_{pas_2} \times Stat_{pas_2})$ into the final F_{pas_2}-coalgebra (U_{pas_2}, id).
The unary operator $:=_x$ is dealt with similarly. $\square$

Corollary 6.3.7 A posteriori *semantical operators of $\mathcal{L}_{pas_2}$ coincide with* a priori *semantical operators on the image of $\mathcal{M}_{pas_2}$.*

The semantical operators corresponding to the language $\mathcal{L}_{co_2}$ can be defined similarly to those corresponding to the language $\mathcal{L}_{pas_2}$. In the following proposition we define only the two operators which are peculiar to $\mathcal{L}_{co_2}$.

Proposition 6.3.8 (Semantical Operators for $\mathcal{L}_{co_2}$) *The following semantical operators are well defined:*

i) $\widetilde{||} : U_{co_2} \times U_{co_2} \to U_{co_2}$

$$p \; \widetilde{||} \; q = \begin{cases} E & \textit{if } p = E \wedge q = E \\ p & \textit{if } p \neq E \wedge q = E \\ q & \textit{if } p = E \wedge q \neq E \\ \{(\sigma, Par_p \cup Par_q \cup Com_1 \cup Com_2) \mid & \\ \qquad \sigma \in SynState_{co_2}\} & \textit{otherwise,} \end{cases}$$

where

$Par_p = \{(\sigma', \; p_1 \; \widetilde{||} \; q) \mid (\sigma', p_1) \in p(\sigma)\}$

$Par_q = \{(\sigma', \; p \; \widetilde{||} \; q_1) \mid (\sigma', q_1) \in q(\sigma)\}$

$Com_1 = \{((\pi_1(\sigma), \{(y, \mathcal{M}_{co_2}(\pi_2(\sigma)(y))) \mid y \in PVar \;\wedge\; y \neq x\} \cup$
$\quad \{(x, q_2)\}), p_1 \; \widetilde{||} \; q_1) \mid ((\overline{c}, x), p_1) \in p(\sigma) \;\wedge\; ((c, q_2), q_1) \in q(\sigma)\}$

$Com_2 = \{((\pi_1(\sigma), \{(y, \mathcal{M}_{co_2}(\pi_2(\sigma)(y))) \mid y \in PVar \;\wedge\; y \neq x\} \cup$
$\quad \{(x, p_2)\}), p_1 \; \widetilde{||} \; q_1) \mid ((c, p_2), p_1) \in p(\sigma) \;\wedge\; ((\overline{c}, x), q_1) \in q(\sigma)\}.$

ii) $\widetilde{!_c} : U_{pas_2} \to U_{pas_2}$

$\widetilde{!_c} \; p = \{(\sigma, \{((c, p), E)\}) \mid \sigma \in SynState_{co_2}\}.$

Theorem 6.3.9 (Compositionality of $\mathcal{M}_{co_2}$) *The final semantics $\mathcal{M}_{co_2}$ is compositional.*

Corollary 6.3.10 A posteriori *semantical operators of $\mathcal{L}_{co_2}$ coincide with* a priori *semantical operators on the image of $\mathcal{M}_{co_2}$.*

6.3.4 Alternative Coinductive Characterizations of the Final Equivalence

In this section we discuss alternative coinductive characterizations of the final equivalences $\approx_{pas_2}$ and $\approx_{co_2}$. First of all we prove an interesting property of $\approx_{pas_2}$ ($\approx_{co_2}$), which implies that $\mathcal{M}_{pas_2}$ ($\mathcal{M}_{co_2}$) is determined "up to" $\approx_{pas_2}$($\approx_{co_2}$)-equivalent syntactical states. This is mathematically quite intriguing, since it amounts to saying that $\approx_{pas_2}$ ($\approx_{co_2}$) is a fixed point of a suitable "contravariant" operator. This property can be viewed also as a stronger form of compositionality.

Lemma 6.3.11 *i) Let $\mathcal{R} \subseteq Stat_{pas_2} \times Stat_{pas_2}$ be reflexive and symmetric. If*

$$s\mathcal{R}s' \implies \forall \sigma, \sigma', \sigma_1, s_1 \,.(\sigma\mathcal{R}\sigma' \implies ((\sigma, \; s) \to (\sigma_1, \; s_1) \implies \exists \sigma_1', s_1' \; ((\sigma', \; s') \to (\sigma_1', \; s_1') \;\wedge\; \sigma_1\mathcal{R}\sigma_1' \;\wedge\; s_1\mathcal{R}s_1'))) \; ,$$

then $\mathcal{R}$ is an F_{pas_2}-bisimulation.
Conversely, an F_{pas_2}-bisimulation which is an equivalence relation and a congruence w.r.t. the syntactical operators of $\mathcal{L}_{pas_2}$, satisfies the property above.
ii) The relation $\approx_{pas_2}$ is such that, $\forall s, s' \in Stat_{pas_2}$,

$$s \approx_{pas_2} s' \iff$$

$$\forall \sigma, \sigma', \sigma_1, s_1 \ .(\sigma \approx_{pas_2} \sigma' \implies$$

$$((\sigma,\ s) \to (\sigma_1,\ s_1) \implies \exists \sigma_1', s_1' \ ((\sigma',\ s') \to (\sigma_1',\ s_1') \ \wedge$$

$$\sigma_1 \approx_{pas_2} \sigma_1' \ \wedge \ s_1 \approx_{pas_2} s_1')) \ \wedge$$

$$((\sigma',\ s') \to (\sigma_1',\ s_1') \implies \exists \sigma_1, s_1 \ ((\sigma,\ s) \to (\sigma_1,\ s_1) \ \wedge$$

$$\sigma_1 \approx_{pas_2} \sigma_1' \ \wedge \ s_1 \approx_{pas_2} s_1'))).$$

Proof i) The first part of the proof is immediate. In order to show the converse, let $\mathcal{R}$ be an F_{pas_2}-bisimulation which is an equivalence and a congruence; let $s,\ s'$ be s.t. $s\mathcal{R}s'$. Then let $\sigma,\ \sigma'$ be s.t. $\sigma\mathcal{R}\sigma'$. If $(\sigma,\ s) \to (\sigma_1,\ s_1)$, then, since $\mathcal{R}$ is an F_{pas_2}-bisimulation, $\exists \sigma_2,\ s_2.\ ((\sigma,\ s') \to (\sigma_2,\ s_2) \wedge \sigma_1\mathcal{R}\sigma_2 \wedge s_1\mathcal{R}s_2)$. We claim that, if $(\sigma,\ s') \to (\sigma_2,\ s_2)$, then $\exists \sigma_1',\ s_1'.\ ((\sigma',\ s') \to (\sigma_1',\ s_1') \ \wedge\ \sigma_2\mathcal{R}\sigma_1' \ \wedge\ s_2\mathcal{R}s_1')$. Hence by transitivity of $\mathcal{R}$, we have the thesis. The claim is proved by induction on the length of the derivation of the judgement $(\sigma,\ s') \to (\sigma_2,\ s_2)$. The base case is immediate by definition (for *var* and $:=_1$) and using reflexivity of $\mathcal{R}$ (for $:=_2$). The induction step is dealt with by case analysis, according to the last rule applied in the derivation of $(\sigma,\ s') \to (\sigma_2,\ s_2)$, using the fact that $\mathcal{R}$ is a congruence.
ii) The proof follows from from item i of this lemma, using Proposition 6.2.3 and Corollary 6.2.8.

□

Corollary 6.3.12 *For all $s, s' \in Stat_{pas_2}$ and for all $\sigma, \sigma' \in SynState_{pas_2}$,*

$$(s \approx_{pas_2} s' \ \wedge \ \sigma \approx_{pas_2} \sigma') \implies \forall C[\ ].\mathcal{M}_{pas_2}(C[s])(\sigma) = \mathcal{M}_{pas_2}(C[s'])(\sigma')\ .$$

A similar result holds for $\mathcal{M}_{co_2}$:

Lemma 6.3.13 *i) Let $\mathcal{R} \subseteq Stat_{co_2} \times Stat_{co_2}$ be reflexive and symmetric. If*

$$s\mathcal{R}s' \implies \quad \forall \sigma, \sigma' \ .(\sigma\mathcal{R}\sigma' \implies ((\sigma,\ s) \to (\eta_1,\ s_1) \implies \exists \eta_1', s_1' \ ((\sigma',\ s') \to (\eta_1',\ s_1') \ \wedge\ \eta_1\mathcal{R}\eta_1' \ \wedge\ s_1\mathcal{R}s_1')))\ ,$$

then $\mathcal{R}$ is an F_{co_2}-bisimulation.
Conversely, an F_{co_2}-bisimulation which is an equivalence relation and a congruence w.r.t. the syntactical operators of $\mathcal{L}_{co_2}$, satisfies the property above.
ii) The relation $\approx_{co_2}$ is such that, $\forall s, s' \in Stat_{co_2}$,

$$s \approx_{co_2} s' \iff$$

$$\forall \sigma, \sigma' \ .(\sigma \approx_{co_2} \sigma' \implies$$

$$((\sigma,\ s) \to (\eta_1,\ s_1) \implies \exists \eta_1', s_1' \ ((\sigma',\ s') \to (\eta_1',\ s_1') \ \wedge$$

$$\eta_1 \approx_{co_2} \eta_1' \ \wedge \ s_1 \approx_{co_2} s_1')) \ \wedge$$

$$((\sigma',\ s') \to (\eta_1',\ s_1') \implies \exists \eta_1, s_1 \ ((\sigma,\ s) \to (\eta_1,\ s_1) \ \wedge$$

$$\eta_1 \approx_{co_2} \eta_1' \ \wedge \ s_1 \approx_{co_2} s_1'))).$$

Corollary 6.3.14 *For all $s, s' \in Stat_{co_2}$ and for all $\sigma, \sigma' \in SynState_{co_2}$,*

$$(s \approx_{co_2} s' \ \wedge \ \sigma \approx_{co_2} \sigma') \implies \forall C[\,].\ \mathcal{M}_{co_2}(C[s])(\sigma) = \mathcal{M}_{co_2}(C[s'])(\sigma') \ .$$

We conclude this section by discussing briefly the alternative final semantics obtained using the functor $H_{?_2}$ of Section 6.2.2. We sketch the proof that the equivalences $\approx_{pas_2}$ and $\approx_{co_2}$ coincide with the equivalences $\approx^H_{pas_2}$ and $\approx^H_{co_2}$, respectively. In the proof we use the coinduction principle of Theorem 6.2.11 for reasoning on $\approx_{pas_2}$ ($\approx_{co_2}$).

Lemma 6.3.15 *i) $r \approx^H_{pas_2} r' \implies s[r/x] \approx^H_{pas_2} s[r'/x]$.*
ii) If there is a derivation $\tau(\sigma) : \ (\sigma, s) \to (\sigma_1, s_1)$ in $\mathcal{T}_{pas_2}$, then, for all σ' such that $\sigma' \approx^H_{pas_2} \sigma$, there exists $\tau(\sigma') : \ (\sigma', s) \to (\sigma'_1, s'_1)$ such that $\sigma'_1 \approx^H_{pas_2} \sigma_1 \ \wedge \ s'_1 \approx^H_{pas_2} s_1$.

Proof i) Show, by induction on the complexity of s, that the following relation is an H_{pas_2}-bisimulation:

$$\mathcal{R} = \{(s[r/x], s[r'/x]) \mid s, r, r' \in Stat_{pas_2},\ r \approx^H_{pas_2} r'\}$$

ii) Use item i of this lemma. □

Theorem 6.3.16

$$\approx_{pas_2} \ = \ \approx^H_{pas_2} \ .$$

Proof The inclusion $\approx_{pas_2} \subseteq \approx^H_{pas_2}$ follows from the fact that $\approx_{pas_2}$ is an H_{pas_2}-bisimulation. In order to show the other inclusion, i.e. $\approx^H_{pas_2} \subseteq \approx_{pas_2}$, prove that $\approx^H_{pas_2}$ is an F_{pas_2}-bisimulation, using item ii of Lemma 6.3.15. □

Similar results hold for $\mathcal{L}_{co_2}$:

Lemma 6.3.17 *i) $r \approx^H_{co_2} r' \implies s[r/x] \approx^H_{co_2} s[r'/x]$.*
ii) If there is a derivation $\tau(\sigma) : \ (\sigma, s) \to (\eta_1, s_1)$ in $\mathcal{T}_{co_2}$, then, for all σ' such that $\sigma' \approx^H_{co_2} \sigma$, there exists $\tau(\sigma') : \ (\sigma', s) \to (\eta'_1, s'_1)$ such that $\eta'_1 \approx^H_{co_2} \eta_1 \ \wedge \ s'_1 \approx^H_{co_2} s_1$.

Theorem 6.3.18

$$\approx_{co_2} \ = \ \approx^H_{co_2} \ .$$

6.3.5 A Full Abstraction Result

In this section we consider the metric denotational semantics for $\mathcal{L}_{pas_2}$ *à la* de Bakker (see [Bre94, BV96]), and we compare the equivalence induced by this semantics, denoted by $\approx^{sem}_{pas_2}$, and the equivalence $\approx_{pas_2}$. We will show that $\approx^{sem}_{pas_2}$ coincides with the equivalence $\approx_{pas_2}$. It is interesting to notice that, as a consequence of this, we can characterize an equivalence defined using a contravariant functor (viz. $\approx^{sem}_{pas_2}$) using solely "covariant" tools. The proof of this result is rather complex and requires a deep analysis of the fine structure of derivations in $\mathcal{T}_{pas_2}$. We conjecture that the same result holds also for $\mathcal{L}_{co_2}$.

It would be interesting to pursue this investigation for other examples of metric semantics.

The equivalence $\approx^{sem}_{pas_2}$ is induced by the metric denotational semantics defined using as domain of processes the unique solution P of the following recursive equation in the category CMS of Complete Metric Spaces:

$$P \;\cong\; (((IVar \to Val) \times (PVar \to P_{\frac{1}{2}})) \to^1 \mathcal{P}_{nco}((IVar \to Val) \times (PVar \to P_{\frac{1}{2}}) \times P_{\frac{1}{2}})) + \{E\} \;,$$

where $P_{\frac{1}{2}}$ denotes the metric space obtained from the metric space P by contracting the distance of a factor $\frac{1}{2}$, $\to^1$ denotes the spaces of non distance increasing functions and $\mathcal{P}_{nco}$ denotes the spaces of non-empty compact subsets.

In order to show $\approx^{sem}_{pas_2} = \approx_{pas_2}$, we need some preliminary results on the metric denotational semantics. We proceed as in [BB93] for the language with second order communication $\mathcal{L}_{co_2}$. Namely, we prove that the equivalence $\approx^{sem}_{pas_2}$ coincides with the equivalence induced on $Stat_{pas_2} \times Stat_{pas_2}$ by the extended operational semantics defined using Rutten's technique of "Processes as Terms" ([Rut92]), see Definition 6.3.19 below. The equivalence induced by the extended operational semantics can be characterized coinductively using semantical environments in place of syntactical environments, process terms in place of statements, and the extended notion of transition relation defined in Definition 6.3.19 below.

Intuitively, the inclusion $\approx^{sem}_{pas_2} \subseteq \approx_{pas_2}$ is unproblematic to prove, since the denotational semantics is in general richer than the purely operational semantics (the set of semantical environments is richer than that of syntactical environments), and hence it induces, in principle, a finer notion of equivalence. The difficult part is to prove the other inclusion, i.e. $\approx_{pas_2} \subseteq \approx^{sem}_{pas_2}$. In order to get this result, we use the coinductive characterization of the equivalence induced by the extended operational semantics, and we prove that $\approx_{pas_2}$ is included in an "extended" bisimulation. Intuitively, this is possible since, in a reduction path starting from a semantical environment and a statement, we can postpone to the last reduction step uses of the rule (var) which introduce a purely semantical term. This is the content of the crucial Lemma 6.3.38.

Definition 6.3.19 *i) Let T be the set of terms defined by*

$$t ::= \; E \;\mid\; v := e \;\mid\; t;t \;\mid\; t+t \;\mid\; t \,||\, t \;\mid\; x \;\mid\; x := s \;\mid\; p \;,$$

where $p \in P$.
Let $SemState = (IVar \to Val) \times (PVar \to P)$ be the set of semantical states.
ii) Let $\mathcal{T}^{sem}_{pas_2}$ be the transition system defined as follows. The transition relation $\to_{sem}$ is the smallest subset of $(T \times SemState) \times (T \times SemState)$, satisfying

$$(\rho,\; v := e) \to_{sem} (\rho[\mathcal{V}_{sem}(e)(\rho)/v],\; E) \quad (:=_1),$$

where $\mathcal{V}_{sem} : Expression \times Semstate \to Val$ is a function which associates values to expressions according to a semantical state;

$$\frac{(\rho,\ t) \rightarrow_{sem} (\rho_1,\ t_1)}{(\rho,\ t;t') \rightarrow_{sem} (\rho_1,\ t_1;t')}\ (;_1) \qquad \frac{t \not\rightarrow_{sem} \quad (\rho,\ t') \rightarrow_{sem} (\rho_1',\ t_1')}{(\rho,\ t;t') \rightarrow_{sem} (\rho_1',\ t_1')}\ (;_2)$$

$$\frac{(\rho,\ t) \rightarrow_{sem} (\rho_1,\ t_1)}{(\rho,\ t+t') \rightarrow_{sem} (\rho_1,\ t_1)}\ (+_1) \qquad \frac{(\rho,\ t') \rightarrow_{sem} (\rho_1',\ t_1')}{(\rho,\ t+t') \rightarrow_{sem} (\rho_1',\ t_1')}\ (+_2)$$

$$\frac{(\rho,\ t) \rightarrow_{sem} (\rho_1,\ t_1)}{(\rho,\ t \mid\mid t') \rightarrow_{sem} (\rho_1,\ t_1 \mid\mid t')}\ (||_1) \qquad \frac{(\rho,\ t') \rightarrow_{sem} (\rho_1',\ t_1')}{(\rho,\ t \mid\mid t') \rightarrow_{sem} (\rho_1',\ t \mid\mid t_1')}\ (||_2)$$

$(\rho,\ x) \rightarrow_{sem} (\rho,\ \rho(x))\quad (var)$

$(\rho,\ x := t) \rightarrow_{sem} (\rho[\mathcal{D}^{\#}(t)/x],\ E)\quad (:=_2)$

$(\rho,\ p) \rightarrow_{sem} (\rho_1,\ p_1)\quad (sem) \qquad if\ p \in P\ \wedge\ (\rho_1,\ p_1) \in p(\rho),$

where $\mathcal{D}^{\#}$ denotes the extension of the metric denotational function à la *de Bakker to all terms in T.*

We will now define by finality an operational semantics $\mathcal{O}^{\#}$ on T.

Definition 6.3.20 *Let $G : CMS \rightarrow CMS$ be the endofunctor defined by*

$$G(X) = (SemState \rightarrow^1 \mathcal{P}_{nco}(SemState \times X_{\frac{1}{2}})) + \{E\}$$
$$G(f) = (id \rightarrow^1 \mathcal{P}_{nco}(id \times f)) + id.$$

Proposition 6.3.21 *The following G-coalgebra (T, β) is well defined:*

$$\beta(t) = \begin{cases} \{(\rho,\ \{(\rho_1,\ t_1) \mid (\rho,\ t) \rightarrow (\rho_1,\ t_1)\}) \mid \rho \in SemState\} & if\ t \rightarrow_{sem} \\ E & if\ t \not\rightarrow_{sem}\ . \end{cases}$$

Lemma 6.3.22 *(P, id) is the final G-coalgebra.*

Definition 6.3.23 *i) Let $\mathcal{O}^{\#} : T \rightarrow P$ be the unique morphism from the G-coalgebra (T, β) to the final G-coalgebra (P, id), i.e.*

$$\mathcal{O}^{\#}(t) = \begin{cases} \lambda\rho.\{(\rho_1,\ \mathcal{O}^{\#}(t_1)) \mid (\rho,\ t) \rightarrow_{sem} (\rho_1,\ t_1)\} & if\ t \rightarrow_{sem} \\ E & if\ t \not\rightarrow_{sem} \end{cases}$$

ii) Let $\approx^{\#}$ be the greatest G-bisimulation on the G-coalgebra (T, β).

As usual, a definition by finality induces a coinduction principle:

Definition 6.3.24 *Let $\Phi_{\#} : \mathcal{P}(T \times T) \rightarrow \mathcal{P}(T \times T)$ be the operator defined as follows:*

$\Phi_{\#}(\mathcal{R}) = \{(t, t') \mid \forall\rho.((((\rho, t) \rightarrow (\rho_1, t_1)\ \Rightarrow\ \exists t_1'.((\rho, t') \rightarrow (\rho_1, t_1')\ \wedge\ t_1 \mathcal{R} t_1'))\ \wedge$ $((\rho, t') \rightarrow (\rho_1, t_1')\ \Rightarrow\ \exists t_1.((\rho, t) \rightarrow (\rho_1, t_1)\ \wedge\ t_1 \mathcal{R} t_1')))\}.$

Proposition 6.3.25 *The equivalence induced by $\mathcal{O}^{\#}$ coincides with $\approx^{\#}$, which is the greatest fixed point of $\Phi_{\#}$.*

Following [BB93], one can prove that $\mathcal{O}^\#$ coincides with $\mathcal{D}^\#$, by endowing the space T with a suitable distance induced by the distance on P, and by showing, using an inductive argument on T, that both $\mathcal{O}^\#$ and $\mathcal{D}^\#$ are fixed points of a suitable contractive higher order operator:

Theorem 6.3.26 $\quad \mathcal{O}^\# = \mathcal{D}^\#$.

From Theorem 6.3.26 it follows immediately that the equivalence $\approx^{sem}_{pas_2}$ induced by the metric denotational semantics coincides with the equivalence $\approx^\#$ restricted to $Stat_{pas_2} \times Stat_{pas_2}$. Hence from now onwards we focus on the relation between $\approx_{pas_2}$ and $\approx^\#$.

Proof of "$\approx^{sem}_{pas_2} \subseteq \approx_{pas_2}$".

Lemma 6.3.27 *i) Let $\sigma, \sigma_1 \in SynState_{pas_2}$, $s, s_1 \in Stat_{pas_2}$; if $(\sigma,\ s) \to (\sigma_1,\ s_1)$, then there exists $t_1 \in T$ s.t. $(\rho,\ s) \to_{sem} (\rho_1,\ t_1)$, where $\mathcal{O}^\#(\sigma) = \rho$, $\mathcal{O}^\#(\sigma_1) = \rho_1$ and $\mathcal{O}^\#(s_1) = \mathcal{O}^\#(t_1)$ (by abuse of notation, we use $\mathcal{O}^\#$ also for the natural extension to $SynState_{pas_2}$).*
ii) Let $s \in SynState_{pas_2}$, $t_1 \in T$, and let $\rho, \rho_1 \in SemState$ be s.t. there exists $\sigma \in Synstate_{pas_2}$ s.t. $\mathcal{O}^\#(\sigma) = \rho$. If $(\rho,\ s) \to_{sem} (\rho_1,\ t_1)$, then there exist $\sigma_1 \in Synstate_{pas_2}$, $s_1 \in Stat_{pas_2}$ s.t. $(\sigma,\ s) \to (\sigma_1,\ s_1)$, $\mathcal{O}^\#(\sigma_1) = \rho_1$ and $\mathcal{O}^\#(s_1) = \mathcal{O}^\#(t_1)$.

Proof i) Proceed by induction on the length of the derivation of the judgement $(\sigma,\ s) \to (\sigma_1,\ s_1)$ in the transition system.
ii) Proceed by induction on the length of the derivation of $(\rho,\ s) \to_{sem} (\rho_1,\ t_1)$ in the $\to_{sem}$-transition system. $\square$

Theorem 6.3.28 *The equivalence $\approx^{sem}_{pas_2}$ induced by the metric denotational semantics is not coarser than the equivalence $\approx_{pas_2}$ induced by the final semantics $\mathcal{M}_{pas_2}$, i.e.*

$$\approx^{sem}_{pas_2} \subseteq \approx_{pas_2} \quad .$$

Proof Using items i and ii of Lemma 6.3.27, one can easily show that $\{(s, s') \mid \mathcal{O}^\#(s) = \mathcal{O}^\#(s')\}$ is a F_{pas_2}-bisimulation on the coalgebra $(Stat_{pas_2}, \alpha_{Stat_{pas_2}})$. $\square$

Proof of "$\approx_{pas_2} \subseteq \approx^{sem}_{pas_2}$".

This proof is rather complicated and it requires a number of results about $\mathcal{O}^\#$, some of which are interesting in their own right. First, we need another coinduction principle for $\approx^\#$.

Definition 6.3.29 *Let $\Phi'_\# : \mathcal{P}(T \times T) \to \mathcal{P}(T \times T)$ be the operator defined as:*

$\Phi'_\#(\mathcal{R}) = \{(t, t') \mid \forall \rho, \rho_1, t_1.(((\rho, t) \to (\rho_1, t_1) \ \Rightarrow\ \exists \rho'_1, t'_1.((\rho, t') \to (\rho'_1, t'_1) \ \wedge\ \rho_1 \mathcal{R} \rho'_1 \ \wedge\ t_1 \mathcal{R} t'_1)) \ \wedge\ ((\rho, t') \to (\rho'_1, t'_1) \ \Rightarrow\ \exists \rho_1, t_1.((\rho, t) \to (\rho_1, t_1) \ \wedge\ \rho_1 \mathcal{R} \rho'_1 \ \wedge\ t_1 \mathcal{R} t'_1)))\}$.

Lemma 6.3.30 *$\approx^{\#}$ is the greatest fixed point of $\Phi'_{\#}$.*

Proof Since for all $p \in P$, $\mathcal{O}^{\#}(p) = p$, then $\mathcal{O}^{\#}$ is s.t.

$$\mathcal{O}^{\#}(t) = \begin{cases} \lambda\rho.\{(\mathcal{O}^{\#}(\rho_1),\ \mathcal{O}^{\#}(t_1)) \mid (\rho,\ t) \rightarrow_{sem} (\rho_1,\ t_1)\} & \text{if } t \rightarrow_{sem} \\ E & \text{if } t \not\rightarrow_{sem}\ , \end{cases}$$

i.e. $\mathcal{O}^{\#}$ is the unique G'-coalgebra map from (T, β) to the final G'-coalgebra (P, id), where $G' : CMS \rightarrow CMS$ is defined as follows
$G'(X) = (SemState \rightarrow^1 \mathcal{P}_{nco}((IVar \rightarrow Val) \times (PVar \rightarrow X_{\frac{1}{2}}) \times X_{\frac{1}{2}})) + \{E\}$
$G'(f) = (id \rightarrow^1 \mathcal{P}_{nco}(id \times (id \rightarrow f) \times f)) + id.$ $\square$

Definition 6.3.31 (Substitution) *Let $s, r \in Stat_{pas_2}$. Let $s[r/x] \in Stat_{pas_2}$ be defined inductively as follows:*

- $s = E$ *or* $s = (v := e)$ *or* $s = (y := r')$ $\implies$ $s[r/x] = s;$
- $s = x$ $\implies$ $s[r/x] = r;$
- $s = s_1; s_2$ $\implies$ $s[r/x] = s_1[r/x]; s_2[r/x];$
- $s = s_1 + s_2$ $\implies$ $s[r/x] = s_1[r/x] + s_2[r/x];$
- $s = s_1 \mid\mid s_2$ $\implies$ $s[r/x] = s_1[r/x] \mid\mid s_2[r/x].$

Lemma 6.3.32 *Let $s, s', r, r' \in Stat_{pas_2}$. If $s \approx_{pas_2} s'$ and $r \approx_{pas_2} r'$, then $s[r/x] \approx_{pas_2} s'[r'/x]$.*

Proof Analyzing the fine structure of derivations starting from s, one can show that the relation $\mathcal{R} = \approx_{pas_2} \cup \{(s[r/x], s'[r/x]) \mid s, s', r \in Stat_{pas_2},\ s \approx_{pas_2} s'\}$ is an F_{pas_2}-bisimulation on $(Stat_{pas_2}, \alpha_{Stat_{pas_2}})$. Then, using Corollary 6.2.8 and transitivity of $\approx_{pas_2}$, we conclude the proof. $\square$

A very detailed analysis of the fine structure of derivations in $\mathcal{T}^{sem}_{pas_2}$ is necessary to prove the next theorem.

In the proof of $\mathcal{R} \subseteq \Phi'_{\#}(\mathcal{R})$, necessary to establish Theorem 6.3.39 below, one proceeds by an involved case analysis, making use of the following results.

First of all notice that all derivations in $\mathcal{T}_{pas_2}$ are finite trees with only one leaf, which is either a $:=_1$ axiom or a $:=_2$ axiom or a *var* axiom.

Definition 6.3.33 *Two derivations in $\mathcal{T}_{pas_2}$, $\tau : (\sigma, s) \rightarrow (\sigma_1, s_1)$ and $\tau' : (\sigma', s) \rightarrow (\sigma'_1, s'_1)$ are* strongly similar *if both are obtained instantiating the same sequence of rule schemata.*

Statements which are related by $\approx_{pas_2}$ generate derivations in $\mathcal{T}_{pas_2}$, which are related in a precise sense:

Definition 6.3.34 *Two derivations τ and τ' are $\approx_{pas_2}$-*corresponding *if*

- *τ and τ' use the same axiom,*
- *the objects occurring in corresponding positions in the conclusions of τ and τ' are $\approx_{pas_2}$-related, and*
- *if the axiom used in τ and τ' is var, then it is instantiated to the same variable both in τ and in τ'.*

Lemma 6.3.35 *Let $s, s' \in Stat_{pas_2}$ be such that $s \approx_{pas_2} s'$, and let $\sigma \in SynState_{pas_2}$. If there is a derivation $\tau(\sigma) : (\sigma, s) \to (\sigma_1, s_1)$, then there exists a $\approx_{pas_2}$-corresponding derivation $\tau'(\sigma) : (\sigma, s') \to (\sigma'_1, s'_1)$.*

Proof By case analysis on the axiom used in $\tau(\sigma)$.

1) Suppose that there exists a derivation

$$\tau(\sigma): \quad \frac{\begin{array}{c}(\sigma, x := r) \to (\sigma[r/x], E)\\ \vdots\end{array}}{(\sigma, s) \to (\sigma[r/x], s_1)} \quad ,$$

then, for all σ', we have a derivation $\tau(\sigma')$ strongly similar to $\tau(\sigma)$.

Now if we choose $\overline{\sigma}$ such that $\overline{\sigma}(x) \not\approx_{pas_2} r$, since $s \approx_{pas_2} s'$, there must exists a derivation

$$\tau'(\overline{\sigma}): \quad \frac{\begin{array}{c}(\overline{\sigma}, x := r') \to (\overline{\sigma}[r'/x], E)\\ \vdots\end{array}}{(\overline{\sigma}, s') \to (\overline{\sigma}[r'/x], s'_1)} \quad ,$$

such that $r \approx_{pas_2} r'$ and $s_1 \approx_{pas_2} s'_1$. But then, for all σ', we have a derivation $\tau'(\sigma') : (\sigma', s') \to (\sigma'[r'/x], s'_1)$ which is strongly similar to $\tau'(\overline{\sigma}')$ such that $s_1 \approx_{pas_2} s'_1$.

2) Suppose that there exists a derivation

$$\tau(\sigma): \quad \frac{\begin{array}{c}(\sigma, x) \to (\sigma, \sigma(x))\\ \vdots\end{array}}{(\sigma, s) \to (\sigma, s_1)} \quad ,$$

then, for all σ', we have a derivation $\tau(\sigma')$ strongly similar to $\tau(\sigma)$.
Let $z \in PVar$ be such that z appears neither in s nor in s'. Choose $\overline{\sigma}$ such that $\overline{\sigma}(x) = z$, $\overline{\sigma}(z) = (\overline{v} := \overline{val})$, $\overline{v}$ appears neither in s nor in s', $\forall y \neq z, x.\ \overline{\sigma}(y) = (\overline{v}' := \overline{val}')$, $\overline{v}'$ appears neither in s nor in s', $\overline{v} \neq \overline{v}'$, $\forall v \neq \overline{v}, \overline{v}'.\ \overline{\sigma}(v) = val_v$, $\forall v.val_v \neq \overline{val}, \overline{val}'$, if $v := e$ appears either in s or in s' then $\mathcal{V}(e)(\overline{\sigma}) \neq val_v$. Then, since $s \approx_{pas_2} s'$, one can easily check that there must exist a derivation

$$\tau'(\overline{\sigma}): \quad \frac{\begin{array}{c}(\overline{\sigma}, x) \to (\overline{\sigma}, \overline{\sigma}(x))\\ \vdots\end{array}}{(\overline{\sigma}, s') \to (\overline{\sigma}, \overline{s}'_1)}$$

such that $\overline{s}_1 \approx_{pas_2} \overline{s}'_1$, where $\overline{s}_1$ is the statement reached by $\tau(\overline{\sigma}) : (\overline{\sigma}, s) \to (\overline{\sigma}, \overline{s}_1)$. Put $\overline{s}_1 = \overline{s}(z/x)$ and $\overline{s}'_1 = \overline{s}'(z/x)$, where $\overline{s}(z/x)$ ($\overline{s}'(z/x)$) denotes the statement $\overline{s}$ ($\overline{s}'$) in which the variable z is substituted for the occurrence of x reduced in the first step of the derivation $\tau(\overline{\sigma})$ ($\tau'(\overline{\sigma})$).

But then, for all σ', we have a derivation $\tau'(\sigma')$ strongly similar to $\tau'(\overline{\sigma})$. We will now show that for all σ', $\tau'(\sigma')$ is a derivation such that $\widetilde{s}_1 \approx_{pas_2} \widetilde{s}'_1$, where $\widetilde{s}_1$, $\widetilde{s}'_1$ are such that $\tau(\sigma') : (\sigma', s) \to (\sigma', \widetilde{s}_1)$ and $\tau'(\sigma') : (\sigma', s') \to (\sigma', \widetilde{s}'_1)$. In fact, for all σ', we have: $\widetilde{s}_1 = \overline{s}(\sigma'(x)/x)_1$ and $\widetilde{s}'_1 = \overline{s}'(\sigma'(x)/x)_1$, i.e. $\widetilde{s}_1 = \overline{s}(z/x)_1[\sigma'(x)/z]$ and $\widetilde{s}'_1 = \overline{s}'(z/x)_1[\sigma'(x)/z]$. Hence, being $\overline{s}(z/x)_1 \approx_{pas_2} \overline{s}'(z/x)_1$, from Lemma 6.3.32, $\forall \sigma'. \ \overline{s}(z/x)_1[\sigma'(x)/z] \approx_{pas_2} \overline{s}'(z/x)_1[\sigma'(x)/z]$.

3) Finally, if the axiom used in the derivation $\tau(\sigma)$ is $:=_1$, then the thesis follows from items 1 and 2, by symmetry of $\approx_{pas_2}$. $\square$

Syntactical derivations starting from a given statement are related to semantical derivations starting from the same statement in a precise sense:

Definition 6.3.36 *Let $\tau(\sigma) : (\sigma, s) \to (\sigma_1, s_1)$ be a derivation in $\mathcal{T}_{pas_2}$. The derivation $\tau(\rho) : (\rho, s) \to (\rho_1, t_1)$ in $\mathcal{T}^{sem}_{pas_2}$ is strongly similar to $\tau(\sigma)$ if both are obtained instantiating the same sequence of rule schemata.*

Lemma 6.3.37 *Let $s \in Stat_{pas_2}$. Then, for all $\sigma \in SynState_{pas_2}$ and $\rho \in SemState$, there exists a syntactical derivation $\tau(\sigma)$ of $(\sigma, s) \to (\sigma_1, s_1)$ if and only if there exists a semantical derivation $\tau(\rho)$ of $(\rho, s) \to (\rho_1, t_1)$ strongly similar to $\tau(\sigma)$.*

Proof Straightforward, since the only extra axiom in the system $\mathcal{T}^{sem}_{as_2}$ cannot appear in derivations involving syntactical statements. $\square$

Consider a derivation step, starting from a statement s, in the semantical transition system $\mathcal{T}^{sem}_{pas_2}$: notice that there is only one case (when the axiom used is *var*), in which we reach a not purely syntactical term. We need a last technical lemma which deals with semantical derivations starting from a given statement. It allows to postpone the introduction of a non purely syntactical term, by replacing a pair of semantical steps with a suitable sequence consisting of syntactical steps but the last one:

Lemma 6.3.38 (Sem-Postponement) *i) For all $\rho, \rho', \rho'' \in SemState$, $p \in P$, $t_1, t_2 \in T$, $s \in Stat_{pas_2}$, $x \in PVar$, if there exist derivations*

$$\tau_1(\rho): \quad \frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x)) \\ \vdots\end{array}}{(\rho, s) \to_{sem} (\rho, t_1)} \qquad \text{and} \qquad \tau_2(\rho'): \quad \frac{\begin{array}{c}(\rho', \rho(x)) \to_{sem} (\rho'', p) \\ \vdots\end{array}}{(\rho', t_1) \to_{sem} (\rho'', t_2)} \ ,$$

where the axiom used in $\tau_2(\rho')$ is sem,

then, for all $\sigma \in SynState_{pas_2}$, $\rho''' \in SemState$, $s_1 \in Stat_{pas_2}$, s.t. $\sigma(x) = x$ and $\rho'''(x) = p$, there exist derivations

- $\tau_1(\sigma)$: $$\frac{\begin{array}{c}(\sigma, x) \to (\sigma, x)\\ \vdots\end{array}}{(\sigma, s) \to (\sigma, s_1)}$$

  s.t. $\tau_1(\rho)$ *is strongly similar to* $\tau_1(\sigma)$, *and*

- $\tau_2(\rho''')$: $$\frac{\begin{array}{c}(\rho''', x) \to_{sem} (\rho''', \rho'''(x))\\ \vdots\end{array}}{(\rho''', s_1) \to_{sem} (\rho''', t_2)} \quad ,$$

  where the occurrence of x *reduced in* $\tau_2(\rho''')$ *corresponds to the one reduced in* $\tau_1(\sigma)$.

Vice versa, for all $\sigma \in SynState_{pas_2}$, $\rho''' \in SemState$, $s, s_1 \in Stat_{pas_2}$, $t_2 \in T$, $x \in PVar$, *s.t.* $\sigma(x) = x$, *if there exist derivations*

$$\tau_1(\sigma): \quad \frac{\begin{array}{c}(\sigma, x) \to (\sigma, x)\\ \vdots\end{array}}{(\sigma, s) \to (\sigma, s_1)} \qquad \textit{and} \qquad \tau_2(\rho'''): \quad \frac{\begin{array}{c}(\rho''', x) \to_{sem} (\rho''', \rho'''(x))\\ \vdots\end{array}}{(\rho''', s_1) \to_{sem} (\rho''', t_2)} \quad ,$$

where the occurrence of x *reduced in* $\tau_2(\rho''')$ *corresponds to the one reduced in* $\tau_1(\sigma)$,

then, for all $\rho, \rho', \rho'' \in SemState$, $t_1 \in T$, *s.t.* $(\rho'', \rho'''(x)) \in (\rho(x))(\rho')$, *there exist derivations*

- $\tau_1(\rho)$: $$\frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x))\\ \vdots\end{array}}{(\rho, s) \to_{sem} (\rho, t_1)}$$

  s.t. $\tau_1(\rho)$ *is strongly similar to* $\tau_1(\sigma)$, *and*

- $\tau_2(\rho')$: $$\frac{\begin{array}{c}(\rho', \rho(x)) \to_{sem} (\rho'', \rho'''(x))\\ \vdots\end{array}}{(\rho', t_1) \to_{sem} (\rho'', t_2)} \quad ,$$

  where the axiom used in $\tau_2(\rho')$ *is* sem.

ii) For all $\rho, \rho', \rho'' \in SemState$, $s \in Stat_{pas_2}$, $t_1, t_2 \in T$, $x \in PVar$, *if there exist derivations*

$$\tau_1(\rho): \quad \frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x))\\ \vdots\end{array}}{(\rho, s) \to_{sem} (\rho, t_1)} \qquad \textit{and} \qquad \tau_2(\rho'): \quad \frac{\vdots}{(\rho', t_1) \to_{sem} (\rho'', t_2)} \quad ,$$

where the axiom used in $\tau_2(\rho')$ *is not* sem, *then, for all* $\sigma, \sigma', \sigma'' \in SynState_{pas_2}$, $s_1, s_2 \in Stat_{pas_2}$, *s.t.* $\sigma(x) = x$, *there exist derivations*

- $\tau_1(\sigma)$: $$\frac{\begin{array}{c}(\sigma, x) \to (\sigma, x)\\ \vdots\end{array}}{(\sigma, s) \to (\sigma, s_1)}$$

  s.t. $\tau_1(\rho)$ *is strongly similar to* $\tau_1(\sigma)$,

- $\tau_2(\sigma')$: $$\frac{\vdots}{(\sigma', s_1) \to (\sigma'', s_2)} \quad ,$$

  where the sequence of rule schemata instantiated in $\tau_2(\sigma')$ *is the same as the sequence of rule schemata instantiated in* $\tau_2(\rho')$, *and*

- $\tau_3(\rho)$: $$\frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x))\\ \vdots\end{array}}{(\rho, s_2) \to_{sem} (\rho, t_2)} \quad ,$$

  where the occurrence of x *reduced in* $\tau_3(\rho)$ *corresponds to the occurrence reduced in* $\tau_1(\rho)$.

Vice versa, for all $\sigma, \sigma', \sigma'' \in SynState_{pas_2}$, $\rho \in SemState$, $s, s_1, s_2 \in Stat_{pas_2}$, $t_2 \in T$, $x \in PVar$, *s.t.* $\sigma(x) = x$, *if there exist derivations*

- $\tau_1(\sigma)$: $$\frac{\begin{array}{c}(\sigma, x) \to (\sigma, x)\\ \vdots\end{array}}{(\sigma, s) \to (\sigma, s_1)} \quad , \qquad \tau_2(\sigma'): \quad \frac{\vdots}{(\sigma', s_1) \to (\sigma'', s_2)} \quad ,$$

  where the axiom used in $\tau_2(\sigma')$ *does not involve the occurrence of* x *reduced in* $\tau_1(\sigma)$, *and*

- $\tau_3(\rho)$: $$\frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x))\\ \vdots\end{array}}{(\rho, s_2) \to_{sem} (\rho, t_2)} \quad ,$$

  where the occurrence of x *reduced in* $\tau_3(\rho)$ *corresponds to the occurrence reduced in* $\tau_1(\sigma)$,

then, for all $\rho', \rho'' \in SemState$, $t_1 \in T$, *there exist derivations*

- $\tau_1(\rho)$: $$\frac{\begin{array}{c}(\rho, x) \to_{sem} (\rho, \rho(x))\\ \vdots\end{array}}{(\rho, s) \to_{sem} (\rho, t_1)}$$

s.t. $\tau_1(\rho)$ *is strongly similar to* $\tau_1(\sigma)$*, and*

- $\tau_2(\rho')$: $\dfrac{\vdots}{(\rho', t_1) \rightarrow_{sem} (\rho'', t_2)}$,

  where the sequence of rule schemata instantiated in $\tau_2(\rho')$ *is the same as the sequence of rule schemata instantiated in* $\tau_2(\sigma')$*.*

Proof Proceed by case analysis, according to the axiom used in the derivation $\tau_2(\rho')$ ($\tau_2(\sigma')$), and using Lemma 6.3.37. $\square$

Theorem 6.3.39 $\approx_{pas_2} \subseteq \approx^{\#}$.

Proof (Sketch) The proof uses the coinduction principle of Lemma 6.3.30. Define a relation $\mathcal{R} \subseteq T \times T$, including $\approx_{pas_2}$, as follows:

$$\mathcal{R} \;=\; \approx_{pas_2} \;\cup\; \{(\mathcal{O}^{\#}(s), \mathcal{O}^{\#}(s')) \mid s \approx_{pas_2} s'\} \;\cup\; Id_{P \times P} \;\cup\; \mathcal{R}^{\#} \;,$$

where $\mathcal{R}^{\#}$ is a relation on $(T \setminus Stat_{pas_2}) \times (T \setminus Stat_{pas_2})$ defined as follows:

$\mathcal{R}^{\#} \;=\; \{(t_1, t_1') \mid \exists s_1, s_1' \in Stat_{pas_2}.$
$\exists \tau(\rho) :\; (\rho, s) \rightarrow_{sem} (\rho_1, t_1).\; \exists \tau'(\rho) :\; (\rho, s') \rightarrow_{sem} (\rho_1', t_1').$
$\exists \tau(\sigma) :\; (\sigma, s) \rightarrow (\sigma_1, s_1).\; \exists \tau'(\sigma) :\; (\sigma, s') \rightarrow_{sem} (\sigma_1, s_1')$ s.t.
$(\tau(\sigma)$ and $\tau'(\sigma)$ are $\approx_{pas_2}$-corresponding $\wedge$
$\tau(\rho)$ is strongly similar to $\tau(\sigma) \;\wedge\; \tau'(\rho)$ is strongly similar to $\tau'(\sigma))\}$.

One can show that $\mathcal{R}$ is a $\Phi'_{\#}$-bisimulation, using Lemmata 6.3.32, 6.3.35, and 6.3.38. $\square$

Finally, we have:

Theorem 6.3.40 (Full Abstraction)

$$\approx_{pas_2} \;=\; \approx^{sem}_{pas_2} \;.$$

Chapter 7

Untyped λ-Calculus

In this chapter we discuss the final semantics for the untyped λ-calculus. We expand and generalize [HL95, Len97, Len97a, HL9?]. We investigate two alternative ways of providing a final description of various λ-equivalences and λ-theories, which arise from two different ways of construing the *denotation* of a λ-term as a *function* over syntactic objects, or as a *functional collection of pairs*, respectively. Since we want to provide final accounts of specific interesting λ-theories, we discuss conditions under which these descriptions are *congruences* and *fully-abstract* w.r.t. the corresponding λ-theory. The interest of the two final descriptions lies in the fact that they explain on general grounds different coinduction principles for reasoning on λ-terms, and in the case of the functor G (of Definition 7.2.2), even suggest genuinely new principles.

In Section 7.1, we collect some standard basic definitions on λ-calculus, and we introduce some notation. In particular, we introduce the key-notions of λ-*congruence* and of *applicative equivalence* induced by a λ-congruence. In Section 7.2, we define the functors F and G, which provide two alternative final descriptions of the λ-calculus endowed with various λ-theories. In particular, given a λ-theory $\approx_\sigma$, we show that the equivalence induced by the final semantics provided by the functor F is exactly the applicative equivalence corresponding to $\approx_\sigma$. Hence, in order to get a final description of the original λ-theory $\approx_\sigma$, we need to prove that $\approx_\sigma$ coincides with the corresponding applicative equivalence, $\approx_\sigma^{app}$. This issue is addressed in Section 7.5, where we present various techniques for proving $\approx_\sigma^{app}=\approx_\sigma$. Proving this equality amounts to show that $\approx_\sigma^{app}$ is a *congruence* both w.r.t. syntactical application and λ-abstraction, but, as we will see, the problematic part is to prove that $\approx_\sigma^{app}$ is a *congruence* w.r.t. application. In Section 7.5, we present in detail three techniques for proving this latter fact, which apply to all the examples of λ-theories presented in Section 7.3:

- The *Congruence Candidate Method*. This method is based on the definition of a *candidate relation*, which is a *congruence* w.r.t. application, and which extends the applicative equivalence. In particular, if the candidate relation is an F-bisimulation on the coalgebra induced by the λ-theory $\approx_\sigma$, then we get that the applicative equivalence coincides with the candidate relation, and hence the applicative equivalence itself is a congruence w.r.t. application. We show that, under suitable hypotheses on the reduction strategy, this happens for all the λ-theories which arise from reduction strategies which are instances of one of the following three *general formats*:

1. *lazy strategies*
2. *eager leftmost strategies*
3. *non-deterministic strategies.*

Hence, in particular we get *uniform* proofs of the fact that the applicative equivalence is a congruence w.r.t. application, for all the λ-theories of Section 7.3. The congruence candidate method was introduced by D.Howe for lazy strategies (see [How96]). In [Len97a], we generalize and strengthen Howe's method, in order to deal *uniformly* with all the strategies which are instances of the three general formats above. In this thesis (Section 7.6.2), we expand [Len97a].

- The *Logical Relations Method.* This method is *semantical*, in the sense that it makes use of a suitable $\approx_\sigma$*-computationally adequate CPO-*λ-model. The core of the logical relations method consists in proving a *mixed semantical-syntactical induction-coinduction principle* for the model. This method was originally introduced by Pitts (see [Pit94, Pit96]) for the case of the observational equivalences which arise from lazy strategies. Pitts' technique is based on Plotkin's *minimal invariance property* of the model, as is the case for *initial models.* In [Len97], we generalize Pitts' method also to non-initial models, using a different technique for proving the induction-coinduction principle. This technique is inspired by [EHR92], where a related method is presented for proving that the applicative equivalence corresponding to the observational equivalence $\approx_v$ induced by the lazy by-value strategy (see Section 7.3) is a congruence w.r.t. application. This technique is based on the λ-*definability* of the projection functions in the inverse limit model D^v, and hence it is extendible only to the model D^o of Section 7.4. An interesting by-product of this technique is a *purely syntactical induction-coinduction principle* for the applicative equivalence (see [HL95]). In this thesis (Section 7.6.4) we generalize and expand [Len97], by axiomatizing a set of general properties of computationally adequate models, which make the method work. In particular, we introduce the crucial notion of *finitary* $\approx_\sigma$*-model*, and we prove that, for all λ-theories which have a computationally adequate finitary model, the applicative equivalence is a congruence w.r.t. application. The notion of finitary model is quite general, in fact, as we will see, many classical and non classical λ-theories have a computationally adequate finitary model.

- The *Domain Logic Method*. This is a *semantical* method, based on the logical description via *intersection types* of a *computationally adequate* CPO-λ-model (see Section 7.4). This method was originally introduced by Abramsky and Ong in [AO93] for the observational equivalence induced by the lazy strategy. In this thesis (Section 7.6.4), we carry out a deep analysis of the method, and we work out a general theory, which improves [AO93] in two main respects:

  1. in place of λ-model, we introduce the more general notion of *σ-combinatory algebra*;
  2. we introduce the notion of *adequacy* for a type interpretation on a *σ-applicative structure*, while in [AO93] a uniform definition of type interpretation on applicative σ-structures is required.

  This new formulation of the method is much stronger of that in [AO93], and up to a first approximation, it allows to show that the applicative equivalence is a congruence w.r.t. application for all λ-theories such that (see Proposition 7.6.54 of Section 7.6.4):

  - there exists a *computationally adequate intersection types* model which is a *combinatory σ-algebra*;
  - the quotient of the set of λ-terms by a suitable $\approx_\sigma$*-correct* notion of β-reduction can be endowed with a *combinatory σ-algebra* structure, on which a *sound and adequate* type interpretation is definable.

  The conditions above are satisfied by many interesting λ-theories, in particular, we show in Section 7.6.4 that they are satisfied by all the λ-theories presented in Section 7.3. The content of Section 7.6.4 expands and clarifies [Len97], where the generalization of Abramsky and Ong's method is presented, but the examples are not worked out.

7.1 Basic Definitions

Let $\Lambda(C)$ denote the class of λ-terms built over the set of basic constants C, i.e.:

$$(\Lambda(C) \ni)\ M \ ::= \ x \ \mid \ c \ \mid \ MM \ \mid \ \lambda x.M \ ,$$

where $x \in Var$, $c \in C$.
Let $\Lambda^0(C)$ denote the class of *closed* λ-terms built over the set of basic constants C. If $C = \emptyset$, then $\Lambda(C)$ $(\Lambda^0(C))$ will be denoted simply by Λ (Λ^0).

Definition 7.1.1 • *A* λ-pre-equivalence *is a reflexive and transitive relation on* $\Lambda(C) \times \Lambda(C)$.

- *A* λ-equivalence *is a symmetric* λ*-pre-equivalence.*

- *A* λ-pre-congruence, $\leq_\sigma$[1], *is a* λ*-pre-equivalence which is a congruence w.r.t. application and* λ*-abstraction, i.e., for all* $M, N, M', N' \in \Lambda(C)$,

$$M \leq_\sigma N \ \wedge \ M' \leq_\sigma N' \implies MM' \leq_\sigma NN' \ , \text{ and}$$

$$M \leq_\sigma N \implies \lambda x.M \leq_\sigma \lambda x.N \ .$$

- *A* λ-congruence *is a* λ*-pre-congruence which is a* λ*-equivalence.*

- *A* λ-theory *is the restriction of a* λ*-congruence to* $\Lambda^0(C)$.

The following proposition gives a useful characterization of λ-pre-congruences and λ-congruences ([HR92]):

Proposition 7.1.2 • *Any* λ*-pre-congruence* $\leq_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ *is induced by a suitable set of* σ-valuable *terms* $\mathcal{V}_\sigma \subseteq \Lambda^\circ(C)$[2]*, in the following sense:*

$$M \leq_\sigma N \iff \forall C[\,].\ (C[M], C[N] \in \Lambda^\circ(C) \Longrightarrow (C[M] \in \mathcal{V}_\sigma \Longrightarrow C[N] \in \mathcal{V}_\sigma))\ .$$

• *Any* λ*-congruence* $\approx_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ *is induced by a suitable set* $\mathcal{V}_\sigma \subseteq \Lambda^\circ(C)$*, in the following sense:*

$$M \approx_\sigma N \iff \forall C[\,].\ (C[M], C[N] \in \Lambda^\circ(C) \Longrightarrow (C[M] \in \mathcal{V}_\sigma \iff C[N] \in \mathcal{V}_\sigma))\ .$$

Notation The fact that $M \in \mathcal{V}_\sigma$ will be also denoted by $M \Downarrow_\sigma$.

The relation between λ-theories and β-reduction is formalized and clarified by Proposition 7.1.5 below.

Definition 7.1.3 *A notion of* β-reduction, $\rightarrow_{\beta_r}$, *is the* λ*-pre-congruence generated by a set of pairs (redex,* β*-reduct). The* λ*-congruence generated by the symmetric closure of a* β*-reduction* $\rightarrow_{\beta_r}$ *is the* conversion $=_{\beta_r}$.

Definition 7.1.4 *A notion of* β*-reduction,* $\rightarrow_{\beta_r}$*, is* correct *w.r.t. a* λ*-congruence* $\approx_\sigma$ *if* $=_{\beta_r} \subseteq \approx_\sigma$.

Proposition 7.1.5 *Let* $\approx_\sigma$ *be a* λ*-congruence. Then the notion of* β*-reduction* $\rightarrow_{\beta_r}$ *is correct w.r.t.* $\approx_\sigma$ *if and only if* $\mathcal{V}_\sigma$ *is closed under* β_r*-conversion, i.e.* $(=_{\beta_r} \cap (\Lambda^0(C) \times \Lambda^0(C))) \subseteq \mathcal{V}_\sigma \times \mathcal{V}_\sigma$.

In the sequel, we will often need the following specific notions, notations, and definitions.

Definition 7.1.6 *Let* $\leq_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ *be a* λ*-pre-congruence. We say that:*

- λ*-term application is* left strict *w.r.t.* $\Downarrow_\sigma$*, or simply* $\leq_\sigma$ *is* left strict *if, for all* $M \in \Lambda^0$,

$$M \notin \mathcal{V}_\sigma \Rightarrow \forall N \in \Lambda^0.\ MN \notin \mathcal{V}_\sigma\ ;$$

[1] We use the index σ in the definition of a λ-pre-congruence, since, as we will see in Section 7.3, the main example of a λ-pre-congruence is that determined by a reduction strategy $\rightarrow_\sigma$.

[2] The meaning of these terms will be clear in Section 7.3.

- *λ-term application is* right strict *w.r.t.* $\Downarrow_\sigma$*, or simply* $\leq_\sigma$ *is* right strict *if, for all* $M \in \Lambda^0$,

$$M \notin \mathcal{V}_\sigma \Rightarrow \forall N \in \Lambda^0.\ NM \notin \mathcal{V}_\sigma\ .$$

For many λ-pre-congruences $\leq_\sigma$, it is possible (and convenient) to extend the set $\mathcal{V}_\sigma$ on all $\Lambda(C)$, in such a way that left (right) strictness is preserved. For this purpose, we introduce the following notation:

Notation Let $\delta : \{\Lambda(C), \Lambda^0(C)\} \to \{\Lambda(C), \mathcal{V}_\sigma, \Lambda^0(C), \mathcal{V}_\sigma \cap \Lambda^0(C)\}$ be the function defined by

$$\delta(\Lambda(C)) = \begin{cases} \Lambda(C) & \text{if } \leq_\sigma \text{ is not right strict} \\ \mathcal{V}_\sigma & \text{if } \leq_\sigma \text{ is right strict.} \end{cases}$$

$$\delta(\Lambda^0(C)) = \begin{cases} \Lambda^0(C) & \text{if } \leq_\sigma \text{ is not right strict} \\ \mathcal{V}_\sigma \cap \Lambda^0(C) & \text{if } \leq_\sigma \text{ is right strict.} \end{cases}$$

Two notions of β-reduction are naturally associated to non left strict and left strict λ-pre-congruences respectively, i.e. *full* β-reduction and *value-restricted* (*closed value-restricted*) β-reduction:

Definition 7.1.7 • *Let* $\leq_\sigma$ *be a non right strict λ-pre-congruence. Then full β-reduction is* correct *w.r.t.* $\leq_\sigma$ *if, for all* $M, N \in \Lambda(C)$, $(\lambda x.M)N \leq_\sigma M[N/x]$.

- *Let* $\leq_\sigma$ *be a right strict λ-congruence. Then* (closed) value-restricted *β-reduction is* correct *w.r.t.* $\leq_\sigma$ *if, for all* $M \in \Lambda(C)$, $N \in \delta(\Lambda(C))$ *(*$N \in \delta(\Lambda^0(C))$*)*, $(\lambda x.M)N \leq_\sigma M[N/x]$.

Obviously, if a notion of β-reduction is correct w.r.t. a λ-congruence $\approx_\sigma$, then β-convertible λ-terms are $\approx_\sigma$-congruent.

Other important equivalences induced by a λ-congruence are the *applicative pre-equivalence* and the *applicative equivalence*, which, as we will see in the sequel, often coincide with the corresponding λ-congruence.

Definition 7.1.8 (σ-applicative (pre-)Equivalence) *Let* $\leq_\sigma$ *be a λ-pre-congruence.*

• *The* applicative pre-equivalence $\leq_\sigma^{app} \subseteq \Lambda(C) \times \Lambda(C)$ *is defined by: let* M, N *be such that* $FV(M,N) \subseteq \{x_1, \ldots, x_k\}$,

$M \leq_\sigma^{app} N \iff$
$\forall Q_1, \ldots, Q_k, P_1, \ldots, P_n \in \delta(\Lambda^0(C))$.
$(M[Q_1/x_1 \ldots Q_k/x_k]P_1 \ldots P_n \in \mathcal{V}_\sigma \;\Rightarrow\; N[Q_1/x_1 \ldots Q_k/x_k]P_1 \ldots P_n \in \mathcal{V}_\sigma)$.

• *The* applicative equivalence $\approx_\sigma^{app} \subseteq \Lambda(C) \times \Lambda(C)$ *is defined by:*

$$\approx_\sigma^{app} = \leq_\sigma \cap (\leq_\sigma)^{-1}\ .$$

Definition 7.1.9 *Let $\leq_\sigma^{app}$ be an applicative pre-equivalence. Then $\leq_\sigma^{app}$ is* extensional *if, forall $M, N \in \Lambda(C)$,*

$$\forall P \in \delta(\Lambda^0(C)).\ MP \leq_\sigma^{app} NP \implies M \leq_\sigma^{app} N\ .$$

7.2 Final Descriptions of λ-Theories

In this section we introduce functors which will be used to provide final semantics for various λ-theories. In order to model a λ-theory, one would like to use the functor $F(X) = X \to X$, but this functor is *contravariant* in the first occurrence of X and hence it falls outside our theory. Some alternative *covariant* ways has to be looked for. This makes final semantics for untyped λ-calculus problematic and well worthed a detailed analysis. Two covariant functors can be fruitfully used to this end, each yielding interesting coinduction principles. These are the functors F and G defined below, originally introduced in [HL95]. The intuition behind their definition is to endow the set of closed λ-terms with the coalgebra structure which view a λ-term either as a *function* over closed λ-terms, or as a *functional collection of pairs.* In both cases we need two copies of the semantical domain, the first for modeling terms in $\mathcal{V}_\sigma$, the second for modeling terms not in $\mathcal{V}_\sigma$.

Definition 7.2.1 (Functor F) *Let $F : Class^*(U) \to Class^*(U)$ be the endofunctor defined by*

$$\begin{array}{l} F(X) = \{v\} \times (\delta(\Lambda^0(C)) \to X)\ +\ \{nv\} \times (\delta(\Lambda^0(C)) \to X) \\ F(f) = id_{\{v\}} \times (id_{\delta(\Lambda^0(C))} \to f)\ +\ id_{\{nv\}} \times (id_{\delta(\Lambda^0(C))} \to f). \end{array}$$

Definition 7.2.2 (Functor G) *Let $G : Class^*(U) \to Class^*(U)$ be the endofunctor defined by*

$$\begin{array}{l} G(X) = \{v\} \times \mathcal{P}(X \times X)\ +\ \{nv\} \times \mathcal{P}(X \times X) \\ G(f) = id_{\{v\}} \times (f \times f)^+\ +\ id_{\{nv\}} \times (f \times f)^+. \end{array}$$

Definition 7.2.3 *Let $(\Lambda^0(C),\ \alpha_{\Lambda^0(C)})$ be the $F(G)$-coalgebra defined by*

$$\alpha_{\Lambda^0(C)}(M) = \begin{cases} (v, \{(N, MN) \mid N \in \delta(\Lambda^0(C))\}) & \textit{if } M \in \mathcal{V}_\sigma \\ (nv, \{(N, MN) \mid N \in \delta(\Lambda^0(C))\}) & \textit{if } M \notin \mathcal{V}_\sigma. \end{cases}$$

The coinduction principle associated to the functor F and to the coalgebra $(\Lambda^0(C),\ \alpha_{\Lambda^0(C)})$ can be expressed set-theoretically as follows:

Theorem 7.2.4 *The greatest F-bisimulation on the coalgebra $(\Lambda^0(C),\ \alpha_{\Lambda^0(C)})$, $\approx_\sigma^F$, is the greatest fixed point of the monotone operator $\Phi_\sigma^F : \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C)) \longrightarrow \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C))$ defined as follows:*

$$\Phi_\sigma^F(\mathcal{R}) = \{(M, N) \mid (M \in \mathcal{V}_\sigma \ \Leftrightarrow\ N \in \mathcal{V}_\sigma)\ \wedge\ \forall P \in \delta(\Lambda^\circ(C)).\ MP\ \ \mathcal{R}\ \ NP\}\ .$$

Φ^F_σ-bisimulations can be extended naturally to open terms using substitutions, i.e.:

Definition 7.2.5 *Let $\mathcal{R}$ be a Φ^F_σ-bisimulation. For all $M, N \in \Lambda(C)$ such that $FV(M,N) \subseteq \{x_1,\dots,x_k\}$, define*

$$M \ \mathcal{R}^{ext}\ N \iff$$
$$\forall Q_1,\dots,Q_k \in \delta(\Lambda^0(C)).\ M[Q_1/x_1 \dots Q_k/x_k]\ \mathcal{R}\ \ N[Q_1/x_1 \dots Q_k/x_k]\ .$$

In the sequel, by abuse of notation, we will simply denote $\mathcal{R}^{ext}$ by $\mathcal{R}$.

The equivalence $\approx^F_\sigma$ can be characterized "explicitly" in terms of the σ-*applicative equivalence*:

Lemma 7.2.6 $\approx^{app}_\sigma\ =\ \approx^F_\sigma$.

Proof It is immediate to show that $\approx^{app}_\sigma$ is a Φ^F_σ-bisimulation, hence $\approx^{app}_\sigma \subseteq \approx^F_\sigma$. In order to prove the converse, i.e. $\approx^{app}_\sigma \supseteq \approx^F_\sigma$, one can easily show, by induction on the length of $\overrightarrow{P}$ ($\overrightarrow{P}$ abbreviates $P_1 \dots P_n$ for $n \geq 0$), that:
$(M,N) \in \approx^F_\sigma \implies (M\overrightarrow{P}, N\overrightarrow{P}) \in \approx^F_\sigma$. $\square$

Lemma 7.2.7 *$\leq^{app}_\sigma$ is the greatest fixed point of the following monotone operator $\Phi^{\leq}_\sigma : \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C)) \longrightarrow \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C))$:*

$$\Phi^{\leq}_\sigma(\mathcal{R}) = \big\{(M,N)\ \big|\ (M \in \mathcal{V}_\sigma \Rightarrow N \in \mathcal{V}_\sigma)\ \wedge\ \forall P \in \Lambda^\circ(C).\ MP\ \ \mathcal{R}\ \ NP\ \ \big\}.$$

The coinduction principle associated to the functor G and the coalgebra $(\Lambda^0(C),\ \alpha_{\Lambda^0(C)})$ can be expressed set-theoretically as follows:

Theorem 7.2.8 *The greatest G-bisimulation on the coalgebra $(\Lambda^0(C),\ \alpha_{\Lambda^0(C)})$, $\approx^G_\sigma$, is the greatest fixed point of the monotone operator $\Phi^G_\sigma\ :\ \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C)) \longrightarrow \mathcal{P}(\Lambda^\circ(C) \times \Lambda^\circ(C))$ defined as follows:*

$$\Phi^G_\sigma(\mathcal{R}) = \{(M,N) \mid (M \in \mathcal{V}_\sigma\ \Leftrightarrow\ N \in \mathcal{V}_\sigma)\ \wedge$$
$$\left(\begin{array}{l} \forall P \in \delta(\Lambda^\circ(C)) \exists Q \in \delta(\Lambda^\circ(C)).\ P\ \ \mathcal{R}\ \ Q\ \ \wedge\ \ MP\ \ \mathcal{R}\ \ NQ \\ \wedge \\ \forall P \in \delta(\Lambda^\circ(C)) \exists Q \in \delta(\Lambda^\circ(C)).\ P\ \ \mathcal{R}\ \ Q\ \ \wedge\ \ NP\ \ \mathcal{R}\ \ MQ \end{array}\right)\}\ .$$

Φ^G_σ-bisimulations are extended to open terms as follows. Notice the difference between the following Definition 7.2.9 and Definition 7.2.5 above. The reason for this difference will be clear in Section 7.7.

Definition 7.2.9 *Let $\mathcal{R}$ be a Φ^G_σ-bisimulation. For all $M, N \in \Lambda$ such that $FV(M,N) \subseteq \{x_1,\dots,x_k\}$,*

$$M\ \mathcal{R}^{ext}\ N \iff$$

$$\forall P_1,\dots,P_k \in \delta(\Lambda^0).\ \exists Q_1,\dots,Q_k \in \delta(\Lambda^0).$$
$$(\forall i = 1,\dots,k.\ P_i\ \mathcal{R}\ Q_i\ \wedge\ M[P_1/x_1 \dots P_k/x_k]\ \mathcal{R}\ \ N[Q_1/x_1 \dots Q_k/x_k])\ \wedge$$

$$\forall Q_1,\dots,Q_k \in \delta(\Lambda^0).\ \exists P_1,\dots,P_k \in \delta(\Lambda^0).$$
$$(\forall i = 1,\dots,k.\ P_i\ \mathcal{R}\ Q_i\ \wedge\ M[P_1/x_1 \dots P_k/x_k]\ \mathcal{R}\ \ N[Q_1/x_1 \dots Q_k/x_k])\ .$$

In the sequel, by abuse of notation, we will simply denote $\mathcal{R}^{ext}$ by $\mathcal{R}$.

It is not immediate how to give a natural explicit characterization of the equivalence $\approx_\sigma^G$, like the one for $\approx_\sigma^F$. We leave this as an open problem.

Using the notation introduced in Chapter 3, we have:

Lemma 7.2.10 • *The final coalgebra of the functor F is (U^F, α_{U^F}).*

- *The final coalgebra of the functor G is (U^G, α_{U^G}).*

Proposition 7.2.11 • – *The final semantics $\mathcal{M}_\lambda^F : (\Lambda^0(C), \alpha_{\Lambda^0(C)}) \to (U^F, \alpha_{U^F})$ is defined by*

$$\mathcal{M}_\lambda^F(M) = \begin{cases} (v, \{(N,\ \mathcal{M}_\lambda^F(MN)) \mid N \in \delta(\Lambda^0(C))\}) & \text{if } M \in \mathcal{V}_\sigma \\ (nv, \{(N,\ \mathcal{M}_\lambda^F(MN)) \mid N \in \delta(\Lambda^0(C))\}) & \text{if } M \notin \mathcal{V}_\sigma . \end{cases}$$

– *For all $M, N \in \Lambda^0(C)$,*

$$\mathcal{M}_\lambda^F(M) = \mathcal{M}_\lambda^F(N) \iff M \approx_\sigma^F N \ .$$

• – *The final semantics $\mathcal{M}_\lambda^G : (\Lambda^0(C), \alpha_{\Lambda^0(C)}) \to (U^G, \alpha_{U^G})$ is defined by*

$$\mathcal{M}_\lambda^G(M) = \begin{cases} (v, \{(\mathcal{M}_\lambda^G(N),\ \mathcal{M}_\lambda^G(MN)) \mid & \\ \qquad\qquad N \in \delta(\Lambda^0(C))\}) & \text{if } M \in \mathcal{V}_\sigma \\ (nv, \{(\mathcal{M}_\lambda^G(N),\ \mathcal{M}_\lambda^G(MN)) \mid & \\ \qquad\qquad N \in \delta(\Lambda^0(C))\}) & \text{if } M \notin \mathcal{V}_\sigma . \end{cases}$$

– *For all $M, N \in \Lambda^0(C)$,*

$$\mathcal{M}_\lambda^G(M) = \mathcal{M}_\lambda^G(N) \iff M \approx_\sigma^G N \ .$$

If $\approx_\sigma = \approx_\sigma^F$ ($\approx_\sigma^G$), then we can reason by coinduction, using the coinduction principle of Theorem 7.2.4 (7.2.8), directly on the equivalence $\approx_\sigma$. In the next two sections we will present a list of λ-theories for which $\approx_\sigma = \approx_\sigma^F$, and we will discuss various techniques for showing this equality. The identity $\approx_\sigma = \approx_\sigma^F$ is often called "Context Lemma", following Milner, who has discussed it in the case of typed λ-calculus. The identity $\approx_\sigma = \approx_\sigma^G$ is quite problematic. We shall not discuss it in detail in this thesis. For examples of λ-congruences which do not satisfy the identities above see [Len97a].

7.3 Examples of λ-Congruences and λ-Theories

In this section we present a list of λ-congruences arising from various important reduction strategies, which have been extensively discussed in the literature. We will see in the following sections that each such congruence can be characterized coinductively both as $\approx_\sigma^F$-equivalence (and as $\approx_\sigma^G$-equivalence).

A *reduction strategy* is a procedure for determining, for each λ-term, a specific β-redex in it, to contract. A (possibly non-deterministic) strategy can be formalized as a relation $\rightarrow_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ $(\Lambda^0(C) \times \Lambda^0(C))$ such that, if $(M,N) \in \rightarrow_\sigma$ (also written infix as $M \rightarrow_\sigma N$), then N is a possible result of applying $\rightarrow_\sigma$ to M. The set of terms which do not belong to the domain of $\rightarrow_\sigma$ are partitioned into two disjoint sets: the set of *σ-values*, denoted by Val_σ, and the set of *σ-deadlocks*. Given $\rightarrow_\sigma$, we can define the *evaluation relation* $\Downarrow_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ $(\Lambda^0(C) \times \Lambda^0(C))$, such that $M \Downarrow_\sigma N$ holds if and only if there exists a (possible empty) reduction path from M to a σ-value N. If there exists N such that $M \Downarrow_\sigma N$, then $\rightarrow_\sigma$ *halts successfully* on M and M converges ($M \Downarrow_\sigma$), otherwise $\rightarrow_\sigma$ does not terminate on M or reaches a *deadlock* from M, and M diverges ($M \not\Downarrow_\sigma$). Each reduction strategy induces an *operational semantics*, in that we can imagine a machine which evaluates terms by implementing the given strategy. The *observational equivalence* arises when we consider programs as *black boxes* and only observe their "halting properties".

Definition 7.3.1 (σ-observational Equivalence) *Let $\rightarrow_\sigma$ be a reduction strategy and let $M, N \in \Lambda(C)$. The* observational equivalence *$\approx_\sigma$ is defined by*

$$M \approx_\sigma N \iff \forall C[\ ].(C[M], C[N] \in \Lambda^0(C) \Rightarrow (C[M] \Downarrow_\sigma \Leftrightarrow C[N] \Downarrow_\sigma))\ .$$

It is immediate to check that a σ-observational equivalence is a λ-congruence, and it is induced by the set of $\Downarrow_\sigma$-convergent (σ-valuable) terms.

The list of strategies that we shall consider in this thesis is the following:

$\rightarrow_l$ strategy. The *lazy call-by-name* strategy $\rightarrow_l \subseteq \Lambda^0 \times \Lambda^0$ reduces the *leftmost* β-redex not appearing in a λ-abstraction. $Val_l = \{\lambda x.M \mid M \in \Lambda\} \cap \Lambda^0$. The evaluation $\Downarrow_l$ is the least binary relation over $\Lambda^0 \times Val_l$ satisfying the rules:

$$\frac{}{\lambda x.M \Downarrow_l \lambda x.M} \qquad\qquad \frac{M \Downarrow_l \lambda x.P \quad P[N/x] \Downarrow_l Q}{MN \Downarrow_l Q}$$

Classical β-reduction is correct w.r.t. $\approx_l$ (see [AO93]).
This is the reduction strategy of lazy functional languages such as Miranda.

$\rightarrow_v$ strategy. Plotkin's *lazy call-by-value* strategy $\rightarrow_v \subseteq \Lambda^0 \times \Lambda^0$ reduces the *leftmost* β-redex, not appearing within a λ-abstraction, whose argument is a λ-abstraction. $Val_v = \{\lambda x.M \mid M \in \Lambda\} \cap \Lambda^0$. The evaluation $\Downarrow_v$ is the least binary relation over $\Lambda^0 \times Val_v$ satisfying the following rules:

$$\frac{}{\lambda x.M \Downarrow_v \lambda x.M} \qquad\qquad \frac{M \Downarrow_v \lambda x.P \quad N \Downarrow_v Q \quad P[Q/x] \Downarrow_v U}{MN \Downarrow_v U}$$

A notion of β-reduction which is correct w.r.t. $\approx_v$ is Plotkin's $\rightarrow_{\beta_v} \subseteq \Lambda \times \Lambda$, i.e.:
$(\lambda x.M)N \rightarrow_{\beta_v} M[N/x]$, if N is a variable or an abstraction.
Notice that the β_v-reduction is far from being the largest notion of β-reduction correct w.r.t. $\approx_v$. E.g., we can extend this notion by allowing the reduction whenever N reduces to a variable or an abstraction. This is exactly the value

restricted β-reduction of Definition 7.1.7, when we extend the notion of values and convergence to all Λ.
This is the reduction strategy implemented in the $SECD$ machine of Landin and used in ML.

$\to_o$ **strategy.** Let Ω be a new constant. The non-deterministic strategy $\to_o \subseteq \Lambda^0(\{\Omega\}) \times \Lambda^0(\{\Omega\})$ ([HR92]) rewrites λ-terms which contain occurrences of the constant Ω by reducing any β-redex. $Val_o = \Lambda^0$. Normal forms which are not in Val_o are the $\to_o$-deadlock terms. The evaluation relation $\Downarrow_o$ is the least binary relation over $\Lambda^0(\{\Omega\}) \times Val_o$ satisfying the following rules:

$$\frac{M \in Val_o}{M \Downarrow_o M} \qquad \frac{C[(\lambda x.M)N] \notin Val_o \quad C[M[N/x]] \Downarrow_o P}{C[(\lambda x.M)N] \Downarrow_o P}$$

β-reduction is trivially correct w.r.t. $\approx_o$.

$\to_h$ **strategy.** The *head call-by-name* strategy $\to_h \subseteq \Lambda \times \Lambda$ reduces the *leftmost* β-redex, if the term is not in head normal form. Val_h is the set of λ-terms in head normal form. The evaluation $\Downarrow_h$ is the least binary relation over $\Lambda \times Val_h$ satisfying the following rules, for $n \geq 0$:

$$\frac{}{xM_1 \ldots M_n \Downarrow_h xM_1 \ldots M_n} \qquad \frac{M \Downarrow_h N}{\lambda x.M \Downarrow_h \lambda x.N} \qquad \frac{M[N/x]M_1 \ldots M_n \Downarrow_h P}{(\lambda x.M)NM_1 \ldots M_n \Downarrow_h P}$$

β-reduction is correct w.r.t. $\approx_h$ (see e.g. [Bar84]).

$\to_n$ **strategy.** The *normalizing* strategy $\to_n \subseteq \Lambda \times \Lambda$ reduces the leftmost β-redex. Val_n is the set of λ-terms in normal form. The evaluation $\Downarrow_n$ is the least binary relation over $\Lambda \times Val_n$ satisfying the following rules, for $n \geq 0$:

$$\frac{M_1 \Downarrow_n M_1' \ \ldots \ M_n \Downarrow_n M_n'}{xM_1 \ldots M_n \Downarrow_n xM_1' \ldots M_n'} \qquad \frac{M \Downarrow_n N}{\lambda x.M \Downarrow_n \lambda x.N} \qquad \frac{M[N/x]M_1 \ldots M_n \Downarrow_n P}{(\lambda x.M)NM_1 \ldots M_n \Downarrow_n P}$$

β-reduction is correct w.r.t. $\approx_n$.

Another important λ-congruence arises when we take as set of values the set SN of strongly β-normalizing terms:

Definition 7.3.2 *Let $\approx_{SN} \subseteq \Lambda(C) \times \Lambda(C)$ be the equivalence defined by*

$M \approx_{SN} N \iff$
$\forall C[\ ].\ (C[M], C[N] \in \Lambda^\circ(C) \Longrightarrow (C[M] \in SN \iff C[N] \in SN))$.

This λ-congruence $\approx_{SN}$ can be also viewed as the observational equivalence induced by any *perpetual strategy*. The following strategy, $\to_p$, is an example of a perpetual strategy.

$\to_p$ **strategy.** Barendregt's *perpetual* strategy $\to_p \subseteq \Lambda \times \Lambda$ reduces the leftmost β-redex not in the operator of a redex, which is either an $I\beta$-redex, or a $K\beta$-redex whose argument is a normal form. Val_p is the set of λ-terms in normal form. One can easily show that the evaluation $\Downarrow_p$ is the least binary relation over $\Lambda \times Val_p$ satisfying the following rules, for $n \geq 0$:

$$\frac{M_1 \Downarrow_p M_1' \;\ldots\; M_n \Downarrow_p M_n'}{xM_1 \ldots M_n \Downarrow_p xM_1' \ldots M_n'} \qquad \frac{M \Downarrow_p N}{\lambda x.M \Downarrow_p \lambda x.N}$$

$$\frac{N \Downarrow_p \quad M[N/x]M_1 \ldots M_n \Downarrow_p V}{(\lambda x.M)NM_1 \ldots M_n \Downarrow_p V}$$

The closed value-restricted reduction $\to_{\beta_{N^0}}$, defined by

$(\lambda x.M)N \to_{\beta_{N^0}} M[N/x]$, if $N \in \Lambda^0$ and $N \Downarrow_p$,

is correct w.r.t. $\approx_p$ (see [HL9?]).
Notice that the value-restricted notion of β-reduction is not correct w.r.t. $\approx_p$, since $\lambda x.x. \not\approx_p (\lambda xy.y)(zz)$; e.g. take $C[\;] = (\lambda z.[\;])(\lambda x.xx)$. However, other notions of β-reductions, intermediate between the closed value-restricted and the value-restricted β-reduction, are correct (see [HL9?]), e.g. the reduction $\to_{\beta_{KN}}$ defined by

$(\lambda x.M)N \to_{\beta_{KN}} M[N/x]$, if $(\lambda x.M)N$ is either an $I\beta$-redex or a $K\beta$-redex with $N \in \Lambda^0$ and $N \Downarrow_p$,

is correct w.r.t. $\approx_p$.

In what follows, we will often denote by $\approx_p$ the λ-congruence $\approx_{SN}$ of strongly normalizing terms, but we stress that this λ-congruence does not depend on the particular perpetual strategy $\to_p$.

7.4 Computationally Adequate Models

In this section we phrase in terms of intersection types ([CDHL82, BCD83, Abr91]) the computational adequacy of an algebraic lattice λ-model D^σ. In particular, we present computationally adequate models for all the observational equivalences induced by the strategies of Section 7.3. We assume that the finitary logical presentation of D^σ is given by the intersection types theory $\mathcal{T}_\sigma = (T_\sigma, \leq_\sigma)$.

Definition 7.4.1 *Let $\mathcal{T}_\sigma = (T_\sigma, \leq_\sigma)$ be the intersection type theory corresponding to D^σ. Then*

$$(T_\sigma \ni)\ \phi ::= \alpha \mid \phi \wedge \phi \mid \phi \to \phi\ ,$$

where $\alpha \in TB_\sigma$, and TB_σ is a non-empty set of base types.

The relation $\leq_\sigma$ on types is the least relation containing the set IB_σ of inequalities involving the base types in TB_σ, and closed under the rules:

$\phi \leq_\sigma \phi \qquad \phi_1 \wedge \phi_2 \leq_\sigma \phi_1 \qquad \phi_1 \wedge \phi_2 \leq_\sigma \phi_2$

$$\frac{\phi_1 \leq_\sigma \phi_2 \quad \phi_2 \leq_\sigma \phi_3}{\phi_1 \leq_\sigma \phi_3} \qquad \frac{\phi \leq_\sigma \phi_1 \quad \phi \leq_\sigma \phi_2}{\phi \leq_\sigma \phi_1 \wedge \phi_2} \qquad \frac{\phi_2 \leq_\sigma \phi_1 \quad \psi_1 \leq_\sigma \psi_2}{\phi_1 \rightarrow \psi_1 \leq_\sigma \phi_2 \rightarrow \psi_2}$$

$$\phi \rightarrow \phi_1 \wedge \phi_2 \leq_\sigma (\phi \rightarrow \phi_1) \wedge (\phi \rightarrow \phi_2) \qquad (\phi \rightarrow \phi_1) \wedge (\phi \rightarrow \phi_2) \leq_\sigma \phi \rightarrow \phi_1 \wedge \phi_2$$

We put

$$\phi_1 =_\sigma \phi_2 \textit{ if and only if } \phi_1 \leq_\sigma \phi_2 \ \wedge \ \phi_2 \leq_\sigma \phi_1 \ .$$

A type environment *is a partial function* $\Gamma : Vars \longrightarrow T_\sigma$. *Let* Env_Γ *denote the set of type environments. The type assignment system* S_σ *includes the following rules and, possibly, a nonempty set* SB_σ *of axioms involving the base types, and the base constants of the language:*

$$\Gamma[\phi/x] \vdash_\sigma x : \phi \qquad (var)$$

$$\frac{\Gamma \vdash_\sigma M : \phi \quad \Gamma \vdash_\sigma M : \psi}{\Gamma \vdash_\sigma M : \phi \wedge \psi} \ (\wedge I) \qquad \frac{\Gamma \leq_\sigma \Delta \quad \Delta \vdash_\sigma M : \phi \quad \phi \leq_\sigma \psi}{\Gamma \vdash_\sigma M : \psi} \ (\leq)$$

$$\frac{\Gamma[\phi/x] \vdash_\sigma M : \psi}{\Gamma \vdash_\sigma \lambda x.M : \phi \rightarrow \psi} \ (\rightarrow I) \qquad \frac{\Gamma \vdash_\sigma M : \phi \rightarrow \psi \quad \Gamma \vdash_\sigma N : \phi}{\Gamma \vdash_\sigma MN : \psi} \ (\rightarrow E)$$

The canonical interpretation $[\![\ ]\!]^{D^\sigma}$ *in the filter model* D^σ *can be characterized as follows:*

$$[\![M]\!]^{D^\sigma}_\rho = \{\phi \mid \exists \Gamma \sqsubseteq_{D^\sigma} \rho.\ \Gamma \vdash_\sigma M : \phi\} \ ,$$

where $\rho : Var \rightarrow VD^\sigma$, *for a suitable* $VD^\sigma \subseteq D^\sigma$. $\Gamma \sqsubseteq_{D^\sigma} \rho$ *means that* $\forall x \in Var.\ \Gamma(x) \sqsubseteq_{D^\sigma} \rho(x)$, *and* $\sqsubseteq_{D^\sigma}$ *denotes the order relation on* D^σ.

The computational adequacy of the model D^σ can now take the strong form:

Definition 7.4.2 (Computational Adequacy) *An algebraic lattice model* D^σ *is* computationally adequate *if there exists a filter of types* $T^{conv}_\sigma \subseteq T_\sigma$, *and a subset* Env^{conv}_Γ *of* $\mathcal{T}^{conv}_\sigma$*-type environments such that*

$$M \Downarrow_\sigma \iff \exists \Gamma \in Env^{conv}_\Gamma.\ \exists \phi \in T^{conv}_\sigma.\ (\Gamma \vdash_\sigma M : \phi) \ .$$

7.4.1 A Computationally Adequate Model for $\approx_l$

A computationally adequate model for $\approx_l$ is the model D^l, studied in [AO93]. The model D^l is the inverse limit initial solution of the equation $D \simeq [D \rightarrow D]_\perp$ in the category $CPO_\perp$. The intersection type presentation of the model D^l is given by:

Definition 7.4.3 *Let* $\mathcal{T}_l = (T_l, \leq_l)$ *be the intersection type theory:*

$$(T_l \ni)\ \phi ::= \ \omega \ \mid \ \phi \wedge \phi \ \mid \ \phi \rightarrow \phi \ .$$

The axiomatization of $\leq_l$ consists of the standard rules of Definition 7.4.1, and of the following rule involving the base type ω:

$$\phi \leq_l \omega \ .$$

The type system S_l consists of the rules of Definition 7.4.1, and moreover:

$$\Gamma \vdash_l M : \omega \qquad (\omega) \ .$$

Theorem 7.4.4 (Computational Adequacy of D^l)

$$M \Downarrow_l \iff \exists \phi \neq \omega. \ \vdash_l M : \phi \ .$$

7.4.2 A Computationally Adequate Model for $\approx_v$

A computationally adequate model for $\approx_v$ is the model D^v, studied in [EHR92]. The model D^v is the inverse limit initial solution of the equation $D \simeq [D \rightarrow_\perp D]_\perp$ in the category $CPO_\perp$, where $[D \rightarrow_\perp D]$ denotes the cpo of strict continuous functions. The intersection type presentation of the model D^v is given by:

Definition 7.4.5 *Let $\mathcal{T}_v = (T_v, \leq_v)$ be the intersection type theory:*

$$(T_v \ni) \ \phi ::= \ \nu \ | \ \phi \wedge \phi \ | \ \phi \rightarrow \phi \ .$$

The axiomatization of $\leq_v$ consists of the standard rules of Definition 7.4.1, and of the following rule involving the base type ν:

$$\phi \leq_v \nu \ .$$

The type system S_v consists of the rules of Definition 7.4.1, and moreover:

$$\Gamma \vdash_v \lambda x.M : \nu \qquad (\nu) \ .$$

Theorem 7.4.6 (Computational Adequacy of D^v)

$$M \Downarrow_v \iff \exists \phi \in \ T_v. \ \vdash_v M : \phi \ .$$

7.4.3 A Computationally Adequate Model for $\approx_o$

A computationally adequate model for $\approx_o$ is the model D^o, studied in [HR92]. The model D^o is the inverse limit solution of the equation $D \simeq [D \rightarrow D]$ in the category CPO, with domain $D_0^o = \{\perp, \nu\}$ and projection $j_{1,0}^o : D_1 \rightarrow D_0$ defined by: $j_{1,0}^o(\perp) = \perp$, $j_{1,0}^o(d) = \nu$, for $d \neq_{D^o} \perp$. The intersection type presentation of the model D^o is given by:

Definition 7.4.7 *Let $\mathcal{T}_o = (T_o, \leq_o)$ be the intersection type theory:*

$$(T_o \ni) \ \phi ::= \ \omega \ | \ \nu \ | \ \phi \wedge \phi \ | \ \phi \rightarrow \phi \ .$$

The axiomatization of $\leq_o$ consists of the standard rules of Definition 7.4.1, and moreover of the following rules involving the base types ω and ν:

$$\phi \leq_o \omega \quad \omega \leq_o \omega \to \omega \quad \nu =_o \nu \to \nu \ .$$

The type system S_o consists of the rules of Definition 7.4.1 and moreover:

$$\Gamma \vdash_o M : \omega \quad (\omega) \ .$$

Theorem 7.4.8 (Computational Adequacy of D^o)

$$M \Downarrow_o \iff \vdash_o M : \nu \ .$$

7.4.4 A Computationally Adequate Model for $\approx_h$

A computationally adequate model for $\approx_h$ is the standard D_∞ model, which, by uniformity, we will call D^h. The model D^h is the inverse limit solution of the equation $D \simeq [D \to D]$ in the category CPO, with domain $D_0^h = \{\bot, 1\}$ and projection $j_{1,\bot}^h : D_1^h \to D_0^h$ defined by: $j_{1,\bot}^h(d) = \bot$, if $d \neq \lambda d \in D_0^h.1$, $j_{1,\bot}(\lambda d \in D_0^h.1) = 1$. The intersection type presentation of the model D^h is given by:

Definition 7.4.9 *Let $\mathcal{T}_h = (T_h, \leq_h)$ be the intersection type theory:*

$$(T_h \ni) \ \phi ::= \ \omega \ | \ 1 \ | \ \phi \wedge \phi \ | \ \phi \to \phi \ .$$

The axiomatization of $\leq_h$ consists of the standard rules of Definition 7.4.1, and moreover of the following rules involving the base types ω and 1:

$$\phi \leq_h \omega \quad \omega \leq_h \omega \to \omega \quad 1 =_h \omega \to 1 \ .$$

The type system S_h consists of the rules of Definition 7.4.1 and moreover:

$$\Gamma \vdash_h M : \omega \quad (\omega) \ .$$

Theorem 7.4.10 (Computational Adequacy of D^h)

$$M \Downarrow_h \iff \exists \Gamma \ \exists \phi \neq_h \omega . \ \Gamma \vdash_h M : \phi \ .$$

7.4.5 A Computationally Adequate Model for $\approx_n$

A computationally adequate model for $\approx_n$ is the model D^n, studied in [CDZ87]. The model D^n is the inverse limit solution of the equation $D \simeq [D \to D]$ in the category CPO, with domain $D_0^n = \{\bot, 0, 1\}$, with $0 \sqsubseteq_{D^n} 1$, and projection $j_{1,0}^n : D_1^n \to D_0^n$ defined by: $j_{1,0}^n(\bot) = \bot$, $j_{1,0}^n(d) = 0$, if $d \neq \bot, \lambda d \in D_0^n.1$, $j_{1,0}(\lambda d \in D_0^n.1) = 1$. The intersection type presentation of the model D^n is given by:

Definition 7.4.11 *Let $\mathcal{T}_n = (T_n, \leq_n)$ be the intersection type theory:*

$$(T_n \ni)\ \phi ::= \ \omega \ \mid \ 0 \ \mid \ 1 \ \mid \ \phi \wedge \phi \ \mid \ \phi \to \phi \ .$$

The axiomatization of $\leq_n$ consists of the standard rules of Definition 7.4.1, and moreover of the following rules involving the base types ω and 1:

$$\phi \leq_n \omega \qquad \omega \leq_n \omega \to \omega \qquad 0 =_n 1 \to 0 \qquad 1 =_n 0 \to 1 \ .$$

The type system S_n consists of the rules of Definition 7.4.1 and moreover:

$$\Gamma \vdash_n M : \omega \qquad (\omega) \ .$$

Theorem 7.4.12 (Computational Adequacy of D^n)

$$M \Downarrow_n \iff \Gamma_1 \vdash_n M : 0 \ ,$$

where $\Gamma_1 = \{x : 1 \mid x \in Var\}$.

7.4.6 A Computationally Adequate Model for $\approx_p$

A computationally adequate model for $\approx_p$ is the model D^p, studied in [HL9?]. The model D^p is the inverse limit solution of the equation $D \simeq [D \to_\perp D]$ in the category $CPO_\perp$, with domain $D_0^p = \{\perp, 0, 1\}$, with $0 \sqsubseteq_{D^p} 1$, and projection $j_{1,0}^p : D_1^p \to D_0^p$ defined by: $j_{1,0}^p(\perp) = \perp$, $j_{1,0}^p(d) = 0$, if $d \neq_{D^p} \perp, \lambda d \in D_0^p.$ if $d =_{D^p} \perp$ then $\perp$ else 1, $j_{1,0}(\lambda d \in D_0^p.$ if $d =_{D^p} \perp$ then $\perp$ else $1) = 1$. The intersection type presentation of the model D^p is given by:

Definition 7.4.13 *Let $\mathcal{T}_p = (T_p, \leq_p)$ be the intersection type theory:*

$$(T_p \ni)\ \phi ::= \ 0 \ \mid \ 1 \ \mid \ \phi \wedge \phi \ \mid \ \phi \to \phi \ .$$

The axiomatization of $\leq_p$ consists of the standard rules of Definition 7.4.1, and moreover of the following rules involving the base types 0 and 1:

$$\phi \leq_p 0 \qquad 1 \leq_p \phi \qquad 0 =_p 1 \to 0 \qquad 1 =_p 0 \to 1 \ .$$

The type assignment system S_p is such that $SB_p = \emptyset$.

Theorem 7.4.14 (Computational Adequacy of D^p)

$$M \Downarrow_p \iff \exists \Gamma.\ \Gamma \vdash_p M : 0 \ .$$

7.5 Φ_σ^F-coinductive Characterizations

In order to give a final semantics to a λ-congruence $\approx_\sigma$ using the functor F of Section 7.2, i.e. to provide a Φ_σ^F-coinductive characterization for $\approx_\sigma$, we neeed to establish that $\approx_\sigma^{app} = \approx_\sigma$. The inclusion $\approx_\sigma \subseteq \approx_\sigma^{app}$ is immediate; the other inclusion, however, is rather difficult to show and requires very subtle techniques. In particular we need to show that (see Theorem 7.5.1 below):

- $\approx_\sigma^{app}$ is a congruence w.r.t. application, i.e. for all $M, N, P, Q \in \Lambda(C)$, $M \approx_\sigma^{app} N \ \wedge\ P \approx_\sigma^{app} Q \implies MP \approx_\sigma^{app} NQ$;
- $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction, i.e., for all $M, N \in \Lambda(C)$ $M \approx_\sigma^{app} N \implies \lambda x.M \approx_\sigma^{app} \lambda x.N$.

It is immediate to see that when discussing congruence w.r.t. application we can restrict ourselves to the case that M, N, P, Q are all closed.

Sufficient conditions which guarantee that $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction are discussed in Section 7.6.1 below. The congruence w.r.t. application instead is the real challenge. There are various techniques for showing this fact, both syntactical and semantical. We mention the following:

1. *Coinductive argument* on *congruence candidate relations.* This method was introduced by D.Howe for the lazy call-by-name strategy $\rightarrow_l$, and later generalized to a class of *lazy* strategies by-name and by-value, including $\rightarrow_v$ ([How96]). In [Len97], we generalize and strengthen Howe's method so as to deal also with non-lazy strategies. In particular, the generalization of [Len97] applies to all the strategies of Section 7.3.
2. Method based on a *Separability algorithm.* This method is based on the existence of an effective procedure (see e.g. [Bar84]) which, given two non $\approx_\sigma$-equivalent terms, M, N, allows to define an applicative context $C[\ ]$ such that either $C[M] \Downarrow_\sigma$ and $C[M] \not\Downarrow_\sigma$, or vice versa. To our knowledge, this method works only for $\approx_h$, $\approx_n$.
3. *Logical Relations* method based on a *mixed induction-coinduction principle.* This semantical method is presented in [Len97]. It is the generalization of the technique originally introduced by Pitts ([Pit96]) for observational equivalences which have a suitable computationally adequate initial model, like $\approx_l$ and $\approx_v$. In [Len97], this method is applied to all the strategies of Section 7.3. The method in [EHR92] for $\approx_v$ can be viewed as a weaker variant.
4. Method based on the *Domain Logic* corresponding to the *intersection types* presentation of a suitable computationally adequate *CPO-λ*-model. This semantical method, introduced in [Len97], is the generalization of the technique originally introduced by Abramsky and Ong in [AO93] for the special case of $\approx_l$. In [Len97], this method is applied to all the strategies of Section 7.3.

We discuss in detail methods 1, 3, and 4.

Theorem 7.5.1 *Suppose that $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction and application. Then $\approx_\sigma^{app} \subseteq \approx_\sigma$.*

Proof We prove by induction on the context $C[\ ]$ that, for all $M, N \in \Lambda(C)$,

$$M \approx_\sigma^{app} N \implies \forall C[\ ].(C[M], C[N] \in \Lambda(C) \ \Rightarrow\ C[M] \approx_\sigma^{app} C[N]) \ .$$

□

7.6 Compositionality of the Final Semantics

An immediate corollary of Theorem 7.5.1 above is the *compositionality* of the final semantics $\mathcal{M}_\lambda^F$. In particular, in order to express compositionality w.r.t. λ-abstraction, we extend the definition of $\mathcal{M}_\lambda^F$ on all $\Lambda(C)$, by introducing syntactical environments $\rho : Var \to \Lambda^0(C)$, and defining, for all M such that $FV(M) \subseteq \{x_1, \ldots, x_n\}$,

$$(\mathcal{M}_\lambda^F)_\rho(M) = (\mathcal{M}_\lambda^F)_\rho(M[\rho(x_1)/x_1, \ldots, \rho(x_n)/x_n]) \ .$$

Corollary 7.6.1 *The final semantics $\mathcal{M}_\lambda^F$ is compositional.*

7.6.1 $\approx_\sigma^{app}$ is a Congruence w.r.t. λ-Abstraction

In this section, we discuss conditions on $\approx_\sigma^{app}$ under which $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction. We isolate two relevant sets of sufficient conditions:

Theorem 7.6.2 *If*

- *β-reduction (closed value-restricted β-reduction) is correct w.r.t. $\approx_\sigma$, and*
- *$\approx_\sigma^{app}$ is extensional,*

then $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction.

Theorem 7.6.3 *If*

- *β-reduction (closed value-restricted β-reduction) is correct w.r.t. $\approx_\sigma$, and*
- *for all $M \in \Lambda(C)$,*

$$\exists P \in \delta(\Lambda^0(C)).\ M[P/x] \Downarrow \Longleftrightarrow\ M \Downarrow ,$$

then $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction.

Using Theorems 7.6.2 and 7.6.3, one can show that all the applicative equivalences induced by the strategies of Section 7.3 are congruences w.r.t. λ-abstraction. In particular, the only non trivial cases are those of $\sigma = p$ and $\sigma = n$. For $\sigma = p$ we use Theorem 7.6.3; for showing that the implication $(\Leftarrow)$ in the second hypothesis holds, we choose as P a suitable permutator, i.e. a term of the shape $\lambda z_1 \ldots z_k y.y z_1 \ldots z_k$ (for more details see [HL9?]). For $\sigma = n$ we use Theorem 7.6.2, and for showing that the second hypothesis holds, we need to exploit extensively the separability technique of [Hyl75]. We omit this proof.

Hence we get:

Theorem 7.6.4 *For all $\sigma = l, v, o, h, n, p$, $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction.*

For those λ-congruences which arise from a notion of convergence defined on all $\Lambda(C)$ we can define the notions of *observational equivalence extended to open contexts* and of *applicative equivalence extended to open contexts*, as follows:

Definition 7.6.5 *Let $\Downarrow_\sigma \subseteq \Lambda(C) \times \Lambda(C)$ be a notion of convergence.*

- *The* observational equivalence extended to open contexts $\approx_\sigma^{ext} \subseteq \Lambda(C) \times \Lambda(C)$ *is defined by:*
  $M \approx_\sigma N \iff (C[M] \Downarrow_\sigma \Leftrightarrow C[N] \Downarrow_\sigma)$.

- *The* applicative equivalence extended to open contexts $\approx_\sigma^{app,ext} \subseteq \Lambda(C) \times \Lambda(C)$ *is defined by: let M, N be such that $FV(M,N) \subseteq \{x_1, \ldots, x_k\}$,*
  $M \approx_\sigma^{app} N \iff$
  $\forall Q_1, \ldots, Q_k \in \delta(\Lambda^0(C)),\ \forall P_1, \ldots, P_n \in \delta(\Lambda(C))$.
  $(M[Q_1/x_1 \ldots Q_k/x_k]P_1 \ldots P_n \Downarrow_\sigma\ \Leftrightarrow\ N[Q_1/x_1 \ldots Q_k/x_k]P_1 \ldots P_n \Downarrow_\sigma)$.

Applicative equivalence and extended applicative equivalence coincide for $\sigma = h, n, p$. This fact is a corollary of Theorem 7.6.4:

Corollary 7.6.6 *For $\sigma = h, n, p$, $\approx_\sigma^{app} = \approx_\sigma^{app,ext}$.*

The following proposition is an immediate consequence of the definition of $\Downarrow_\sigma$:

Proposition 7.6.7 *For $\sigma = h, n, p$, $\approx_\sigma = \approx_\sigma^{ext}$.*

7.6.2 The Congruence Candidate Method

The congruence candidate method is a syntactical method which nonetheless is quite uniform and modular. It makes essential use of the coinduction principle of Theorem 7.2.4, and it is based on the definition of a *candidate relation*, which is a congruence w.r.t. application, and which extends $\approx_\sigma^{app}$. The aim is to show that the candidate relation is a Φ_σ^F-bisimulation; hence the coinduction principle of Theorem 7.2.4 guarantees that $\approx_\sigma^{app}$ itself is a congruence w.r.t. application. We outline the:

General pattern of the congruence candidate method:

- Build a *candidate relation* $\widehat{\approx}_\sigma^{app} \subseteq \Lambda(C) \times \Lambda(C)$ such that

  1. $\widehat{\approx}_\sigma^{app} \supseteq \approx_\sigma^{app}$;
  2. $\widehat{\approx}_\sigma^{app}$ is a congruence w.r.t. application;
  3. $(\widehat{\approx}_\sigma^{app})_{|\Lambda^0(C) \times \Lambda^0(C)}$ is a Φ_σ^F-bisimulation.

- Use the coinduction principle of Theorem 7.2.4 to deduce that $\approx_\sigma^{app}$ is a congruence w.r.t. application.

More in detail, the congruence candidate method proceeds as follows. First of all, we have to explain how to build the *candidate relation* $\approx_{\sigma}^{app}$. Candidate relations are defined in terms of the extensions to open terms of Φ_{σ}^{F}-bisimulations (see Definition 7.2.5):

Definition 7.6.8 (Candidate Relation) *Let $\eta \subseteq \Lambda(C) \times \Lambda(C)$ be a reflexive and transitive Φ_{σ}^{F}-bisimulation. Define the* candidate relation $\widehat{\eta} \subseteq \Lambda(C) \times \Lambda(C)$ *by induction on M as follows:*

$$\frac{x\ \eta\ N}{x\ \widehat{\eta}\ N} \qquad \frac{M_1\ \widehat{\eta}\ M_1' \quad M_2\ \widehat{\eta}\ M_2' \quad M_1'M_2'\ \eta\ N}{M_1M_2\ \widehat{\eta}\ N} \qquad \frac{M\ \widehat{\eta}\ M' \quad \lambda x.M'\ \eta\ N}{\lambda x.M\ \widehat{\eta}\ N}$$

Notice that the candidate relation is not simply the contextual closure of η; this subtle definition of $\widehat{\eta}$ is necessary to guarantee the crucial Substitutivity Lemma 7.6.10. The following lemma is an easy consequence of the definition of $\widehat{\eta}$.

Lemma 7.6.9 *Let $\eta \subseteq \Lambda(C)\times\Lambda(C)$ be a reflexive and transitive Φ_{σ}^{F}-bisimulation. Then:*

i) *$\widehat{\eta}$ is reflexive.*

ii) *$\eta \subseteq \widehat{\eta}$.*

iii) *$\widehat{\eta}$ is a congruence w.r.t. application.*

iv) *$M\widehat{\eta}M' \ \wedge\ M'\eta N \implies M\widehat{\eta}N$.*

Lemma 7.6.10 (Substitutivity) *For all $M, M' \in \Lambda(C)$, $N, N' \in \delta(\Lambda(C))$,*

$$M\widehat{\eta}M' \ \wedge\ N\widehat{\eta}N' \implies M[N/x]\widehat{\eta}M'[N'/x]\ .$$

Proof By induction on the structure of M.

- $M \equiv x$: $\quad \dfrac{x\ \eta\ M'}{x\ \widehat{\eta}\ M'}$

$x\eta M' \implies N'\eta M'[N'/x]$, from the definition of η.
$N\widehat{\eta}N' \ \wedge\ N'\eta M'[N'/x] \implies N\widehat{\eta}M'[N'/x]$, from item iv of Lemma 7.6.9.

- $M \equiv M_1M_2$: $\exists M_1', M_2'$ s.t. $\quad \dfrac{M_1\ \widehat{\eta}\ M_1' \quad M_2\ \widehat{\eta}\ M_2' \quad M_1'M_2'\ \eta\ M'}{M_1M_2\ \widehat{\eta}\ M'}$

By induction hypothesis, $M_1[N/x]\widehat{\eta}M_1'[N'/x]$ and $M_2[N/x]\widehat{\eta}M_2'[N'/x]$. Moreover, by definition of η, $M_1'M_2'[N'/x]\eta M'[N'/x]$. Hence:

$$\frac{M_1[N/x]\ \widehat{\eta}\ M_1'[N'/x] \quad M_2[N/x]\ \widehat{\eta}\ M_2'[N'/x] \quad M_1'M_2'[N'/x]\ \eta\ M'[N'/x]}{M_1M_2[N/x]\ \widehat{\eta}\ M'[N'/x]}\ .$$

- $M \equiv \lambda y.M_1$: $\exists M_1'$ s.t. $\quad \dfrac{M_1\ \widehat{\eta}\ M_1' \quad \lambda y.M_1'\ \eta\ M'}{\lambda y.M_1\ \widehat{\eta}\ M'}$

By induction hypothesis, $M_1[N/x]\widehat{\eta}M'_1[N'/x]$. By definition of η, $(\lambda y.M'_1)[N'/x]\eta M'[N'/x]$. Hence:

$$\frac{M_1[N/x]\ \widehat{\eta}\ M'_1[N'/x] \quad (\lambda y.M'_1)[N'/x]\ \eta\ M'[N'/x]}{(\lambda y.M_1)[N/x]\ \widehat{\eta}\ M'[N'/x]} \ . \qquad \square$$

Thus, if we take η to be the equivalence $\approx^{app}_{\sigma}$, we get a relation $\widehat{\approx}^{app}_{\sigma}$, which, by item ii of Lemma 7.6.9, extends $\approx^{app}_{\sigma}$. Moreover, by item iii of the same lemma, it is a congruence w.r.t. application. In order to show that $\approx^{app}_{\sigma}$ is itself a congruence w.r.t. application, we prove that $(\widehat{\approx}^{app}_{\sigma})_{|\Lambda^0(C)\times\Lambda^0(C)} = (\approx^{app}_{\sigma})_{|\Lambda^0(C)\times\Lambda^0(C)}$. This is done using the coinduction principle of Theorem 7.2.4, by proving that $(\widehat{\approx}^{app}_{\sigma})_{|\Lambda^0(C)\times\Lambda^0(C)}$ is a Φ^F_{σ}-bisimulation. In order to prove that $(\widehat{\approx}^{app}_{\sigma})_{|\Lambda^0(C)\times\Lambda^0(C)}$ is a Φ^F_{σ}-bisimulation, it is sufficient to show that, for all $M, N \in \Lambda^0(C)$,

$$M\widehat{\approx}^{app}_{\sigma}N \ \wedge \ M \Downarrow_{\sigma} \implies N \Downarrow_{\sigma} \ .$$

Hence we can state the following

Theorem 7.6.11 *If, for all $M, N \in \Lambda^0(C)$,*

$$M\widehat{\approx}^{app}_{\sigma}N \ \wedge \ M \Downarrow_{\sigma} \implies N \Downarrow_{\sigma} \qquad (*) \ ,$$

then $\approx^{app}_{\sigma}$ is a congruence w.r.t. application.

The validity of the hypothesis $(*)$ of Theorem 7.6.11 depends on the particular λ-congruence. Here we show that hypothesis $(*)$ holds for all the λ-congruences derived from the strategies presented in Section 7.3.

The proof of this fact makes an essential use of the Substitutivity Lemma, and moreover, it requires the validity of some further properties, depending on the strategy $\to_{\sigma}$.

In order to make proofs uniform, we group the strategies of Section 7.3 under three general formats.

General Formats

Lazy Strategies. $\to_l$, $\to_v$ can be viewed as special cases of the general format of lazy strategy on a λ-calculus with variables by name and by values (see [How89, How96]).

Eager Leftmost Strategies. $\to_h$, $\to_n$, and $\to_p$ are eager in the sense that they reduce under the scope of a λ-abstraction. They can be viewed as special instances of the following general format:

$$\frac{M_{i_1} \Downarrow_{\sigma} M'_{i_1} \ \ldots \ M_{i_n} \Downarrow_{\sigma} M'_{i_n}}{xM_1\ldots M_k \Downarrow_{\sigma} xM'_1\ldots M'_k} \ i_1,\ldots,i_n \in \{1,\ldots,k\},\ n \geq 0 \qquad \frac{M \Downarrow_{\sigma} N}{\lambda x.M \Downarrow_{\sigma} \lambda x.N}$$

$$\frac{M[N/x]M_1\ldots M_n \Downarrow_{\sigma} V \quad (N \Downarrow_{\sigma})}{(\lambda x.M)NM_1\ldots M_n \Downarrow_{\sigma} V} \quad n \geq 0, \quad \text{where } (N \Downarrow_{\sigma}) \text{ can be omitted.}$$

Non-deterministic Strategies. $\to_o$ can be viewed as a special case of the following general format: let $\emptyset \subset Val \subset \Lambda(\{C\})$ be closed under β-reduction,

$$\frac{M \in Val}{M \Downarrow_\sigma M} \qquad\qquad \frac{C[(\lambda x.M)N] \notin Val \quad C[M[N/x]] \Downarrow_\sigma P}{C[(\lambda x.M)N] \Downarrow_\sigma P}$$

Notice that there are many ways to extend $\rightarrow_o$ on open terms in order to get a strategy of the above format; we will take the natural one.

Congruence Candidate Technique for Lazy Strategies

For the sake of completeness, we outline briefly the proof of the fact that $(\widehat{\approx}_\sigma^{app})_{|\Lambda^0(C)\times\Lambda^0(C)}$ is a Φ_σ^F-bisimulation for $\sigma = l, v$. The strategies $\rightarrow_l, \rightarrow_v$ are special cases of Howe's general format of lazy strategies, see [How89, How96] for more details.

Lemma 7.6.12 *For all* $M, N \in \Lambda^0$,
$(M \approx_v^{app} N \ \wedge \ M \Downarrow_v V) \implies \exists U.\ (N \Downarrow_v U \ \wedge \ V \approx_v^{app} U)$.

Proposition 7.6.13 $(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}$ *is a* Φ_σ^F*-bisimulation, for* $\sigma \in \{l, v\}$.

Proof (Sketch, see [How89, How96] for more details.) Let $M(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}N$. From items i and iii of Lemma 7.6.9 it follows immediately that, for all $P \in \Lambda^0$, $MP(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}NP$. The difficult part of the proof consists in proving that $M(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}N \ \wedge \ M \Downarrow_\sigma \implies N \Downarrow_\sigma$.
This can be shown by induction on the derivation of $M \Downarrow_\sigma$, using Lemmata 7.6.9, 7.6.10, and, for $\sigma = v$, also Lemma 7.6.12. $\square$

Congruence Candidate Technique for Eager Leftmost Strategies

Proposition 7.6.14 *Let* $\rightarrow_\sigma$ *be a eager leftmost strategy s.t.* $\approx_\sigma^{app}$ *is a congruence w.r.t.* λ*-abstraction. Then* $(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}$ *is a* Φ_σ^F*-bisimulation.*

Proof The only non trivial part of the proof consists in proving that
$$M(\widehat{\approx}_\sigma^{app})_{|\Lambda^0\times\Lambda^0}N \ \wedge \ M \Downarrow_\sigma \implies N \Downarrow_\sigma.$$
Since the evaluation relation is axiomatized on the whole Λ, the above fact cannot be proved simply by induction on the derivation of $M \Downarrow_\sigma$. However, it follows from the stronger result obtained by dropping the restriction on closed λ-terms, i.e.:
$$M\widehat{\approx}_\sigma^{app}N \ \wedge \ M \Downarrow_\sigma \implies N \Downarrow_\sigma.$$
To show this, we proceed by induction on the derivation of $M \Downarrow_\sigma$.

- $M \equiv xM_1 \dots M_k$: then, by hypothesis $\exists V_{i_1}, \dots, V_{i_n}$ s.t.

$$\frac{M_{i_1} \Downarrow_\sigma V_{i_1} \dots M_{i_n} \Downarrow_\sigma V_{i_n}}{xM_1 \dots M_k \Downarrow_\sigma xV_1 \dots V_k} \quad i_1, \dots, i_n \in \{1, \dots, k\},\ n \geq 0$$

and $\exists N_1, \dots, N_k, N^0, \dots, N^{k-1}$ s.t.

$$\dfrac{\dfrac{x \approx^{app}_{\sigma} N^0}{x \approxeq^{app}_{\sigma} N^0} \quad M_1 \approxeq^{app}_{\sigma} N_1 \quad N^0 N_1 \approx^{app}_{\sigma} N^1}{x M_1 \approxeq^{app}_{\sigma} N^1}$$

$$\vdots$$

$$\dfrac{x M_1 \ldots M_{k-1} \approxeq^{app}_{\sigma} N^{k-1} \qquad M_k \approxeq^{app}_{\sigma} N_k \qquad N^{k-1} N_k \approx^{app}_{\sigma} N}{x M_1 \ldots M_k \approxeq^{app}_{\sigma} N}$$

Hence
$x \approx^{app}_{\sigma} N^0 \implies x N_1 \approx^{app}_{\sigma} N^0 N_1$
$x N_1 \approx^{app}_{\sigma} N^0 N_1 \ \wedge\ N^0 N_1 \approx^{app}_{\sigma} N^1 \implies x N_1 \approx^{app}_{\sigma} N^1$

$$\vdots$$

$x N_1 \ldots N_k \approx^{app}_{\sigma} N^{k-1} N_k \ \wedge\ N^{k-1} N_k \approx^{app}_{\sigma} N \implies x N_1 \ldots N_k \approx^{app}_{\sigma} N$.
By induction hypothesis, from $M_{i_1} \approxeq^{app}_{\sigma} N_{i_1}, \ldots, M_{i_n} \approxeq^{app}_{\sigma} N_{i_n}$, it follows that $N_{i_1} \Downarrow_{\sigma}, \ldots, N_{i_n} \Downarrow_{\sigma}$. Thus $x N_1 \ldots N_k \Downarrow_{\sigma}$. Hence, from $x N_1 \ldots N_k \approx^{app}_{\sigma} N$, using the fact that $\approx^{app}_{\sigma}$ is a congruence w.r.t. λ-abstraction, it follows that $N \Downarrow_{\sigma}$.

- $M \equiv \lambda x.M_1$: then, by hypothesis $\exists V_1$ s.t. $\dfrac{M_1 \Downarrow_{\sigma} V_1}{\lambda x.M_1 \Downarrow_{\sigma} \lambda x.V_1}$

and $\exists N_1$ s.t. $\dfrac{M_1 \approxeq^{app}_{\sigma} N_1 \quad \lambda x.N_1 \approx^{app}_{\sigma} N}{\lambda x.M_1 \approxeq^{app}_{\sigma} N}$.

By induction hypothesis $N_1 \Downarrow_{\sigma}$. Hence $\lambda x.N_1 \Downarrow_{\sigma}$. Thus, from $\lambda x.N_1 \approx^{app}_{\sigma} N$, using the fact that $\approx^{app}_{\sigma}$ is a congruence w.r.t. λ-abstraction, $N \Downarrow_{\sigma}$.

- $M \equiv (\lambda x.M_1) M_2 \ldots M_k$: then, by hypothesis $\exists V$ s.t.

$$\dfrac{M_1[M_2/x] M_3 \ldots M_k \Downarrow_{\sigma} V \quad (M_2 \Downarrow_{\sigma})}{(\lambda x.M_1) M_2 \ldots M_k \Downarrow_{\sigma} V} \quad k \geq 2$$

and $\exists N_1, \ldots, N_k, N^1, \ldots, N^{k-1}$ s.t.

$$\dfrac{\dfrac{M_1 \approxeq^{app}_{\sigma} N_1 \quad \lambda x.N_1 \approx^{app}_{\sigma} N^1}{\lambda x.M_1 \approxeq^{app}_{\sigma} N^1} \quad M_2 \approxeq^{app}_{\sigma} N_2 \quad N^1 N_2 \approx^{app}_{\sigma} N^2}{(\lambda x.M_1) M_2 \approxeq^{app}_{\sigma} N^2}$$

$$\vdots$$

$$\dfrac{(\lambda x.M_1) M_2 \ldots M_{k-1} \approxeq^{app}_{\sigma} N^{k-1} \qquad M_k \approxeq^{app}_{\sigma} N_k \qquad N^{k-1} N_k \approx^{app}_{\sigma} N}{(\lambda x.M_1) M_2 \ldots M_k \approxeq^{app}_{\sigma} N}$$

Hence
$N^{k-2} N_{k-1} \approx^{app}_{\sigma} N^{k-1} \ \wedge\ N^{k-1} N_k \approx^{app}_{\sigma} N \implies$
$\qquad N^{k-2} N_{k-1} N_k \approx^{app}_{\sigma} N$
$N^{k-3} N_{k-2} \approx^{app}_{\sigma} N^{k-2} \ \wedge\ N^{k-2} N_{k-1} N_k \approx^{app}_{\sigma} N \implies$
$\qquad N^{k-3} N_{k-2} N_{k-1} N_k \approx^{app}_{\sigma} N$

$$\vdots$$

$N^1 N_2 \approx^{app}_{\sigma} N^2 \ \wedge\ N^2 N_3 \ldots N_k \approx^{app}_{\sigma} N \implies N^1 N_2 \ldots N_k \approx^{app}_{\sigma} N$
$\lambda x.N_1 \approx^{app}_{\sigma} N^1 \ \wedge\ N^1 N_2 \ldots N_k \approx^{app}_{\sigma} N \implies (\lambda x.N_1) N_2 \ldots N_k \approx^{app}_{\sigma} N$.

To show that $N \Downarrow_\sigma$, it is sufficient to prove that $(\lambda x.N_1)N_2 \ldots N_k \Downarrow_\sigma$. Then, from the definition of $(\approx_\sigma^{app}$, since $\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction, we get the thesis. To show that $(\lambda x.N_1)N_2 \ldots N_k \Downarrow_\sigma$, it is sufficient to prove that $N_1[N_2/x]N_3 \ldots N_k \Downarrow_\sigma$, and possibly also that $N_2 \Downarrow_\sigma$. This latter fact follows by induction hyp.. To show $N_1[N_2/x]N_3 \ldots N_k \Downarrow_\sigma$, we proceeds as follows. From $M_1 \widehat{\approx}_\sigma^{app} N_1, \ldots, M_k \widehat{\approx}_\sigma^{app} N_k$, using the Substitutivity Lemma, we get $M_1[M_2/x]M_3 \ldots M_k \widehat{\approx}_\sigma^{app} N_1[N_2/x]N_3 \ldots N_k$. Since $M_1[M_2/x]M_3 \ldots M_k \Downarrow_\sigma$, by induction hypothesis, $N_1[N_2/x]N_3 \ldots N_k \Downarrow_\sigma$. $\square$

Congruence Candidate Technique for Non-deterministic Strategies

Since Val_σ is closed under β-reduction, for $\rightarrow_\sigma$ non-deterministic strategies of the format of Section 7.6.2, we immediately get, using Proposition 7.1.5:

Lemma 7.6.15 *Let $\rightarrow_\sigma$ be a non-deterministic strategy. Then β-reduction is correct w.r.t. $\approx_\sigma$.*

Lemma 7.6.16 *Let $\rightarrow_\sigma$ be a non-deterministic strategy. For all contexts $C[\ ]$, if $C[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N$, then $C[P[Q/x]] \widehat{\approx}_\sigma^{app} N$.*

Proof The proof proceeds by induction on the structure of $C[\ ]$.
• $C[\ ] \in Var$: the thesis is immediate.
• $C[\ ] \equiv [\ ]$: from the hypothesis $(\lambda x.P)Q \widehat{\approx}_\sigma^{app} N$, $\exists N_1, N_2, N_3$ s.t.

$$\frac{\dfrac{P \widehat{\approx}_\sigma^{app} N_1 \qquad \lambda x.N_1 \approx_\sigma^{app} N_2}{\lambda x.P \widehat{\approx}_\sigma^{app} N_2} \qquad Q \widehat{\approx}_\sigma^{app} N_3 \qquad N_2 N_3 \approx_\sigma^{app} N}{(\lambda x.P)Q \widehat{\approx}_\sigma^{app} N}$$

$\lambda x.N_1 \approx_\sigma^{app} N_2 \ \wedge\ N_2 N_3 \approx_\sigma^{app} N \implies (\lambda x.N_1)N_3 \approx_\sigma^{app} N$;
using Lemma 7.6.15, we get $N_1[N_3/x] \approx_\sigma^{app} N$; moreover, by the Substitutivity Lemma, $P \widehat{\approx}_\sigma^{app} N_1 \ \wedge\ Q \widehat{\approx}_\sigma^{app} N_3 \implies P[Q/x] \widehat{\approx}_\sigma^{app} N_1[N_3/x]$.
Hence, from $P[Q/x] \widehat{\approx}_\sigma^{app} N_1[N_3/x]$ and $N_1[N_3/x] \approx_\sigma^{app} N$, using item iv of Lemma 7.6.9, it follows that $P[Q/x] \widehat{\approx}_\sigma^{app} N$.
• $C[\ ] \equiv C_1[\ ]C_2[\ ]$: from the hyp. $C_1[(\lambda x.P)Q]C_2[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N$, $\exists N_1, N_2$ s.t.

$$\frac{C_1[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N_1 \quad C_2[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N_2 \quad N_1 N_2 \approx_\sigma^{app} N}{C_1[(\lambda x.P)Q]C_2[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N}\ .$$

By induction hypothesis, $C_1[P[Q/x]] \widehat{\approx}_\sigma^{app} N_1$ and $C_2[P[Q/x]] \widehat{\approx}_\sigma^{app} N_2$, hence $C_1[P[Q/x]]C_2[P[Q/x]] \widehat{\approx}_\sigma^{app} N_1 N_2$. Then, from $N_1 N_2 \approx_\sigma^{app} N$, using item iv of Lemma 7.6.9, we get the thesis.
• $C[\ ] \equiv \lambda y.C_1[\ ]$: from the hypothesis $\lambda y.C_1[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N$, $\exists N_1$ s.t.

$$\frac{C_1[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N_1 \quad \lambda y.N_1 \approx_\sigma^{app} N}{\lambda y.C_1[(\lambda x.P)Q] \widehat{\approx}_\sigma^{app} N}\ .$$

By induction hypothesis, $C_1[P[Q/x]] \widehat{\approx}_\sigma^{app} N_1$, hence $\lambda y.C_1[P[Q/x]] \widehat{\approx}_\sigma^{app} \lambda y.N_1$.

Then, from $\lambda y.N_1 \approx_\sigma^{app} N$, using item iv of Lemma 7.6.9, we get the thesis. $\square$

As we remarked earlier, the proof of the fact that $(\lessapprox_\sigma^{app})_{|\Lambda^0 \times \Lambda^0}$ is a Φ_σ^F-bisimulation depends essentially on the strategy. The hypotheses of the proposition below have been tuned to the strategy $\rightarrow_o$. Different sets of hypotheses are probably necessary to deal with other non-deterministic strategies.

Proposition 7.6.17 *Let $\rightarrow_\sigma$ be a non-deterministic strategy s.t.:*

1. *$\approx_\sigma^{app}$ is a congruence w.r.t. λ-abstraction;*
2. *for all $M \in \Lambda(C)$,*

   i) $M \Downarrow_\sigma \iff \lambda x.M \Downarrow_\sigma$ *and*

   ii) $\lambda x.M \in Val_\sigma \implies M \in Val_\sigma$*;*
3. *for all $M_1, M_2 \in \Lambda(C)$,*

   i) $(M_1 \Downarrow_\sigma \;\wedge\; M_2 \Downarrow_\sigma) \implies M_1 M_2 \Downarrow_\sigma$ *and*

   ii) $M_1 M_2 \in Val_\sigma \implies (M_1 \in Val_\sigma \;\wedge\; M_2 \in Val_\sigma)$,

then $(\lessapprox_\sigma^{app})_{|\Lambda^0(C)\times\Lambda^0(C)}$ is a Φ_σ^F-bisimulation.

Proof We prove, by induction on the minimal length k of a convergent path from $M \in \Lambda(C)$, that: $\quad M \lessapprox_\sigma^{app} N \;\wedge\; M \Downarrow_\sigma \implies N \Downarrow_\sigma$.

- Suppose $k = 0$. Then we proceed by induction on the structure of M:
  - $M \equiv x$: $\quad \dfrac{x \approx_\sigma^{app} N}{x \lessapprox_\sigma^{app} N}$; from $x \approx_\sigma^{app} N$, using hypotheses 1 and 2i), we get $N \Downarrow_\sigma$.
  - $M \equiv \lambda x.M_1$: $\quad \exists N_1$ s.t. $\quad \dfrac{M_1 \lessapprox_\sigma^{app} N_1 \quad \lambda x.N_1 \approx_\sigma^{app} N}{\lambda x.M_1 \lessapprox_\sigma^{app} N}$; hence, by hypothesis 2ii), $M_1 \in Val_\sigma$; from $M_1 \lessapprox_\sigma^{app} N_1$, using the induction hypothesis, it follows that $N_1 \Downarrow_\sigma$. Hence by hypothesis 2i) $\lambda x.N_1 \Downarrow_\sigma$, and, by hypotheses 1 and 2i), $N \Downarrow_\sigma$.
  - $M \equiv M_1 M_2$: $\quad \exists N_1, N_2$ s.t. $\quad \dfrac{M_1 \lessapprox_\sigma^{app} N_1 \quad M_2 \lessapprox_\sigma^{app} N_2 \quad N_1 N_2 \approx_\sigma^{app} N}{M_1 M_2 \lessapprox_\sigma^{app} N}$
    Since, by hypothesis 3ii), $M_1, M_2 \in Val_\sigma$, by induction hypothesis, $N_1 \Downarrow_\sigma$ and $N_2 \Downarrow_\sigma$, i.e., by hypothesis 3i), $N_1 N_2 \Downarrow_\sigma$. Hence, by hypotheses 1 and 2i), $N \Downarrow_\sigma$.
- Suppose $k > 0$. $M \equiv C[(\lambda x.P)Q] \rightarrow_\sigma C[P[Q/x]] \Downarrow_\sigma$ (the length of a minimal convergent path from $C[P[Q/x]]$ is $k-1$). From $C[(\lambda x.P)Q] \lessapprox_\sigma^{app} N$, by Lemma 7.6.16, it follows that $C[P[Q/x]] \lessapprox_\sigma^{app} N$. Hence, by induction hypothesis, $N \Downarrow_\sigma$.

$\square$

Corollary 7.6.18 $(\lessapprox_o^{app})_{|\Lambda^0\times\Lambda^0}$ *is a Φ_o^F-bisimulation.*

Proof We extend $\rightarrow_o$ on open terms in such a way that a (possibly open) λ-term converges if and only if there exists a β-reduction path to a (possibly open) λ-term not containing any occurrence of Ω. Then Proposition 7.6.17 is applicable. $\square$

7.6.3 The Logical Relations Method

The logical relations method is semantical, in the sense that it is based on a suitable *computationally adequate* CPO-λ-model. This method is the generalization of the method originally used by Pitts in [Pit94, Pit96, Pit96a] for the special cases of by-name and by-value lazy reduction strategies. Pitts' technique is based on the following *minimal invariance property* of the model, as is the case for initial models: D^σ is an invariant object of a suitable functor F_σ on the category $CPO_\perp$ via the isomorphism $i_\sigma : D^\sigma \to F_\sigma(D^\sigma)$, and, moreover, id_{D^σ} is the least fixed point of the continuous function $\delta_\sigma : [D^\sigma \to D^\sigma] \to [D^\sigma \to D^\sigma]$ defined as $\delta_\sigma(e) = i_\sigma^{-1} \circ F_\sigma(e) \circ i_\sigma$. Categorically, this amounts to say that D^σ is the initial algebra for a suitable functor on the algebraically compact category $CPO_\perp$ ([Fre90]). In [Len97a], it is shown how to extend it to many more strategies and models; in particular, all the λ-congruences induced by the strategies of Section 7.3 can be dealt with. In this section we improve on [Len97] by capturing a set of general properties of models which make the method work. We start by identifying a suitable class of c.p.o.'s models, the *finitary $\approx_\sigma$-models.*

Definition 7.6.19 (Finitary Applicative Structure) *A* finitary applicative structure *is a pair $(D, \bullet_D)$ such that*

- *$(D, \sqsubseteq_D)$ is a c.p.o.;*
- *$\bullet_D : D \times D \to D$ is continuous;*
- *there exist* projection *functions, i.e. functions $\pi_n : D \to D$, such that $\forall n \geq 0.\ \pi_n \sqsubseteq_D id_D$ and $\forall n, m \geq 0.\ \pi_m \circ \pi_n = \pi_{min\{m,n\}}$; for all $d \in D$, we will use d_n for denoting $\pi_n(d)$, and D_n for denoting the set $\{\pi_n(d) \mid d \in D\}$;*
- *$\forall d \in D\ \forall n > 0.\ d_n \bullet_D d =_D (d \bullet_D d_{n-1})_{n-1}$, where $=_D$ denotes $\sqsubseteq_D \cap (\sqsubseteq_D)^{-1}$;*
- *$\forall d \in D.\ d =_D \bigsqcup_n d_n$;*
- *$d =_D e \iff \forall n.\ d_n =_D e_n$,*

Definition 7.6.20 (Finitary $\approx_\sigma$-Model) *Let $\approx_\sigma$ be a λ-congruence. A finitary $\approx_\sigma$-model is a triple $(D, \bullet_D, [\![\ ]\!]^D)$, where*

1. *$(D, \bullet_D)$ is a finitary applicative structure;*

2. $[\![\;]\!] : \Lambda \times Env \to D$, *where* $Env = [Var \to E]$,

$$E = \begin{cases} V & \text{if } \approx_\sigma \text{ is right strict,} \\ D & \text{otherwise,} \end{cases}$$

and $V = D \setminus \{\bot\}$ *is the set of semantic values;*

3. $[\![x]\!]^D_\rho =_D \rho(x)$;

4. $[\![c]\!]^D_\rho =_D d_c$, *for some* $d_c \in D$;

5. $[\![MN]\!]^D_\rho =_D [\![M]\!]^D_\rho \bullet_D [\![N]\!]^D_\rho$;

6. $\forall d \in E.\ [\![\lambda x.M]\!]^D_\rho \bullet_D d =_D [\![M]\!]^D_{\rho[d/x]}$.

In the sequel, when clear from the context, we will simply denote $=_D$ by $=$.

Here we formulate the logical relations method for finitary $\approx_\sigma$-models. We will show that, if there exists a finitary $\approx_\sigma$-model satisfying a further technical property, then $\approx^{app}_\sigma$ is a congruence w.r.t. application. As by-product of this, we get that, in case $\approx^{app}_\sigma$ is also a congruence w.r.t. λ-abstraction, then the finitary $\approx_\sigma$-model is computationally adequate w.r.t. $\approx_\sigma$.

The core of the logical relations method consists in defining a suitable relation $\lhd^\sigma \subseteq D^\sigma \times \Lambda^0(C)$, where D^σ is a finitary $\approx_\sigma$-model. Then, one proceeds to show that

1. $\lhd^\sigma$ is a congruence w.r.t. application;

2. $M \leq^{app}_\sigma N \iff [\![M]\!]^{D^\sigma} \lhd^\sigma N$.

Notation In what follows, we will denote by $\mathcal{E}$ the union $\bigcup_{n\in\mathbf{N}}(D^\sigma_n \times \Lambda(C))$.

The relation $\lhd^\sigma$ is defined inductively in terms of the relations $\lhd^\sigma_n \subseteq D^\sigma_n \times \Lambda^0(C)$ as follows:

Definition 7.6.21 • *The relations* $\unlhd^\sigma_n \subseteq D^\sigma_n \times \Lambda^0(C)$ *are defined inductively as follows:*

$$\unlhd^\sigma_0 = \{(d,P) \mid P \in \Lambda^0(C) \ \wedge\ d \sqsubseteq_{D^\sigma} ([\![P]\!]^{D^\sigma})_0\}$$
$$\begin{aligned}\unlhd^\sigma_{n+1} = \{(d,P) \in D^\sigma_{n+1} \times \Lambda^0(C) \mid\ & (d_0 =_{D^\sigma} \bot \ \wedge\ \forall (d',P') \in \unlhd^\sigma_n.\ (dd',PP') \in \unlhd^\sigma_n) \vee \\ & (d_0 \neq_{D^\sigma} \bot \ \wedge\ P \Downarrow_\sigma \ \wedge\ \forall (d',P') \in \unlhd^\sigma_n.\ (dd',PP') \in \unlhd^\sigma_n)\}.\end{aligned}$$

• *Let* $\unlhd^\sigma \subseteq \mathcal{E}$ *be defined by* $\unlhd^\sigma = \bigcup_{n\in\mathbf{N}} \unlhd^\sigma_n$.

• *Let* $\lhd^\sigma_n = \{(d,P) \in D^\sigma \times \Lambda^0(C) \mid d_n \unlhd^\sigma_n P\}$.

• *Finally, let* $\lhd^\sigma = \bigcap_{n\in\mathbf{N}} \lhd^\sigma_n$.

The following definition is instrumental:

Definition 7.6.22 *A relation* $\mathcal{R} \subseteq D^\sigma \times \Lambda^0(C)$ *is called* limit-closed *if, whenever for all* $n \in \mathbf{N}$ $(d_n, P) \in \mathcal{R}$*, then also* $(d, P) \in \mathcal{R}$.
Given a relation $\mathcal{R} \subseteq D^\sigma \times \Lambda^0(C)$*, we will denote by* $\overline{\mathcal{R}}$ *the least limit-closed relation including* $\mathcal{R}$.

In the following definition we characterize the class of finitary $\approx_\sigma$-models to which the logical relations method applies:

Definition 7.6.23 ($\lhd^\sigma$-Adequate Finitary $\approx_\sigma$-Models) *A* $\lhd^\sigma$*-adequate finitary* $\approx_\sigma$*-model is a finitary* $\approx_\sigma$*-model such that*

1. *for all* $d \in D^\sigma_0$*, for all* $P \in \Lambda^0(C)$*,*
$$d \unlhd^\sigma_0 P \implies d \unlhd^\sigma_1 P \ .$$
2. *for all* $d \in D^\sigma_1$*, for all* $P \in \Lambda^0(C)$*,*
$$d \unlhd^\sigma_1 P \implies d_0 \unlhd^\sigma_0 P \ .$$

Notice that the first condition in Definition 7.6.23 is satisfied when we assume that the finitary $\approx_\sigma$-model is computationally adequate and the computational adequacy of the model can be decided in D^σ_0, i.e.:

Definition 7.6.24 *Let* D^σ *be a finitary* $\approx_\sigma$*-model. Then* D^σ *is* 0-computationally adequate *if*

$$M \Downarrow_\sigma \iff \exists \rho \in Env.\ ([\![M]\!]^D_\rho)_0 \neq_D \bot \ .$$

Lemma 7.6.25 *If* D^σ *is a finitary 0-computationally adequate* $\approx_\sigma$*-model, then for all* $d \in D^\sigma_0$*, for all* $P \in \Lambda^0(C)$*,*

$$d \unlhd^\sigma_0 P \implies d \unlhd^\sigma_1 P \ .$$

The following lemma is instrumental:

Lemma 7.6.26 *Let* D^σ *be a* $\lhd^\sigma$*-adequate finitary* $\approx_\sigma$*-model. Then*

i) $\forall d \in D^\sigma_n, \forall P \in \Lambda^0(C),\ d \unlhd^\sigma_n P \implies \forall m.\ d_m \unlhd^\sigma_m P$.

ii) $\lhd^\sigma = \overline{\unlhd^\sigma}$.

Proof i) By hypothesis $d \unlhd^\sigma_0 P \Rightarrow d \unlhd^\sigma_1 P$. Moreover, one can prove, by induction on n, that, for all $n \geq 0$,

$$d \unlhd^\sigma_n P \implies (d_{n-1} \unlhd^\sigma_{n-1} P \ \wedge\ d \unlhd^\sigma_{n+1} P) \ .$$

The non trivial part in proving this fact is the base case, i.e. $d \unlhd^\sigma_1 P \implies (d_0 \unlhd^\sigma_0 P \ \wedge\ d \unlhd^\sigma_2 P)$. In particular, $d \unlhd^\sigma_1 P \implies d \unlhd^\sigma_2 P$ amounts to show

that, for all $d, d' \in D_1^\sigma$, $P, P' \in \Lambda^0(C)$, $d \trianglelefteq_1^\sigma P \wedge d' \trianglelefteq_1^\sigma P' \implies dd' \trianglelefteq_1^\sigma PP'$. But this follows since the model is $\triangleleft^\sigma$-adequate. In fact, from $d' \trianglelefteq_1^\sigma P'$, by the second condition of Definition 7.6.23, $d_0' \trianglelefteq_0^\sigma P'$, hence $dd' =_{D^\sigma} dd_0' \trianglelefteq_0^\sigma PP'$, and, by the first condition of Definition 7.6.23, $dd' \trianglelefteq_1^\sigma PP'$.
ii) The thesis follows from item i). □

Now we introduce the operator T_σ. We will prove that the least fixed point of T_σ is $(\trianglelefteq^\sigma, \trianglelefteq^\sigma)$.

Proposition 7.6.27 *i) Let $\mathcal{R}_\sigma$ be defined as follows:*
$\mathcal{R}_\sigma = \{((\mathcal{R}^-, \mathcal{R}^+), (\supseteq, \subseteq)) \mid \mathcal{R}^-, \mathcal{R}^+ \subseteq \mathcal{E} \wedge \trianglelefteq_0^\sigma \subseteq \mathcal{R}^-, \mathcal{R}^+ \wedge A \cap \mathcal{R}^- = \emptyset \wedge A \cap \mathcal{R}^+ = \emptyset\}$,
where $A = \{(d, P) \in D_0^\sigma \times \Lambda^0(C) \mid d \not\trianglelefteq_0^\sigma P\}$. Then $\mathcal{R}_\sigma$ is a complete lattice.
ii) The following operator $T_\sigma : \mathcal{R}_\sigma \to \mathcal{R}_\sigma$ is well defined:

$$\begin{aligned}
&T_\sigma(\mathcal{R}^-, \mathcal{R}^+) = \\
&(\{(d,P) \mid d \trianglelefteq_0^\sigma P \vee (d_0 =_{D^\sigma} \bot \wedge \forall (d', P') \in \mathcal{R}^+ . (dd', PP') \in \mathcal{R}^-) \vee \\
&\qquad (d_0 \neq_{D^\sigma} \bot \wedge P \Downarrow_\sigma \wedge \forall (d', P') \in \mathcal{R}^+ . (dd', PP') \in \mathcal{R}^-)\} \cap \mathcal{E}, \\
&\{(d,P) \mid d \trianglelefteq_0^\sigma P \vee (d_0 =_{D^\sigma} \bot \wedge \forall (d', P') \in \mathcal{R}^+ . (dd', PP') \in \mathcal{R}^-) \vee \\
&\qquad (d_0 \neq_{D^\sigma} \bot \wedge P \Downarrow_\sigma \wedge \forall (d', P') \in \mathcal{R}^+ . (dd', PP') \in \mathcal{R}^-)\} \cap \mathcal{E}).
\end{aligned}$$

Theorem 7.6.28 *Let D^σ be a $\triangleleft^\sigma$-adequate finitary $\approx_\sigma$-model. Then*

i) *$(\trianglelefteq^\sigma, \trianglelefteq^\sigma)$ is the least fixed point of T_σ.*

ii) *The following induction-coinduction principle holds:*

$$\frac{\mathcal{R}^- \subseteq \pi_1(T_\sigma(\mathcal{R}^-, \mathcal{R}^+)) \qquad \mathcal{R}^+ \supseteq \pi_2(T_\sigma(\mathcal{R}^-, \mathcal{R}^+))}{\overline{\mathcal{R}^-} \subseteq \triangleleft^\sigma \subseteq \overline{\mathcal{R}^+}} \; .$$

Proof i) Using Lemma 7.6.26i), one can prove that $(\trianglelefteq^\sigma, \trianglelefteq^\sigma)$ is a fixed point of T_σ. Moreover, one can check that, for all $n > 0$,
$T_\sigma^{n+1}(\mathcal{E} \setminus A, \trianglelefteq_0^\sigma) = (\triangleleft_n^\sigma \cap \mathcal{E}, \trianglelefteq_n^\sigma)$.
This follows from the fact that, for $n > 0$, $T(\triangleleft_n^\sigma \cap \mathcal{E}, \trianglelefteq_n^\sigma) = (\triangleleft_{n+1}^\sigma \cap \mathcal{E}, \trianglelefteq_{n+1}^\sigma)$.
Hence the least fixed point of T_σ is
$(\bigcap_{n \in \mathbf{N}} \pi_1(T_\sigma^n(\mathcal{E} \setminus A, \trianglelefteq_0^\sigma)), \bigcup_{n \in \mathbf{N}} \pi_2(T_\sigma^n(\mathcal{E} \setminus A, \trianglelefteq_0^\sigma))) = (\triangleleft^\sigma \cap \mathcal{E}, \trianglelefteq^\sigma) = (\trianglelefteq^\sigma, \trianglelefteq^\sigma)$.
ii) The proof follows from Lemma 7.6.26ii) and item i). □

The relation $\triangleleft^\sigma$ is a congruence w.r.t. application:

Theorem 7.6.29 *Let D^σ be an finitary $\approx_\sigma$-model. Then*

$$d \triangleleft^\sigma P \iff \forall e \triangleleft^\sigma Q . \; de \triangleleft^\sigma PQ \; .$$

Proof ($\Rightarrow$) The proof follows from the fact that, since D^σ is an finitary $\approx_\sigma$-model, application in the model D^σ satisfies the following property: for all $d,e \in D^\sigma$, $de = \bigsqcup_{n\in\mathbf{N}} d_{n+1}e_n$. Suppose that $d \lhd^\sigma P$ and $e \lhd^\sigma Q$. We will prove that, for all n, $(de)_n \lhd^\sigma PQ$. From the hypotheses and from the definition of $\lhd^\sigma$, using the property of application, we immediately get, for all n,k, $((\bigsqcup_{m\in\mathbf{N}} d_{m+1}e_m)_n)_k \trianglelefteq^\sigma_k PQ$, i.e., from the definition of $\lhd^\sigma$, we have, for all n, $(\bigsqcup_{m\in\mathbf{N}} d_{m+1}e_m)_n \lhd^\sigma PQ$.
($\Leftarrow$) From the Definition of $\lhd^\sigma$. $\square$

The proof of the following lemma makes use of the induction-coinduction principle of Theorem 7.6.28:

Lemma 7.6.30 *Let D^σ be a $\trianglelefteq^\sigma$-adequate finitary $\approx_\sigma$-model. For all $d,d' \in D^\sigma$, and for all $P,P' \in \Lambda^0(C)$,*

$$d \sqsubseteq_{D^\sigma} d' \lhd^\sigma P' \leq^{app}_\sigma P \implies d \lhd^\sigma P \; .$$

Proof Apply the induction-coinduction principle to $\mathcal{R}^+ = \trianglelefteq^\sigma$, $\mathcal{R}^- = \{(d,P) \mid \exists d' \in D^\sigma \; \exists P' \in \Lambda^0(C).\; d \sqsubseteq_{D^\sigma} d' \lhd^\sigma P' \leq^{app}_\sigma P\} \cap \mathcal{E}$. $\square$

The following theorem is crucial:

Theorem 7.6.31 *Let D^σ be a $\lhd^\sigma$-adequate finitary $\approx_\sigma$-model. Let $M \in \Lambda(C)$, with $FV(M) \subseteq \{x_1,\ldots,x_n\}$. Then*

$$\forall i = 1,\ldots,n.\; (d_i \in E^\sigma \;\wedge\; d_i \lhd^\sigma P_i) \implies [\![M]\!]^{D^\sigma}_{\rho[d_i/x_i]} \lhd^\sigma M[P_i/x_i] \; ,$$

where

$$E^\sigma = \begin{cases} D^\sigma \setminus \{\bot\} & \text{if } \approx_\sigma \text{ is right strict} \\ D^\sigma & \text{otherwise.} \end{cases}$$

Proof The proof is by induction on M. For dealing with the application case, we use Theorem 7.6.29; for dealing with the abstraction case, we use property 6 of Definition 7.6.20. $\square$

Corollary 7.6.32 *Let D^σ be a $\lhd^\sigma$-adequate finitary $\approx_\sigma$-model. Then, for all $P \in \Lambda^0(C)$,*

$$[\![P]\!]^{D^\sigma} \lhd^\sigma P \; .$$

Lemma 7.6.33 *Let D^σ be a $\lhd^\sigma$-adequate finitary $\approx_\sigma$-model. Then*

$$[\![P']\!]^{D^\sigma} \lhd^\sigma P \iff P' \leq^{app}_\sigma P \; .$$

Proof The implication ($\Rightarrow$) can be proved by coinduction. In fact it is easy to check, using Corollary 7.6.32 and Theorem 7.6.29 ($\Rightarrow$), that $\{(P',P) \mid [\![P']\!]^{D^\sigma} \lhd^\sigma P\}$ is a $\Phi^{\leq}_\sigma$-bisimulation (see Section 7.2, Lemma 7.2.7). The proof of the implication ($\Leftarrow$) follows from Lemma 7.6.30, and from Corollary 7.6.32. $\square$

Finally, we are in the position of stating the main theorem:

Theorem 7.6.34 *Let D^σ be a $\lhd^\sigma$-adequate finitary $\approx_\sigma$-model. Then*

- *$\leq_\sigma^{app}$ is a congruence w.r.t. application.*
- *If $\leq_\sigma^{app}$ is a congruence w.r.t. λ-abstraction, then D^σ is computationally adequate.*

Proof i) From Theorem 7.6.29 and Lemma 7.6.33 it follows that $\leq_\sigma^{app}$ is a congruence w.r.t. application.
ii) Using Corollary 7.6.32 and Lemma 7.6.30, one can show that, if $[\![M]\!]^{D^\sigma} \sqsubseteq_\sigma [\![N]\!]^{D^\sigma}$, then $[\![M]\!]^{D^\sigma} \lhd^\sigma N$, hence by Lemma 7.6.33 $M \leq_\sigma^{app} N$. Then, by i) and Theorem 7.5.1, we have $M \leq_\sigma N$. $\square$

The Logical Relations Method at Work

In order to apply the logical relations method to all the λ-congruences of Section 7.3, we have to prove that the corresponding models of Section 7.4 are finitary $\lhd^\sigma$-adequate $\approx_\sigma$-models. The only non trivial fact to verify is that these models are $\lhd^\sigma$-adequate.

Lemma 7.6.35 *For all $\sigma = v, l, o, h, n, p$, D^σ is an finitary $\lhd^\sigma$-adequate $\approx_\sigma$-model.*

Proof For all $\sigma \neq h$, the first condition of Definition 7.6.23 follows from Lemma 7.6.25. For $\sigma = h$ simply observe that if $d \lhd_0^\sigma P$, then $d =_h \perp$, since for all $P \in \Lambda^0$, $1 \not\sqsubseteq_{D^\sigma} [\![P]\!]_0^h$. In order to show that the second condition of Definition 7.6.23 holds, for all σ's, one can proceed by case inspection. $\square$

7.6.4 The Domain Logic Method

This is a semantical method for showing that $\approx_\sigma^{app}$ is a congruence w.r.t. application, based on the logical description via intersection types of a computationally adequate CPO-model. This method was originally introduced by Abramsky and Ong in [AO93] for the λ-congruence induced by the lazy strategy. A generalization of Abramsky and Ong's method is presented in [Len97a]. The method in [Len97a] generalizes that in [AO93] in two main respects, in order to make it applicable to many more λ-theories:

- in place of λ-model, the weaker notion of σ-combinatory algebra is introduced;
- the condition on type interpretation on applicative structures is relaxed, by requiring only that the type interpretation be *adequate*. In fact, in order to show the crucial Theorem 7.6.51, a uniformly defined notion of type interpretation on the whole class of applicative structures (like in [AO93]) is not necessary. An *adequate* type interpretation on the applicative structure $\mathcal{A}$ is enough.

The domain logic method introduced in [Len97a] applies to all the λ-congruences induced by the strategies in Section 7.3, but in [Len97a], only the details for $\approx_v$ are worked out. Here we review in detail the general theory, and then we work out the details for all the strategies of Section 7.3.

The General Theory of the Domain Logic Method

In this section, we review the general theory of the domain logic method. The theory that we develop is rather difficult. We can summarize the content of this section as follows. First of all we introduce the notion of *applicative σ-structure with convergence*, that is an applicative structure on which is defined a notion of *applicative equivalence*, arising from the notion of convergence. On applicative structures we can interpret the intersection type theory $\mathcal{T}_\sigma$ corresponding to the $\approx_\sigma$-computationally adequate model D^σ. The interesting type interpretations are those which are *adequate* (Definition 7.6.38), since, up to a first approximation, for applicative structures with an adequate interpretation, which moreover are *approximable*, the applicative equivalence is a congruence w.r.t. application. Then we introduce the key-notion of *combinatory σ-algebra*, which generalizes the standard notion of combinatory λ-algebra, and we state the Soundness Theorem 7.6.48 for combinatory σ-algebras. w.r.t. the type theory $\mathcal{T}_\sigma$, and the Completeness Theorem 7.6.49, which generalizes the Completeness Theorem of [BCD83]. Up to a first approximation, our main result states that, if the model D^σ is a combinatory σ-algebra and $\mathcal{A}$ is a combinatory σ-algebra with an *adequate* $\mathcal{T}_\sigma$ type interpretation for which the Soundness Theorem holds, then $\mathcal{A}$ is approximable, i.e. the applicative equivalence on $\mathcal{A}$ is a congruence w.r.t. application. Hence, in order to show that the applicative equivalence induced by the λ-theory $\approx_\sigma$ is a congruence w.r.t. application, we have to endow the set of λ-terms with a structure of combinatory σ-algebra with an adequate sound $\mathcal{T}_\sigma$-interpretation. This is done by taking the quotient of $\Lambda(C)$ by a suitable notion of β-conversion (see Proposition 7.6.53 and Proposition 7.6.54). This point is rather delicate, and it justify the introduction of the new notion of combinatory σ-algebra, in place of the notion of λ-model of [AO93]. We generalize and clarify the argument of [AO93], which appears to be incomplete.

First of all we introduce the notion of *applicative σ-structure with convergence*:

Definition 7.6.36 (Applicative σ-structure with Convergence) *An* applicative σ-structure with convergence *is an applicative structure $\mathcal{A} = (A, \bullet_\mathcal{A}, \Downarrow_\mathcal{A})$ endowed with a notion of applicative equivalence $\approx^{app}_\mathcal{A} \subseteq A \times A$ such that*

- $\mathcal{V}_\mathcal{A} = \{a \in A \mid a \Downarrow_\mathcal{A}\} \neq \emptyset, A$*;*
- *if the application between λ-terms is right (left) strict w.r.t. $\Downarrow_\sigma$, then also $\bullet_\mathcal{A}$ is right (left) strict w.r.t. $\Downarrow_\mathcal{A}$;*

- *te functions δ^l, δ^r are defined on applicative structures with convergence as follows: for all $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}})$,*

$$\delta^l(\mathcal{A}) = \begin{cases} \mathcal{V}_{\mathcal{A}} & \textit{if } \bullet_{\mathcal{A}} \textit{ is left strict} \\ A & \textit{otherwise} \end{cases}$$

$$\delta^r(\mathcal{A}) = \begin{cases} \mathcal{V}_{\mathcal{A}} & \textit{if } \bullet_{\mathcal{A}} \textit{ is right strict} \\ A & \textit{otherwise} \end{cases}$$

- $a \approx^{app}_{\mathcal{A}} b \iff$
  $\forall k \geq 0\ \forall c_1, \ldots, c_k \in \delta^r(A).\ (ac_1 \ldots c_k \Downarrow_{\mathcal{A}} \Leftrightarrow bc_1 \ldots c_k \Downarrow_{\mathcal{A}})$.

There is a natural way to see the λ-model D^σ as an applicative σ-structure with convergence:

Proposition 7.6.37 *Let D^σ be a computationally adequate algebraic lattice model. Then D^σ can be viewed as an applicative σ-structure whose notion of convergence is:*

$$a \Downarrow_{D^\sigma} \iff \exists \phi \in T^{conv}_\sigma.\ a \sqsupseteq_{D^\sigma} \phi\ .$$

We can define various notions of interpretations for the types in $\mathcal{T}_\sigma$ on an applicative σ-structure with convergence. In the following definition, we isolate the class of type interpretations we are interested in, i.e. the *adequate* $\mathcal{T}_\sigma$-interpretations. This definition is crucial and does not appear in the original presentation of the domain logic method of [AO93].

Definition 7.6.38 *Let $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}})$ be an applicative σ-structure with convergence. An interpretation of the type theory $\mathcal{T}_\sigma$, $[\![\ ]\!]^{t\mathcal{A}} : T_\sigma \to \mathcal{P}(A)$, is* adequate *on $\mathcal{A}$, if $[\![\ ]\!]^{t\mathcal{A}}$ satisfies the following properties:*

- $[\![\phi \wedge \psi]\!]^{t\mathcal{A}} = [\![\phi]\!]^{t\mathcal{A}} \cap [\![\psi]\!]^{t\mathcal{A}}$

  $[\![\phi \to \psi]\!]^{t\mathcal{A}} = \{a \in \delta^l(A) \mid \forall c \in [\![\phi]\!]^{t\mathcal{A}} \cap \delta^r(A).\ a \bullet_{\mathcal{A}} c \in [\![\psi]\!]^{t\mathcal{A}}\}$.
- $\phi \leq_\sigma \psi \;\Rightarrow\; [\![\phi]\!]^{t\mathcal{A}} \subseteq [\![\psi]\!]^{t\mathcal{A}}$
- $a \Downarrow_{\mathcal{A}} \iff \exists \phi \in T^{conv}_\sigma.\ a \in [\![\phi]\!]^{t\mathcal{A}}$.

An adequate $\mathcal{T}_\sigma$-interpretation can be immediately defined on the model D^σ:

Proposition 7.6.39 *Let D^σ be a computationally adequate algebraic lattice model. The type interpretation $[\![\ ]\!]^{tD^\sigma}$ on D^σ, defined as,*

$$a \in [\![\phi]\!]^{tD^\sigma} \iff a \sqsupseteq_{D^\sigma} \phi\ ,$$

is adequate.

An intersection type interpretation on an applicative structure naturally induces a *logical equivalence*:

Definition 7.6.40 *Let* $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{t\mathcal{A}})$ *be an applicative* σ*-structure with type interpretation, the* logical equivalence $\approx^{\mathcal{L}}_{\mathcal{A}} \subseteq A \times A$ *is defined as follows*

$$a \approx^{\mathcal{L}}_{\mathcal{A}} b \iff \forall \phi.\ a \in [\![\phi]\!]^{t\mathcal{A}} \Leftrightarrow b \in [\![\phi]\!]^{t\mathcal{A}}\ .$$

Now, for technical reasons, we need to introduce the following notion of applicative σ-substructure:

Definition 7.6.41 *Let* $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}})$ *be an applicative* σ*-structure. An* applicative σ-substructure with convergence *of* $\mathcal{A}$ *is an applicative* σ*-structure* $\mathcal{A}' = (A', \bullet_{\mathcal{A}'}, \Downarrow_{\mathcal{A}'})$*, such that*

- $A' \subseteq A$,
- $\bullet_{\mathcal{A}'} = \bullet_{\mathcal{A}|A' \times A'}$*, and*
- $\Downarrow_{\mathcal{A}'} = \Downarrow_{\mathcal{A}|A'}$.

Now we discuss conditions under which the applicative equivalence on an applicative structure $\mathcal{A}$ with $\mathcal{T}_\sigma$-type interpretation coincides with the logical equivalence. In order to make the inclusion $\approx^{app}_{\mathcal{A}} \subseteq \approx^{\mathcal{L}}_{\mathcal{A}}$ hold, it is sufficient to check that it holds on base types. The other inclusion is more problematic, and we need to assume a further property on the applicative σ-structure, i.e. *approximability.*

Theorem 7.6.42 *Let* $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{t\mathcal{A}})$ *be an applicative* σ*-structure with adequate type interpretation. Let* $\mathcal{A}' = (A', \bullet_{\mathcal{A}'}, \Downarrow_{\mathcal{A}'})$ *be an applicative* σ*-substructure of* $\mathcal{A}$ *such that* $\approx^{app}_{\mathcal{A}'} = \approx^{app}_{\mathcal{A}\ |A' \times A'}$*. If, for all* $\phi \in TB_\sigma$*, and for all* $a, b \in A$*,* $a \approx^{app}_{\mathcal{A}} b \ \Rightarrow\ (a \in [\![\phi]\!]^{t\mathcal{A}} \Leftrightarrow b \in [\![\phi]\!]^{t\mathcal{A}})$*, then*

$$\approx^{app}_{\mathcal{A}'} \subseteq \approx^{\mathcal{L}}_{\mathcal{A}'}\ .$$

Now we introduce the notion of *approximable* applicative σ-structure. The approximability condition allows to decide *finitarily* the convergence property.

Definition 7.6.43 *Let* $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{t\mathcal{A}})$ *be an applicative* σ*-structure with type interpretation, and* $\mathcal{A}' = (A', \bullet_{\mathcal{A}'}, \Downarrow_{\mathcal{A}'})$ *be an applicative* σ*-substructure.* $\mathcal{A}'$ *is* approximable *if, for all* $a, b_1, \ldots, b_n \in A'$*, if* $ab_1 \ldots b_n \Downarrow_{\mathcal{A}'}$*, then* $\exists \phi_1, \ldots, \phi_n \in T_\sigma\ \exists \phi \in T^{conv}_\sigma.\ (a \in [\![\phi_1 \to \ldots \phi_n \to \phi]\!]^{t\mathcal{A}} \ \wedge\ \forall i.\ b_i \in [\![\phi_i]\!]^{t\mathcal{A}} \cap \delta^r(\mathcal{A}')$.

Proposition 7.6.44 *Let* D^σ *be a computationally adequate algebraic lattice model. Then* D^σ *with the type interpretation* $[\![\ ]\!]^{tD^\sigma}$ *of Proposition 7.6.39 is approximable.*

Theorem 7.6.45 *Let* $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{t\mathcal{A}})$ *be an applicative* σ*-structure with adequate type interpretation. Let* $\mathcal{A}' = (A', \bullet_{\mathcal{A}'}, \Downarrow_{\mathcal{A}'})$ *be an approximable applicative* σ*-substructure of* $\mathcal{A}$*. Then* $\approx^{app}_{\mathcal{A}'} \supseteq \approx^{\mathcal{L}}_{\mathcal{A}'}$.

Moreover, if $\mathcal{A}$ *and* $\mathcal{A}'$ *are such that*

- $\approx^{app}_{\mathcal{A}'} = \approx^{app}_{\mathcal{A}}{}_{|A' \times A'}$ *and,*

- $\forall \phi \in TB_\sigma, \forall a, b \in A,\ a \approx^{app}_{\mathcal{A}} b \ \Rightarrow\ (a \in [\![\phi]\!]^{t\mathcal{A}} \Leftrightarrow b \in [\![\phi]\!]^{t\mathcal{A}})$,

then $\approx^{app}_{\mathcal{A}'} = \approx^{\mathcal{L}}_{\mathcal{A}'}$.

The applicative equivalence on approximable applicative σ-structures is a congruence w.r.t. application, more precisely:

Theorem 7.6.46 *Let $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{t\mathcal{A}})$ be an applicative σ-structure with adequate type interpretation. Let $\mathcal{A}' = (A', \bullet_{\mathcal{A}'}, \Downarrow_{\mathcal{A}'})$ be an approximable applicative σ-substructure of $\mathcal{A}$. If $\mathcal{A}$ and $\mathcal{A}'$ are such that*

- $\approx^{app}_{\mathcal{A}'} = \approx^{app}_{\mathcal{A}}{}_{|A' \times A'}$, *and*

- *for all $\phi \in TB_\sigma$, for all $a, b \in A$, $a \approx^{app}_{\mathcal{A}} b \ \Rightarrow\ (a \in [\![\phi]\!]^{\mathcal{A}} \Leftrightarrow b \in [\![\phi]\!]^{\mathcal{A}})$,*

then $\approx^{app}_{\mathcal{A}'}$ is a congruence w.r.t. $\bullet_{\mathcal{A}'}$.

Proof It is sufficient to show that, for all $k \geq 0$, for all $d_1, \ldots, d_k \in A'$, $abd_1 \ldots d_k \Downarrow_{\mathcal{A}'} \Rightarrow\ acd_1 \ldots d_k \Downarrow_{\mathcal{A}'}$.
From $abd_1 \ldots d_k \Downarrow_{\mathcal{A}'}$, by approximability of $\mathcal{A}'$, there exist $\phi_0, \ldots, \phi_k \in T_\sigma$, $\phi \in T^{conv}_\sigma$ s.t. $a \in [\![\phi_0 \to \ldots \phi_k \to \phi]\!]^{t\mathcal{A}} \ \wedge\ b \in [\![\phi_0]\!]^{t\mathcal{A}}$ and $\forall i = 1, \ldots, k.\ b_i \in [\![\phi_i]\!]^{t\mathcal{A}}$. By Theorem 7.6.45, $c \in [\![\phi_0]\!]^{t\mathcal{A}}$, hence $acd_1 \ldots d_k \in [\![\phi]\!]^{t\mathcal{A}}$, and $acd_1 \ldots d_k \Downarrow_{\mathcal{A}'}$.
□

Now we introduce the new crucial notion of *combinatory σ-algebra*, which generalizes the standard notion of combinatory λ-algebra (see e.g. [Bar84]).

Definition 7.6.47 (Combinatory σ-algebra) *A combinatory σ-algebra $\mathcal{A}$ is a structure $(A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{\mathcal{A}})$ such that*

- *$(A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}})$ is an applicative σ-structure with convergence;*

- *$[\![\ ]\!]^{\mathcal{A}} : \Lambda(C) \times Env \to A$ is an interpretation function such that, for*

$$Env \ni \rho : Var \to \delta^r(\mathcal{A}) \ ,$$

  1. *$[\![x]\!]^{\mathcal{A}}_\rho = \rho(x)$*
  2. *$[\![MN]\!]^{\mathcal{A}}_\rho = [\![M]\!]^{\mathcal{A}}_\rho \bullet_{\mathcal{A}} [\![N]\!]^{\mathcal{A}}_\rho$*
  3. *$[\![\lambda x.M]\!]^{\mathcal{A}}_\rho \bullet_{\mathcal{A}} a = [\![M]\!]^{\mathcal{A}}_{\rho[a/x]}$, for all $a \in \delta^r(\mathcal{A})$*
  4. *$\forall M \in \Lambda^0(C).\ (M \Downarrow_\sigma \ \Rightarrow\ [\![M]\!]^{\mathcal{A}} \Downarrow_{\mathcal{A}})$.*

A combinatory σ-algebra $\mathcal{A}$ is adequate *w.r.t. D^σ, if* $\forall M \in \Lambda^0(C).\ ([\![M]\!]^{\mathcal{A}} \Downarrow_{\mathcal{A}} \Rightarrow\ [\![M]\!]^{D^\sigma} \Downarrow_{D^\sigma})$.

The kind of combinatory σ-algebras we are interested in are those which arise from applicative structures with adequate type interpretations. For these class of combinatory σ-algebras we formulate the *soundness* property w.r.t. the type theory $\mathcal{T}_\sigma$.

Theorem 7.6.48 (Soundness) *Let $\mathcal{A}$ be a combinatory σ-algebra with adequate $\mathcal{T}_\sigma$-interpretation. If, for all axioms ax in SB_σ,*

$$\forall \Gamma, M, \phi, \rho.\ \Gamma \vdash_\sigma M : \phi \ \ ax \quad \Longrightarrow \ \forall \rho\ (\forall x. \rho(x) \in [\![\Gamma(x)]\!]^{t\mathcal{A}} \ \Rightarrow \ [\![M]\!]^{\mathcal{A}}_{\rho} \in [\![\phi]\!]^{t\mathcal{A}})\ ,$$

then,

$$\forall \Gamma, M, \phi, \rho.\ \Gamma \vdash_\sigma M : \phi \ \Longrightarrow \ \forall \rho\ (\forall x. \rho(x) \in [\![\Gamma(x)]\!]^{t\mathcal{A}} \ \Rightarrow \ [\![M]\!]^{\mathcal{A}}_{\rho} \in [\![\phi]\!]^{t\mathcal{A}})\ .$$

Proof The proof is by induction on the length of the derivation $\Gamma \vdash_\sigma M : \phi$. We consider the case $\bullet_{\mathcal{A}}$ not right strict (the other case is similar).
Base Case (axioms in SB_σ): immediate.
Inductive Step: If the last rule in the derivation of $\Gamma \vdash_\sigma M : \phi$ is $\wedge I$, the thesis follows immediately from the induction hypothesis. If the last rule is $\leq$, then the thesis follows from the induction hypothesis and Definition 7.6.38. If the last rule is *var*, then the thesis follows from the definition of the interpretation of variables in a combinatory σ-algebra.
If the last rule is $\rightarrow I$, i.e. $\dfrac{\Gamma[\phi/x] \vdash_\sigma M : \psi}{\Gamma \vdash_\sigma \lambda x.M : \phi \rightarrow \psi}$, then, by induction hypothesis, $\forall \rho\ (\forall y. \rho(y) \in [\![\Gamma[\phi/x](y)]\!]^{t\mathcal{A}} \ \Rightarrow \ [\![M]\!]^{\mathcal{A}}_{\rho} \in [\![\psi]\!]^{t\mathcal{A}})$. We have to show that $\forall \rho\ (\forall y. \rho(y) \in [\![\Gamma(y)]\!]^{t\mathcal{A}} \ \Rightarrow \ [\![\lambda x.M]\!]^{\mathcal{A}}_{\rho} \in [\![\phi \rightarrow \psi]\!]^{t\mathcal{A}})$, i.e., if ρ is such that $\forall y.\ \rho(y) \in [\![\Gamma(y)]\!]^{t\mathcal{A}}$, then $\forall a \in [\![\phi]\!]^{t\mathcal{A}}.\ [\![\lambda x.M]\!]^{\mathcal{A}}_{\rho} \bullet_{\mathcal{A}} a \in [\![\psi]\!]^{t\mathcal{A}}$. But, from the definition of term interpretation in a combinatory σ-algebra, we have $[\![\lambda x.M]\!]^{\mathcal{A}}_{\rho} \bullet_{\mathcal{A}} a = [\![M]\!]^{\mathcal{A}}_{\rho[a/x]}$, and moreover $\forall y.\ \rho[a/x](y) \in [\![\Gamma[\phi/x](y)]\!]^{t\mathcal{A}}$. Hence, using the induction hypothesis, we get the thesis.
Finally, if the last rule is $\rightarrow E$, i.e. $\dfrac{\Gamma \vdash_\sigma M : \phi \rightarrow \psi \quad \Gamma \vdash_\sigma N : \phi}{\Gamma \vdash_\sigma MN : \psi}$, then, by induction hypothesis, $\forall \rho\ (\forall y. \rho(y) \in [\![\Gamma(y)]\!]^{t\mathcal{A}} \ \Rightarrow \ ([\![M]\!]^{\mathcal{A}}_{\rho} \in [\![\phi \rightarrow \psi]\!]^{t\mathcal{A}} \wedge [\![N]\!]^{\mathcal{A}}_{\rho} \in [\![\phi]\!]^{t\mathcal{A}}))$. Hence $[\![M]\!]^{\mathcal{A}}_{\rho} \bullet_{\mathcal{A}} [\![N]\!]^{\mathcal{A}}_{\rho} \in [\![\psi]\!]^{t\mathcal{A}}$. $\square$

It is immediate to see that the model D^σ is sound and complete w.r.t. the type theory $\mathcal{T}_\sigma$:

Theorem 7.6.49 (Soundness and Completeness of D^σ) *Let D^σ be a computationally adequate algebraic lattice model. Then,*

$$\forall \Gamma, M, \phi, \rho.\ \Gamma \vdash_\sigma M : \phi \iff \forall \rho\ (\forall x.\ \rho(x) \in [\![\Gamma(x)]\!]^{tD^\sigma} \ \Rightarrow \ [\![M]\!]^{D^\sigma}_{\rho} \in [\![\phi]\!]^{tD^\sigma})\ .$$

Notice that it is not always the case that an algebraic lattice model D^σ with term interpretation $[\![\ ]\!]^{D^\sigma}$ is a combinatory σ-algebra. If this is the case, then the type assignment system $\mathcal{T}_\sigma$ is sound and complete w.r.t. the class of combinatory σ-algebras with adequate type interpretation, which are sound. This is the generalization of the *Completeness* Theorem of [BCD83].

Another product of the domain logic method is the *computational adequacy* of $D^{|sigma}$ w.r.t. a class of combinatory σ-algebras.

Definition 7.6.50 *Let $\mathcal{A}$ be a combinatory σ-algebra. D^σ is* computationally adequate w.r.t. $\mathcal{A}$ *if,*

$$\forall M \in \Lambda^0(C).\ [\![M]\!]^{\mathcal{A}} \Downarrow_{\mathcal{A}} \iff [\![M]\!]^{D^\sigma} \Downarrow_{D^\sigma} \ .$$

The following theorem is the crucial result of the method. Item ii) of the following theorem generalizes Proposition 7.2.4 of [AO93].

Theorem 7.6.51 *Let D^σ be a computationally adequate algebraic lattice model which is a combinatory σ-algebra. Let $\mathcal{A} = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{\mathcal{A}})$ be a combinatory σ-algebra with an adequate $\mathcal{T}_\sigma$-interpretation $[\![\ ]\!]^{t\mathcal{A}}$ such that*

1. *the Soundness Theorem holds for $\mathcal{A}$, and*
2. *$\mathcal{A}$ is adequate w.r.t. the filter model D^σ.*

Then

i) *D^σ is computationally adequate w.r.t. $\mathcal{A}$;*

ii) *let $\mathcal{A}^0 = (A^0, \bullet_{\mathcal{A}^0}, \Downarrow_{\mathcal{A}^0})$ be the applicative σ-substructure of $\mathcal{A}$, where $A^0 \subseteq A$ denotes the interpretation domain of closed λ-terms. Then $\mathcal{A}^0$ is approximable.*

Proof
i) Use Theorem 7.6.49.
ii) We carry out the proof in the case $\bullet_{\mathcal{A}}$ is not right strict (the other case is similar). We show that, for all M,N_1,...,$N_k \in \Lambda^0(C)$,
$[\![MN_1 \dots N_k]\!]^{\mathcal{A}} \Downarrow_{\mathcal{A}^0} \implies \exists\phi_1, \dots, \phi_k \in T_\sigma, \exists\phi \in T_\sigma^{conv}.$
$([\![M]\!]^{\mathcal{A}} \in [\![\phi_1 \to \dots \phi_k \to \phi]\!]^{t\mathcal{A}} \wedge \forall i.\ [\![N_i]\!]^{\mathcal{A}} \in [\![\phi_i]\!]^{t\mathcal{A}})$. From $[\![MN_1 \dots N_k]\!]^{\mathcal{A}} \Downarrow_{\mathcal{A}^0}$, since $\mathcal{A}$ is adequate w.r.t. D^σ, it follows that $[\![MN_1 \dots N_k]\!]^{D^\sigma} \Downarrow_{D^\sigma}$. By approximability of D^σ, $\exists\phi_1, \dots, \phi_k \in T_\sigma, \exists\phi \in T_\sigma^{conv}$ such that
$[\![M]\!]^{D^\sigma} \in [\![\phi_1 \to \dots \phi_k \to \phi]\!]^{tD^\sigma} \ \wedge\ \forall i = 1, \dots, k.\ [\![N_i]\!]^{D^\sigma} \in [\![\phi_i]\!]^{tD^\sigma}$.
From the completeness of D^σ, $\exists\phi_1, \dots, \phi_k \in T_\sigma, \exists\phi \in T_\sigma^{conv}$ such that
$\vdash_\sigma M : \phi \to \dots \phi_k \to \phi \ \wedge\ \forall i = 1, \dots, k.\ \vdash_\sigma N_i : \phi_i$.
By soundness of $\mathcal{A}$, $[\![M]\!]^{\mathcal{A}} \in [\![\phi_1 \to \dots \phi_k \to \phi]\!]^{t\mathcal{A}} \ \wedge\ \forall i = 1, \dots, k.\ [\![N_i]\!]^{\mathcal{A}} \in [\![\phi_i]\!]^{t\mathcal{A}}$. □

Now we are in the position of summarizing the crucial result of this section:

Proposition 7.6.52 *Let $\approx_\sigma$ be a λ-congruence. If*

- *$\approx_\sigma$ has a a computationally adequate algebraic lattice model D^σ which is a combinatory σ-algebra;*
- *$\mathcal{A}_\sigma = (A, \bullet_{\mathcal{A}}, \Downarrow_{\mathcal{A}}, [\![\ ]\!]^{\mathcal{A}})$ is a combinatory σ-algebra with an adequate $\mathcal{T}_\sigma$-interpretation $[\![\ ]\!]^{t\mathcal{A}}$ such that*

1. the Soundness Theorem holds for $\mathcal{A}$, and

2. $\mathcal{A}$ is adequate w.r.t. the filter model D^σ;

- *The applicative σ-substructure of the interpretation domain of closed λ-terms, $\mathcal{A}^0_\sigma = (A^0, \bullet_{\mathcal{A}^0}, \Downarrow_{\mathcal{A}^0})$ is such that $\approx^{app}_{\mathcal{A}^0_\sigma} = (\approx^{app}_{\mathcal{A}_\sigma})_{|A^0 \times A^0}$;*

then $\approx^{app}_{\mathcal{A}^0_\sigma}$ is a congruence w.r.t. application.

In order to prove that $\approx^{app}_\sigma$ is a congruence w.r.t. application using Proposition 7.6.52 above, we have to endow the set of λ-terms with a combinatory algebra structure. This is not immediate, since the applicative structure of λ-terms with term application does not satisfy condition 3 of Definition 7.6.47. In order to make this condition hold, we have to quotient the set of λ-terms with a suitable notion of β-conversion. Let β_r be the notion of β-reduction correct w.r.t. $\approx_\sigma$ as in Definition 7.1.7, i.e. full β-reduction, if $\approx_\sigma$ is not right strict, (closed) value-restricted β-reduction, if $\approx_\sigma$ is right strict. Hence, in order to prove that $\approx^{app}_\sigma$ is a congruence w.r.t. application, we endow the set $\Lambda(C)/=_{\beta_r}$ (or $\Lambda^0(C)/=_{\beta_r}$) with a structure of combinatory σ-algebra with convergence as follows:

Proposition 7.6.53 *Let $\mathcal{A}_\sigma = (\Lambda(C)/=_{\beta_r}, \bullet_{=_{\beta_r}}, \Downarrow_\sigma, [\![\ ]\!]^{\mathcal{A}_\sigma})$ be the combinatory σ-algebra with convergence, where:*

- *by abuse of notation, $\Downarrow_\sigma$ denotes convergence of $=_{\beta_r}$-classes;*
- *$[\![\ ]\!]^{\mathcal{A}_\sigma} : \Lambda(C) \times Env \to \Lambda(C)/=_{\beta_r}$;*
- *$Env \ni \rho : Var \to \delta^r(\mathcal{A}_\sigma)$;*
- *$[\![M]\!]^{\mathcal{A}_\sigma}_\rho = [\rho(M)]_{=_{\beta_r}}$, where*
  $\rho(M) = M[\rho(x_1)/x_1, \ldots, \rho(x_n)/x_n]$, with $FV(M) = \{x_1, \ldots, x_n\}$.

Proposition 7.6.54 *Let $\approx_\sigma$ be a λ-congruence. If*

- *$\approx_\sigma$ has a computationally adequate algebraic lattice model D^σ which is a combinatory σ-algebra,*
- *the combinatory σ-algebra $\mathcal{A}_\sigma = (\Lambda(C)/=_{\beta_r}, \bullet_{=_{\beta_r}}, \Downarrow_\sigma, [\![\ ]\!]^{\mathcal{A}_\sigma})$ of Proposition 7.6.53 can be endowed with an adequate $\mathcal{T}_\sigma$-interpretation such that the Soundness Theorem 7.6.48 holds, and*
- *the applicative σ-substructure of $\mathcal{A}_\sigma$, $\mathcal{A}^0_\sigma = (\Lambda^0(C)/=_{\beta_r}, \bullet_{=_{\beta_r}}, \Downarrow_\sigma)$, is such that $\approx^{app}_{\mathcal{A}^0_\sigma} = \approx^{app}_{\mathcal{A}_\sigma|(\Lambda^0(C)/=_{\beta_r} \times \Lambda^0(C)/=_{\beta_r})}$,*

then the applicative equivalence on $\mathcal{A}^0_\sigma$, $\approx^{app}_{\mathcal{A}^0_\sigma}$, is a congruence w.r.t. application. Hence $\approx^{app}_\sigma$ is a congruence w.r.t. application.

The Domain Logic Method at Work

Case $\sigma = l$. The computationally adequate model D^l of Section 7.4 is, by definition, a combinatory l-algebra (see [AO93]). In this case we can consider directly the combinatory l-algebra on closed terms, $\mathcal{A}_l^0$, and we define on it an adequate $\mathcal{T}_l$-interpretation.

Definition 7.6.55 *We define on the combinatory l-algebra $\mathcal{A}_l^0 = (\Lambda^0/=_\beta, \bullet_{=_\beta}, \Downarrow_l, [\![\ ]\!]^{\mathcal{A}_l^0})$ the type interpretation $[\![\ ]\!]^{t\mathcal{A}_l^0}$ as follows:*

$[\![\omega]\!]^{t\mathcal{A}_l^0} = \Lambda^0/=_\beta$

$[\![\phi \wedge \psi]\!]^{t\mathcal{A}_l^0} = [\![\phi]\!]^{t\mathcal{A}_l^0} \cap [\![\psi]\!]^{t\mathcal{A}_l^0}$

$[\![\phi \to \psi]\!]^{t\mathcal{A}_l^0} = \{a \in \Lambda^0/=_\beta \mid a \Downarrow_l \ \wedge \forall c \in [\![\phi]\!]^{t\mathcal{A}_l^0}.\ a \bullet_{\mathcal{A}_l^0} c \in [\![\psi]\!]^{t\mathcal{A}_l^0}\}.$

Theorem 7.6.56 *The notion of type interpretation $[\![\ ]\!]^{t\mathcal{A}_l^0}$ is adequate. Moreover, the combinatory l-algebra $\mathcal{A}_l^0$ with the type interpretation $[\![\ ]\!]^{t\mathcal{A}_l^0}$ is sound w.r.t. the type assignment system S_l.*

Proof In order to prove that the type interpretation $[\![\ ]\!]^{t\mathcal{A}_l^0}$ is adequate, we have to show only that:

1. $\phi \leq_l \psi \ \Rightarrow\ [\![\phi]\!]^{t\mathcal{A}_l^0} \subseteq [\![\psi]\!]^{t\mathcal{A}_l^0}$
2. $a \Downarrow_l \iff a \in [\![\omega \to \omega]\!]^{t\mathcal{A}_l^0}$.

Item 1 is easily proved by induction on the derivation of $\phi \leq_l \psi$. Item 2 follows immediately from the definition of $[\![\ ]\!]^{t\mathcal{A}_l^0}$.

The fact that the type interpretation $[\![\ ]\!]^{t\mathcal{A}_l^0}$ is sound w.r.t. the type assignment system S_l is immediate. □

Notice that the notion of type interpretation defined above can be given uniformly on all applicative l-structure with convergence, and, on D^l, it coincides with the natural notion of type interpretation of Proposition 7.6.39. Thus the type assignment system $\mathcal{T}_l$ is sound and complete w.r.t. a large class of structures (cfr. [BCD83]).

Case $\sigma = v$. The model D^v of Section 7.4 is by definition a combinatory v-algebra (see [EHR92]). Also in this case, as for $\sigma = l$, we consider directly the combinatory algebra on closed terms.

Definition 7.6.57 *We define on the combinatory v-algebra $\mathcal{A}_v^0 = (\Lambda^0/=_{\beta_v}, \bullet_{=_{\beta_v}}, \Downarrow_v, [\![\ ]\!]^{\mathcal{A}_v^0})$ the type interpretation $[\![\ ]\!]^{t\mathcal{A}_v^0}$ as follows:*

$[\![\nu]\!]^{t\mathcal{A}_v^0} = \{a \in \Lambda^0/=_{\beta_v} \mid a \Downarrow_{\mathcal{A}_v^0}\}$

$[\![\phi \wedge \psi]\!]^{t\mathcal{A}_v^0} = [\![\phi]\!]^{t\mathcal{A}_v^0} \cap [\![\psi]\!]^{t\mathcal{A}_v^0}$

$[\![\phi \to \psi]\!]^{t\mathcal{A}_v^0} = \{a \in \Lambda^0/=_{\beta_v} \mid a \Downarrow_v \ \wedge \forall c \in [\![\phi]\!]^{t\mathcal{A}_v^0}.\ a \bullet_{\mathcal{A}_v^0} c \in [\![\psi]\!]^{t\mathcal{A}_v^0}\}.$

Theorem 7.6.58 *The notion of type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_v}$ *is adequate. Moreover, the combinatory v-algebra* $\mathcal{A}^0_v$ *with the type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_v}$ *is sound w.r.t. the type assignment system* S_v*.*

Notice that, as for the case $\sigma = l$, the notion of type interpretation of Definition 7.6.57 can be given uniformly on all applicative v-structure with convergence, and, on D^v, it coincides with the natural notion of type interpretation of Proposition 7.6.39. Thus, also in this case, the type assignment system $\mathcal{T}_v$ is sound and complete w.r.t. a large class of structures (cfr. [BCD83]).

Case $\sigma = o$**.** The model D^o of Section 7.4 is by definition a combinatory o-algebra (see [HR92]). Also in this case, as for $\sigma = l, v$, we consider directly the combinatory algebra on closed terms.

Definition 7.6.59 *We define on the combinatory o-algebra* $\mathcal{A}^0_o = (\Lambda^0(\{\Omega\})/=_\beta, \bullet_{=_\beta}, \Downarrow_o, [\![\ ]\!]^{\mathcal{A}^0_o})$ *the type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_o}$ *as follows:*

$[\![\omega]\!]^{t\mathcal{A}^0_o} = (\Lambda^0(\{\Omega\}))/=_\beta$

$[\![\nu]\!]^{t\mathcal{A}^0_o} = \{[M]_{=_\beta} \in \Lambda^0(\{\Omega\})/=_\beta \mid [M]_{=_\beta} \Downarrow_{\mathcal{A}^0_o}\}$

$[\![\phi \wedge \psi]\!]^{t\mathcal{A}^0_o} = [\![\phi]\!]^{t\mathcal{A}^0_o} \cap [\![\psi]\!]^{t\mathcal{A}^0_o}$

$[\![\phi \to \psi]\!]^{t\mathcal{A}^0_o} = \{[M]_{=_\beta} \in \Lambda^0(\{\Omega\})/=_\beta \mid \forall [N]_{=_\beta} \in [\![\phi]\!]^{t\mathcal{A}^0_o}.\ [MN]_{=_\beta} \in [\![\psi]\!]^{t\mathcal{A}^0_o}\}.$

Theorem 7.6.60 *The notion of type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_o}$ *is adequate. Moreover, the combinatory o-algebra* $\mathcal{A}^0_o$ *with the type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_o}$ *is sound w.r.t. the type assignment system* S_o*.*

Proof The only non trivial part is to show that the type interpretation $[\![\ ]\!]^{t\mathcal{A}^0_o}$ is adequate. In order to do this, we have to prove that:

1. $\phi \leq_o \psi \;\Rightarrow\; [\![\phi]\!]^{t\mathcal{A}^0_o} \subseteq [\![\psi]\!]^{t\mathcal{A}^0_o}$
2. $[M]_{=_\beta} \Downarrow_o \iff [M]_{=_\beta} \in [\![\nu]\!]^{t\mathcal{A}^0_o}$.

Item 2 follows immediately from the definition of $[\![\ ]\!]^{t\mathcal{A}^0_o}$.

The proof of item 1 is carried on by induction on the derivation of $\phi \leq_o \psi$. The only problematic case is the base cases $\nu \to \nu \leq_o \nu$. In order to show the inclusion $[\![\nu \to \nu]\!]^{t\mathcal{A}^0_o} \subseteq [\![\nu]\!]^{t\mathcal{A}^0_o}$, we prove that, for all $M \in \Lambda^0(\{\Omega\})$,

$$\forall N \in \Lambda^0(\{\Omega\}).\ (N \Downarrow_o \Rightarrow\ MN \Downarrow_o) \implies M \Downarrow_o\ .$$

This follows from $M(\Delta\Delta) \Downarrow_o \Rightarrow\ M \Downarrow_o$. □

Notice that the notion of type interpretation of Definition 7.6.61 above can be given uniformly on all applicative o-structures with convergence, but this notion of interpretation on D^o does not coincide with the natural notion of type interpretation of Proposition 7.6.39.

Case $\sigma = h$. In this case we cannot consider directly the combinatory h-algebra on closed λ-terms, since it is not possible to define a non trivial adequate type interpretation directly on closed terms. Hence we have two possibilities:

1. either we can enrich $\Lambda^0(C)$ with suitable constants that behave as the open terms to which can be assigned types not assignable to closed terms,

2. or we can simply consider the combinatory h-algebra on all $\Lambda(C)$.

In both cases we have to prove that the applicative equivalence $\approx_h^{app}$ coincides with the applicative equivalence on the combinatory algebra so defined. But this follows from Corollary 7.6.6.

We make the latter choice:

Definition 7.6.61 *We define on the combinatory h-algebra $\mathcal{A}_h = (\Lambda/=_\beta, \bullet_{=_\beta}, \Downarrow_h, [\![\ ]\!]^{\mathcal{A}_h})$ the type interpretation $[\![\ ]\!]^{t\mathcal{A}_h}$ as follows:*

$[\![\omega]\!]^{t\mathcal{A}_h} = \Lambda/=_\beta$

$[\![1]\!]^{t\mathcal{A}_h} = \{[M]_{=_\beta} \in \Lambda/=_\beta | \ \forall k \geq 0\ \forall [N_1]_{=_\beta}, \ldots, [N_k]_{=_\beta} \in \Lambda/=_\beta\ .\ [MN_1\ldots N_k]_{=_\beta} \Downarrow_{\mathcal{A}_h}\}$

$[\![\phi \wedge \psi]\!]^{t\mathcal{A}_h} = [\![\phi]\!]^{t\mathcal{A}_h} \cap [\![\psi]\!]^{t\mathcal{A}_h}$

$[\![\phi \to \psi]\!]^{t\mathcal{A}_h} = \{a \in \Lambda^0/=_\beta | \ a \Downarrow_h\ \wedge \forall c \in [\![\phi]\!]^{t\mathcal{A}_h}.\ a \bullet_{\mathcal{A}_h} c \in [\![\psi]\!]^{t\mathcal{A}_h})\}$.

Theorem 7.6.62 *The notion of type interpretation $[\![\ ]\!]^{t\mathcal{A}_h}$ is adequate. Moreover, the combinatory h-algebra $\mathcal{A}_h$ with the type interpretation $[\![\ ]\!]^{t\mathcal{A}_h}$ is sound w.r.t. the type assignment system S_h.*

Proof The only non trivial part is to show that the type interpretation $[\![\ ]\!]^{t\mathcal{A}_h}$ is adequate. In order to do this, we have to prove that:

1. $\phi \leq_h \psi\ \Rightarrow\ [\![\phi]\!]^{t\mathcal{A}_h} \subseteq [\![\psi]\!]^{t\mathcal{A}_h}$

2. $[M]_{=_\beta} \Downarrow_h \iff \exists \phi \in T_h^{conv}.\ [M]_{=_\beta} \in [\![\phi]\!]^{t\mathcal{A}_h}$

Item 1 is proved immediately by induction on the structure of the proof of $\phi \leq_h \psi$. The proof of item 2 ($\Leftarrow$) is trivial. In order to show the converse, we use the computational adequacy of D^h and the following fact: let $M, N_1, \ldots N_n \in \Lambda$ and $\{x_1, \ldots, x_n\} \supseteq FV(M)$, then

$$x_1 : \phi_1 \ldots x_n : \phi_n \vdash_h M : \phi \wedge \forall i.\ [N_i]_{=_\beta} \in [\![\phi_i]\!]^{t\mathcal{A}_h} \implies [M[N_i/x_i]]_{=_\beta} \in [\![\phi]\!]^{t\mathcal{A}_h}\ .$$

This fact is easily proved by induction on the derivation of $x_1 : \phi_1 \ldots x_n : \phi_n \vdash_h M : \phi$. □

Notice that the notion of type interpretation of Definition 7.6.61 above can be given uniformly on all applicative h-structures with convergence, but this notion of interpretation on D^h does not coincide with the natural notion of type interpretation of Proposition 7.6.39.

Case $\sigma = n$. Also for $\sigma = n$, as for $\sigma = h$, we consider the combinatory n-algebra $\mathcal{A}_n$ of all λ-terms, since it is not possible not to define an adequate type interpretation directly on closed terms.

Definition 7.6.63 *We define on the combinatory n-algebra $\mathcal{A}_n = (\Lambda/=_\beta, \bullet_{=_\beta}, \Downarrow_n, [\![\ ]\!]^{\mathcal{A}_n})$ the type interpretation $[\![\ ]\!]^{t\mathcal{A}_n}$ as follows:*

$[\![\omega]\!]^{t\mathcal{A}_n} = \Lambda/=_\beta$

$[\![0]\!]^{t\mathcal{A}_n} = \{[M]_{=_\beta} \in \Lambda/=_\beta \mid [M]_{=_\beta} \Downarrow_{\mathcal{A}_n}\}$

$[\![1]\!]^{t\mathcal{A}_n} = \{[M]_{=_\beta} \in \Lambda/=_\beta \mid \forall k \geq 0\ \forall [N_1]_{=_\beta}, \ldots, [N_k]_{=_\beta} \in \Lambda/=_\beta\ .$
$(\forall i.\ [N_i]_{=_\beta} \Downarrow_{\mathcal{A}_n}\ \Rightarrow\ [MN_1 \ldots N_k]_{=_\beta} \Downarrow_{\mathcal{A}_n})\}$

$[\![\phi \wedge \psi]\!]^{t\mathcal{A}_n} = [\![\phi]\!]^{t\mathcal{A}_n} \cap [\![\psi]\!]^{t\mathcal{A}_n}$

$[\![\phi \to \psi]\!]^{t\mathcal{A}_n} = \{[M]_{=_\beta} \in \Lambda/=_\beta \mid \forall [N]_{=_\beta} \in [\![\phi]\!]^{t\mathcal{A}_n}.\ [MN]_{=_\beta} \in [\![\psi]\!]^{t\mathcal{A}_n}\}.$

Theorem 7.6.64 *The notion of type interpretation $[\![\ ]\!]^{t\mathcal{A}_n}$ is adequate. Moreover, the combinatory n-algebra $\mathcal{A}_n$ with the type interpretation $[\![\ ]\!]^{t\mathcal{A}_n}$ is sound w.r.t. the type assignment system S_n.*

Proof The only non trivial part is to show that the type interpretation $[\![\ ]\!]^{t\mathcal{A}_n}$ is adequate. In order to do this, we have to prove that:

1. $\phi \leq_n \psi\ \Rightarrow\ [\![\phi]\!]^{t\mathcal{A}_n} \subseteq [\![\psi]\!]^{t\mathcal{A}_n}$
2. $[M]_{=_\beta} \Downarrow_n \iff [M]_{=_\beta} \in [\![0]\!]^{t\mathcal{A}_n}$.

Item 2 follows immediately from the definition of $[\![\ ]\!]^{t\mathcal{A}_n}$.

The proof of item 1 is carried out by induction on the derivation of $\phi \leq_n \psi$. The only difficult cases are the following base cases:
a) $0 = 1 \to 0$ and
b) $1 = 0 \to 1$.
First of all notice that, putting $\Gamma_1 = \{x : 1 \mid x \in FV(M)\}$, we have:
$[M]_{=_\beta} \in [\![0]\!]^{t\mathcal{A}_n}\ \Leftrightarrow\ \Gamma_1 \vdash_n M : 0$: proved using the definition of $[\![\ ]\!]^{t\mathcal{A}_n}$ and Theorem 7.4.12.
$[M]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n}\ \Leftrightarrow\ \Gamma_1 \vdash_n M : 1$: proved using the definition of $[\![\ ]\!]^{t\mathcal{A}_n}$ and Theorem 4.3 of [CDZ87].
Item a) amounts to showing that

$$M \Downarrow_n \iff \forall [N]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n}.\ MN \Downarrow_n \quad (***).$$

Proof of a($\Rightarrow$): The proof proceeds by induction on the structure of the normal form P to which M converges. The problematic case is that of $P \equiv \lambda x.Q$, with Q normal form. Let $N \in [\![1]\!]^{t\mathcal{A}_n}$. We have to prove that $Q[N/x] \Downarrow_n$. Since Q is a normal form, by Theorem 7.4.12, we have $\Gamma_1 \vdash_n Q : 0$. Hence $\Gamma_1 \vdash_n Q[N/x] : 0$, and applying Theorem 7.4.12 again, $Q[N/x] \Downarrow_n$.
Proof of a($\Leftarrow$): Suppose that $\exists [N]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n}.\ MN \Downarrow_n$, and moreover $[\![N]\!]^{tD^n}_{\rho_1} = 1$, where $\rho_1(x) = 1$, for all $x \in Var$. Consider $N \equiv x$, then clearly $[x]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n}$. We show, by induction on the derivation of $MN \Downarrow_n$, that $M \Downarrow_n$. The

only problematic case is that of $M \equiv \lambda x.M_1$. Since $(\lambda x.M_1)N \Downarrow_n$, by Theorem 7.4.12, we have $\Gamma_1 \vdash_n M_1[N/x] : 0$, and, since $[\![N]\!]^{tD^n}_{\rho_1} = 1$, $\Gamma_1 \vdash_n M_1 : 0$. Applying Theorem 7.4.12 again, we get $M_1 \Downarrow_n$.

Item b) amounts to showing that

$$[M]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n} \iff \forall N \Downarrow_n . [MN]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n} .$$

The implication ($\Rightarrow$) follows immediately from the Definition of $[\![\ ]\!]^{t\mathcal{A}_n}$. In order to show the converse, it is sufficient to prove that $\forall N \Downarrow_n . [MN]_{=_\beta} \in [\![1]\!]^{t\mathcal{A}_n} \Rightarrow M \Downarrow_n$. This follows from the fact that $[\![1]\!]^{t\mathcal{A}_n} \subseteq [\![0]\!]^{t\mathcal{A}_n}$, using implication ($\Leftarrow$) of fact $(***)$. $\square$

Notice that the notion of type interpretation of Definition 7.6.61 above can be given uniformly on all applicative n-structures with convergence, but this notion of interpretation on D^n does not coincide with the natural notion of type interpretation of Proposition 7.6.39.

Case $\sigma = p$. Also in this case, as for $\sigma = h, n$, it is not possible not to define a non trivial adequate type interpretation directly on closed terms. But for $\sigma = p$ we cannot consider the combinatory algebra of all λ-terms, since value-restricted β-reduction is not correct w.r.t. $\approx_p$ (and $\approx_p^{app}$) (see Section 7.3). So we are forced to make the choice of enriching the set of closed λ-terms with a new syntactic constant 1, whose intended meaning is that of an open term interpretable in the semantic constant 1. Therefore the syntactic constant 1 behave like a free variable:

Definition 7.6.65
- *Let $\Downarrow_{p,1} \subseteq \Lambda(\{1\}) \times Val_{p,1}$ be the notion of convergence axiomatized by adding the following rule to those for the $\rightarrow_p$ strategy defined in Section 7.3:*

  $$\frac{M_1 \Downarrow_{p,1} M_1' \ \ldots \ M_n \Downarrow_{p,1} M_n'}{1M_1 \ldots M_n \Downarrow_{p,1} 1M_1' \ldots M_n'} \quad n \geq 0$$

- *Let $\leq_{p,1} \subseteq \Lambda(\{1\}) \times \Lambda(\{1\})$ be defined by:*

  $$M \leq_{p,1} N \ \textit{ iff } \ \forall C[\,].(C[M], C[N] \in \Lambda^0(\{1\}) \Rightarrow (C[M] \Downarrow_{p,1} \Rightarrow C[N] \Downarrow_{p,1})) .$$

- *Let $\approx_{p,1}$ be $\leq_{p,1} \cap (\leq_{p,1})^{-1}$.*
- *Let $\leq_{p,1}^{app} \subseteq \Lambda(\{1\}) \times \Lambda(\{1\})$ be defined by: for all $M, N \in \Lambda(\{1\})$ such that $FV(M, N) \subseteq \{x_1, \ldots, x_k\}$,*

  $M \leq_{p,1}^{app} N$ *iff*
  $\forall Q_1, \ldots, Q_k, \forall P_1, \ldots, P_n \in \mathcal{V}_p$.
  $(M[Q_1/x_1, \ldots, q_k/x_k]P_1 \ldots P_n \Downarrow_{p,1} \Rightarrow N[Q_1/x_1, \ldots, q_k/x_k]P_1 \ldots P_n \Downarrow_{p,1})$.

- *Let* $\approx^{app}_{p,1}$ *be* $\leq^{app}_{p,1} \cap (\leq^{app}_{p,1})^{-1}$.

Lemma 7.6.66 *The notion of* β*-reduction* $\rightarrow_{\beta_{\Downarrow^0_{p,1}}} \subseteq \Lambda(\{1\}) \times \Lambda(\{1\})$*, defined by*

$$(\lambda x.M)N \rightarrow_{\beta_{\Downarrow^0_{p,1}}} M[N/x] \ , \ \text{if } N \in \{P \in \Lambda^0(\{1\}) \mid P \Downarrow_{p,1}\} \ ,$$

is correct w.r.t. $\approx_{p,1}$.

Then we can endow $\Lambda^0(\{1\})$ with a structure of combinatory p-algebra:

Definition 7.6.67 *We define on the combinatory p-algebra* $\mathcal{A}^0_{p,1} = (\Lambda/=_{\beta_{\Downarrow^0_{p,1}}}, \bullet_{=_{\beta_{\Downarrow^0_{p,1}}}}, \Downarrow_{p,1}, [\![\ ]\!]^{\mathcal{A}^0_{p,1}})$ *a type interpretation* $[\![\ ]\!]^{t\mathcal{A}_p}$ *as follows:*

$$[\![0]\!]^{t\mathcal{A}^0_{p,1}} = \{[M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in \Lambda/=_{\beta_{\Downarrow^0_{p,1}}} \mid [M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{\mathcal{A}^0_{p,1}}\}$$

$$[\![1]\!]^{t\mathcal{A}^0_{p,1}} = \{[M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in \Lambda/=_{\beta_{\Downarrow^0_{p,1}}} \mid \forall k \geq 0\ \forall [N_1]_{=_{\beta_{\Downarrow^0_{p,1}}}}, \ldots, [N_k]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in \Lambda/=_{\beta_{\Downarrow^0_{p,1}}}$$
$$(\forall i.\ [N_i]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{\mathcal{A}^0_{p,1}} \ \Rightarrow\ [MN_1 \ldots N_k]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{\mathcal{A}^0_{p,1}})\}$$

$$[\![\phi \wedge \psi]\!]^{t\mathcal{A}^0_{p,1}} = [\![\phi]\!]^{t\mathcal{A}^0_{p,1}} \cap [\![\psi]\!]^{t\mathcal{A}^0_{p,1}}$$

$$[\![\phi \rightarrow \psi]\!]^{t\mathcal{A}^0_{p,1}} = \{[M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in \Lambda/=_{\beta_{\Downarrow^0_{p,1}}} \mid [M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{p,1} \ \wedge$$
$$\forall [N]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in [\![\phi]\!]^{t\mathcal{A}^0_{p,1}}.\ ([N]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{p,1} \ \Rightarrow\ [MN]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in [\![\psi]\!]^{t\mathcal{A}^0_{p,1}})\}.$$

Theorem 7.6.68 *The notion of type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_{p,1}}$ *is adequate. Moreover, the combinatory p-algebra* $\mathcal{A}^0_{p,1}$ *with the type interpretation* $[\![\ ]\!]^{t\mathcal{A}^0_{p,1}}$ *is sound w.r.t. the type assignment system* S_p.

Proof The only non trivial part is to show that the type interpretation $[\![\ ]\!]^{t\mathcal{A}^0_{p,1}}$ is adequate. In order to do this, we have to prove that:

1. $\phi \leq_p \psi \ \Rightarrow\ [\![\phi]\!]^{t\mathcal{A}^0_{p,1}} \subseteq [\![\psi]\!]^{t\mathcal{A}^0_{p,1}}$
2. $[M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \Downarrow_{p,1} \iff [M]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in [\![0]\!]^{t\mathcal{A}^0_{p,1}}$

Item 2 follows immediately from the definition of $[\![\ ]\!]^{t\mathcal{A}^0_{p,1}}$.

The proof of item 1 is carried out by induction on the derivation of $\phi \leq_p \psi$. The only difficult case is the base case $0 \leq_p 1 \rightarrow 0$. This amounts to show that

$$M \Downarrow_{p,1} \implies \forall [N]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in [\![1]\!]^{t\mathcal{A}^0_{p,1}}.\ MN \Downarrow_{p,1} \ .$$

This can be proved by induction on the derivation of $M \Downarrow_{p,1}$ using the fact that

$$P \Downarrow_{p,1} \ \wedge [Q]_{=_{\beta_{\Downarrow^0_{p,1}}}} \in [\![1]\!]^{t\mathcal{A}^0_{p,1}} \implies P[Q/x] \Downarrow_{p,1} \ ,$$

which in turns is proved by induction on the derivation of $P \Downarrow_{p,1}$. $\square$

Also in this case, the notion of type interpretation of Definition 7.6.67 above can be given uniformly on all applicative p-structures with convergence, but this notion of interpretation on D^p does not coincide with the natural notion of type interpretation of Proposition 7.6.39.

Finally, we are left to show that $\approx_p^{app,1} = \approx_p^{app}$. But this is the analogue of Corollary 7.6.6.

7.7 Φ_σ^G-coinductive Characterizations

In this section, we illustrate a syntactical technique for showing that a λ-theory $\approx_\sigma$ satisfies the Φ_σ^G-coinduction principle of Section 7.2. This technique is inspired by the congruence candidate method used in Section 7.6.2 for showing that a λ-theory $\approx_\sigma$ is applicative, i.e. it satisfies the applicative coinduction principle of Theorem 7.2.4 of Section 7.2. The congruence candidate method presented in this section is used for showing that the equivalence $\approx_\sigma^G$ is a congruence w.r.t. application. In fact, if $\approx_\sigma^G$ is a congruence w.r.t. application, then $\approx_\sigma^G$ is a Φ_σ^F-bisimulation, and hence $\approx_\sigma^G = \approx_\sigma^{app}$. In this section we apply the congruence candidate method to lazy strategies. It is not clear how to extend this method to other strategies.

In the literature, there are also other techniques for proving Φ_σ^G-coinductive characterizations of λ-congruences (see [HL98]). Some of these techniques can be applied also to non-lazy strategies.

7.7.1 The Congruence Candidate Method

This method makes essential use of the coinduction principle of Theorem 7.2.8, and it is based on the definition of a *candidate relation*, which is a congruence w.r.t. application, and which extends $\approx_\sigma^G$. The aim is to show that the candidate relation is a Φ_σ^G-bisimulation; hence the coinduction principle of Theorem 7.2.8 guarantees that $\approx_\sigma^G$ itself is a congruence w.r.t. application. The pattern of this method is very similar to that of the congruence candidate method of Section 7.6.2 used for showing that $\approx_\sigma^{app}$ is a congruence w.r.t. application.

As for the method described in Section 7.6.2, we start by defining candidate relations. These are defined in terms of the extensions to open terms of Φ_σ^G-bisimulations. Here Definition 7.2.9 plays a crucial role.

Definition 7.7.1 (Candidate Relation) *Let $\eta \subseteq \Lambda \times \Lambda$ be a reflexive and transitive Ψ_σ-bisimulation. Define the* candidate relation $\widehat{\eta} \subseteq \Lambda \times \Lambda$ *by induction on M as follows:*

$$\frac{x\ \eta\ N}{x\ \widehat{\eta}\ N} \qquad \frac{M_1\ \widehat{\eta}\ M_1' \quad M_2\ \widehat{\eta}\ M_2' \quad M_1'M_2'\ \eta\ N}{M_1M_2\ \widehat{\eta}\ N} \qquad \frac{M\ \widehat{\eta}\ M' \quad \lambda x.M'\ \eta\ N}{\lambda x.M\ \widehat{\eta}\ N}$$

We can prove the analogues of Lemmata 7.6.9 and 7.6.10 of Section 7.6.2.

Lemma 7.7.2 *Let $\eta \subseteq \Lambda \times \Lambda$ be a reflexive and transitive Φ_σ^G-bisimulation. Then:*

i) *$\widehat{\eta}$ is reflexive.*

ii) *$\eta \subseteq \widehat{\eta}$.*

iii) *$\widehat{\eta}$ is a congruence w.r.t. application.*

iv) *$M\widehat{\eta}M' \ \wedge\ M'\eta N \implies M\widehat{\eta}N$.*

Lemma 7.7.3 (Substitutivity) *Let $\eta \subseteq \Lambda \times \Lambda$ be a reflexive and transitive Φ_σ^G-bisimulation. For all $M, M' \in \Lambda$, $N, N' \in \delta(\Lambda^0)$,*

$$M\widehat{\eta}M' \ \wedge\ N\widehat{\eta}N' \implies M[N/x]\widehat{\eta}M'[N'/x] \ .$$

Proof By induction on the structure of M.

- $M \equiv x$: $\quad \dfrac{x\ \eta\ M'}{x\ \widehat{\eta}\ M'}$

$x\eta M' \implies \exists P \in \delta(\Lambda^0).\ N'\eta P \ \wedge\ P\eta M'[N'/x]$, from the definition of η, and hence, by transitivity of η, $N'\eta M'[N'/x]$.
$N\widehat{\eta}N' \ \wedge\ N'\eta M'[N'/x] \implies N\widehat{\eta}M'[N'/x]$, from item iv of Lemma 7.7.2.

- $M \equiv M_1M_2$: $\exists M_1', M_2'$ s.t. $\quad \dfrac{M_1\ \widehat{\eta}\ M_1' \quad M_2\ \widehat{\eta}\ M_2' \quad M_1'M_2'\ \eta\ M'}{M_1M_2\ \widehat{\eta}\ M'}$

By definition of η, there exists $P \in \delta(\Lambda^0)$ such that
$N'\eta P$ and $M_1'M_2'[P/x]\eta M'[N'/x]$.
In particular, from $N\widehat{\eta}N'$ and $N'\eta P$, we get $N\widehat{\eta}P$. By induction hypothesis, $M_1[N/x]\widehat{\eta}M_1'[P/x]$ and $M_2[N/x]\widehat{\eta}M_2'[NP/x]$. Hence:

$$\frac{M_1[N/x]\ \widehat{\eta}\ M_1'[P/x] \quad M_2[N/x]\ \widehat{\eta}\ M_2'[P/x] \quad M_1'M_2'[P/x]\ \eta\ M'[N'/x]}{M_1M_2[N/x]\ \widehat{\eta}\ M'[N'/x]} \ .$$

- $M \equiv \lambda y.M_1$: $\exists M_1'$ s.t. $\quad \dfrac{M_1\ \widehat{\eta}\ M_1' \quad \lambda y.M_1'\ \eta\ M'}{\lambda y.M_1\ \widehat{\eta}\ M'}$

By definition of η, there exists $P \in \delta(\Lambda^0)$ such that $N'\eta P$ and $(\lambda y.M_1')[P/x]\eta M'[N'/x]$. In particular, from $N\widehat{\eta}N'$ and $N'\eta P$, we get $N\widehat{\eta}P$. By induction hypothesis, $M_1[N/x]\widehat{\eta}M_1'[P/x]$. Hence:

$$\frac{M_1[N/x]\ \widehat{\eta}\ M_1'[P/x] \quad (\lambda y.M_1')[P/x]\ \eta\ M'[N'/x]}{(\lambda y.M_1)[N/x]\ \widehat{\eta}\ M'[N'/x]} \ . \qquad \square$$

Thus, if we take η to be the equivalence $\approx_\sigma^G$, we get a relation $\widehat{\approx}_\sigma^G$, which, by item ii of Lemma 7.7.2, extends $\approx_\sigma^G$. Moreover, by item iii of the same lemma, it is a congruence w.r.t. application. In order to show that $\approx_\sigma^G$ is itself a congruence w.r.t. application, we prove that $(\widehat{\approx}_\sigma^G)_{|\Lambda^0\times\Lambda^0} = (\approx_\sigma^G)_{|\Lambda^0\times\Lambda^0}$. This is done using the coinduction principle of Theorem 7.2.8, by proving that

$(\widehat{\approx}^G_\sigma)_{|\Lambda^0\times\Lambda^0}$ is a Φ^G_σ-bisimulation. In order to prove that $(\widehat{\approx}^G_\sigma)_{|\Lambda^0\times\Lambda^0}$ is a Φ^G_σ-bisimulation, it is sufficient to show that, for all $M, N \in \Lambda^0$,

$$M \widehat{\approx}^G_\sigma N \ \wedge \ M \in \mathcal{V}_\sigma \implies N \in \mathcal{V}_\sigma \ .$$

Hence we can state the following

Theorem 7.7.4 *If, for all $M, N \in \Lambda^0$,*

$$M \widehat{\approx}^G_\sigma N \ \wedge \ M \in \mathcal{V}_\sigma \implies N \in \mathcal{V}_\sigma \quad (*) \ ,$$

then $\approx^G_\sigma$ is a congruence w.r.t. application.

The validity of the hypothesis $(*)$ of Theorem 7.7.4 depends on the particular λ-theory. Here we show that hypothesis $(*)$ holds for all the λ-theories derived from the lazy strategies $\to_l$ and $\to_v$ presented in Section 7.3.

The Congruence Candidate Method for Lazy Strategies

We show Theorem 7.7.4 for the observational equivalences $\approx_l$ and $\approx_v$.

Lemma 7.7.5 *Let $\sigma \in \{l, v\}$. Then*

$$M \to_\sigma N \implies M \approx^G_\sigma N \ .$$

Proof It is easy to see that

$$M \to_\sigma N \implies M \approx^{app}_\sigma N \ .$$

The thesis follows from $\approx^{app}_\sigma \subseteq \approx^G_\sigma$. $\Box$

Theorem 7.7.6 *Let $M, N \in \Lambda^0$, and let $\sigma \in \{l, v\}$. Then*

$$M \widehat{\approx}^G_\sigma N \ \wedge \ M \Downarrow_\sigma \lambda x.P \implies (N \Downarrow_\sigma \lambda x.Q \ \wedge \ P \widehat{\approx}^G_\sigma Q) \ .$$

Proof The proof proceeds by induction of the derivation of $M \Downarrow_\sigma \lambda x.P$.

- $M \equiv \lambda x.P :\ \exists N'$ s.t. $\qquad \dfrac{P \widehat{\approx}^G_\sigma N' \quad \lambda x.N' \approx^G_\sigma N}{\lambda x.P \widehat{\approx}^G_\sigma N}$

From the definition of $\approx^G_\sigma$ it follows that there exists Q such that $N \Downarrow_\sigma \lambda x.Q$. By Lemma 7.7.5, $N \approx^G_\sigma \lambda x.Q$, hence, by transitivity of $\approx^G_\sigma$, $\lambda x.N' \approx^G_\sigma \lambda x.Q$. In particular, using again Lemma 7.7.5, it is easy to check that $N' \approx^G_\sigma Q$. hence, from $P \widehat{\approx}^G_\sigma N'$ and $N' \approx^G_\sigma Q$, using item iv) of Lemma 7.7.2, we get $P \widehat{\approx}^G_\sigma Q$.

- $M \equiv M_1 M_2 :\ \exists N_1, N_2$ s.t. $\qquad \dfrac{M_1 \widehat{\approx}^G_\sigma N_1 \quad M_2 \widehat{\approx}^G_\sigma N_2 \quad N_1 N_2 \approx^G_\sigma N}{M_1 M_2 \widehat{\approx}^G_\sigma N}$

We deal with the case $\sigma = l$, the other case is similar. Since $M_1 M_2 \Downarrow_l$, there exist P, P' such that

$$\frac{M_1 \Downarrow_l \lambda x.P' \quad P'[M_2/x] \Downarrow_l \lambda x.P}{M_1 M_2 \Downarrow_l \lambda x.P} \ .$$

By induction hypothesis, since $M_1 \widehat{\approx}_l^G N_1$ and $M_1 \Downarrow_l \lambda x.P'$, there exists Q' such that $N_1 \Downarrow_l \lambda x.Q'$ and $P' \widehat{\approx}_l^G Q'$. Then, from $M_2 \widehat{\approx}_l^G N_2$, by the Substitutivity Lemma, $P'[M_2/x] \widehat{\approx}_l^G Q'[N_2/x]$. Hence, by induction hypothesis, there exists Q such that $Q'[N_2/x] \Downarrow_l \lambda x.Q$ and $P \widehat{\approx}_l^G Q$. $\square$

Chapter 8

π-Calculus

In this chapter we discuss the final semantics for the π-calculus with *late* and *early* operational semantics, and various notions of bisimulations. The π-calculus [MPW92, Mil93] is a process algebra which models communicating systems that can dynamically change the topology of the channels. This is obtained by allowing (channel) names to appear as values in communications. The possibility of reconfiguring process adjacency gives to the π-calculus a richer expressive power than CCS: for instance, π-calculus can be used to model object oriented languages [Wal95], and higher-order communications can be encoded in it [San92].

However, this possibility of communicating names, and hence of dynamically modifying the scope of local names leads to new problems when one tries to extend to the π-calculus the techniques which have been developed for process algebras (see Chapter 5).

In order to extend the final semantics approach to the π-calculus, we have found it convenient to give first a *higher order* presentation of it, using as metalanguage a logical framework based on typed λ-calculus, such as the Edinburgh Logical Framework *LF* [HHP93]. This *LF* presentation provides useful insights into the nature of the various binding operators of the π-calculus thus allowing to focus on the uses of free and bound names. Capitalizing on this, one can overcome those problematic aspects of the π-calculus, which do not allow for a direct reuse of the techniques in [Acz88, Acz93, Rut92, RT93, RT94]. The proposed presentation-encoding of the π-calculus can be proved to be *faithful* to the original system using standard techniques. It can be viewed also as the specification of a *derivation editor* for the π-calculus, in any of the various implementations of logical frameworks based on constructive type theory such as *Coq* [CCF95].

The reader might reasonably ask how crucial is the use of a logical framework in giving the final semantics of the π-calculus. Of course, at least in principle, a logical framework is not *strictly necessary.* A logical framework after all is just a general logic specification language. A logical framework such as the one we use, however, supports higher-order syntax, and a higher order syntax formulation of the π-calculus allows for a clear separation of concerns in dealing with the various forms of binding operators of the language. Hence, as we will see, this specification of the π-calculus naturally suggests a line of approach in providing a final semantics. It is probably the case, however, that something in the line of our encoding has to be done in order to provide a natural semantics of bounded names. The denotational approach of [FMS96] for instance, makes use of operators which have the same type as ours, in dealing with bounded actions. We believe however that more contrived approaches to the final semantics of π-calculus can be achieved also using more traditional first-order transition-based approaches to semantics such as that appearing in [Qua96].

Utilizing the LF presentation, we define coalgebras for various, both *strong* and *weak*, operational semantics of the π-calculus. More precisely, the equivalences induced by finality on these coalgebras are:

- *late bisimulation*;
- *early bisimulation*;
- *open bisimulation*;
- *late congruence*;
- *weak late bisimulation*, which is the equivalence arising from the late bisimulation and the weak bisimulation on process algebras (see Chapter 5, Section 5.2;
- *weak late ground congruence*, which is the ground congruence arising from the late bisimulation and the weak congruence on process algebras;
- *weak late congruence*, which is the congruence arising from the late congruence and the weak congruence on process algebras.

We consider only the late versions of the strong congruence and of the weak bisimulation and congruences, but our techniques can be easily extended to the early and open versions.

Two versions of denotational semantics have been recently defined for the π-calculus in [Sta96] and [FMS96]. However, these approaches just work for *strong* semantics, and, as far as we know, no denotational semantics have been defined for the *weak* case. From this perspective, the final characterization of the weak semantics is particularly interesting.

This chapter in an expansion of [HLMP98], where only the strong and weak late operational semantics is considered, and final semantics for late bisimulation, late congruence, and weak late bisimulation is provided.

This chapter is organized as follows. In Section 8.1, we review the standard syntax and operational semantics of the π-calculus. In Section 8.2, we show how a convenient presentation of the operational semantics of the π-calculus can be given using LF as metalanguage; in particular, we encode the *late strong* and the *early strong transition relations*, and two weak transitions, i.e. the *weak late transition* and the *weak* late transition*, corresponding to weak late bisimulation, and to weak late ground congruence and weak late congruence, respectively. In

Section 8.3, we present the definitions of the various notions of bisimulations and congruences, for which we provide final descriptions in Section 8.4. Finally, in Section 8.4.1, we discuss an application of the technique by finality, using the final semantics for the π-calculus defined in this chapter.

8.1 Syntax and Operational Semantics

In this section we review the ordinary syntax and operational semantics of (late and early, monadic) π-calculus. More details can be found in [MPW92, Mil93].

Given an infinite set $\mathcal{N}$ of *names*, ranged over by $x, y, z \ldots$. the π-calculus *processes*, denoted by $p, q, r \ldots$, are defined by the syntax:

$$p ::= 0 \;\mid\; \tau.p \;\mid\; x(y).p \;\mid\; \bar{x}y.p \;\mid\; p|q \;\mid\; p{+}q \;\mid\; !p \;\mid\; (\nu x)p \;\mid\; [x=y]p\,.$$

The occurrences of y in $x(y).p$ and $(\nu y)p$ are bound; *free names*, $fn(p)$, are defined as usual. If $\sigma : \mathcal{N} \to \mathcal{N}$, we denote with $p\sigma$ the process p whose free names have been replaced according to substitution σ (possibly with α-conversions to avoid name clashing); we denote with $\{y_1/x_1 \cdots y_n/x_n\}$ the substitution that maps x_i into y_i for $i = 1, \ldots, n$ and which is the identity on the other names. Processes which differ for α-conversion $\cong$ are identified. The *actions* that a process can perform, ranged over by α, are defined by the following syntax:

$$\alpha ::= \tau \;\mid\; x(z) \;\mid\; \bar{x}y \;\mid\; \bar{x}(z)\;.$$

Names x and y are *free names* of α, ($fn(\alpha)$), whereas z is a *bound name* ($bn(\alpha)$); moreover $n(\alpha) = fn(\alpha) \cup bn(\alpha)$. The transitions for the *late operational semantics* are defined by the rules of Table 8.1. The *early operational semantics* differs from the late semantics for the rules IN, COM, and $CLOSE$. The early version of these is presented in Table 8.2.

In transition $p \xrightarrow{x(y)} p'$ (resp. $p \xrightarrow{\bar{x}(y)} p'$), name y represents a reference in p' for the place where the received name of the input will go (resp. where the private name which is emitted occurs). Since y is a reference, it has to be a *fresh* name, i.e., a name which is different from all the other names of p'. The side-conditions which appear in the rules PAR, $OPEN$, and early $CLOSE$ are required to enforce this. These transitions are called *bound transitions* (in contrast, τ- and $\bar{x}y$-transitions are called *free transitions*).

8.2 An LF Encoding of the π-Calculus

In this section we reformulate the syntax and the operational semantics of the π-calculus using as metalanguage the Edinburgh Logical Framework LF [HHP93]. Logical Frameworks are general *specification languages* for formal systems where one can express uniformly all the features and aspects of an arbitrary system, e.g.: syntactic categories, variables, syntactic constructors, binding operators,

Table 8.1: Late operational semantics

$$TAU : \tau.p \xrightarrow{\tau} p \qquad OUT : \bar{x}y.p \xrightarrow{\bar{x}y} p \qquad IN : x(y).p \xrightarrow{x(y)} p$$

$$SUM^l : \frac{p \xrightarrow{\alpha} p'}{p+q \xrightarrow{\alpha} p'} \qquad PAR^l : \frac{p \xrightarrow{\alpha} p'}{p|q \xrightarrow{\alpha} p'|q} \text{if } bn(\alpha) \cap fn(q) = \emptyset$$

$$COM^l : \frac{p \xrightarrow{\bar{x}y} p' \quad q \xrightarrow{x(z)} q'}{p|q \xrightarrow{\tau} p'|q'\{y/z\}} \qquad CLOSE^l : \frac{p \xrightarrow{\bar{x}(y)} p' \quad q \xrightarrow{x(y)} q'}{p|q \xrightarrow{\tau} (\nu y)(p'|q')}$$

$$RES : \frac{p \xrightarrow{\alpha} p'}{(\nu x)p \xrightarrow{\alpha} (\nu x)p'} \text{if } x \notin n(\alpha) \qquad OPEN : \frac{p \xrightarrow{\bar{x}y} p'}{(\nu y)p \xrightarrow{\bar{x}(y)} p'} \text{if } x \neq y$$

$$BANG : \frac{p|!p \xrightarrow{\alpha} p'}{!p \xrightarrow{\alpha} p'} \qquad COND : \frac{p \xrightarrow{\alpha} p'}{[x{=}x]p \xrightarrow{\alpha} p'}$$

$$EQ : \frac{p \equiv p' \quad p' \xrightarrow{\alpha} q' \quad q' \equiv q}{p \xrightarrow{\alpha} q}$$

$$IN^e : x(y).p \xrightarrow{x(z)}_e p\{z/y\}$$

$$COM_l^e : \frac{p \xrightarrow{\bar{x}y}_e p' \quad q \xrightarrow{x(y)}_e q'}{p|q \xrightarrow{\tau}_e p'|q'} \qquad CLOSE_l^e : \frac{p \xrightarrow{\bar{x}(y)}_e p' \quad q \xrightarrow{x(y)}_e q'}{p|q \xrightarrow{\tau}_e (\nu y)(p'|q')} \text{ if } y \notin fn(q')$$

Table 8.2: Early operational semantics

substitution mechanisms, judgements, rules, derivations, etc. Encodings in Logical Frameworks readily provide "derivation editors" for the "object logic", given an editor for the metalanguage.

LF is based on a dependent typed λ-calculus and it exploits the well known "judgements (formulæ) as types", "proofs (derivations) as λ-terms" paradigm. *LF* allows also a smooth treatment of binding operators in terms of *higher order syntax* à la Church. More specifically, *LF* is a system for deriving typing assertions of the shape $\Gamma \vdash P : Q$, whose intended meaning is "in the environment Γ, P is classified by Q". Three kinds of entities are involved, i.e. *terms* (ranged over by M, N), *types* and *typed valued functions* (ranged over by A, B), and *kinds* (ranged over by K). Types are used to classify terms, and kinds are used to classify types and typed valued functions. These entities are defined by the following abstract syntaxes:

$M ::= x \mid MN \mid \lambda x : A.M$
$A ::= X \mid AM \mid \Pi_{x:A}.B \mid \lambda x : A.B$
$K ::= Type \mid \Pi_{x:A}.K.$

Π is the *dependent type* constructor. Intuitively $\Pi_{x:A}.B(x)$ denotes the type of those functions, f, whose domain is A but whose values belong to a codomain *depending* on the input, i.e. $f(a) \in B(a)$, for all $a \in A$. Hence, in *LF*, $A \rightarrow B$ is just notation abbreviating $\Pi_{x:A}.B$, when x does not occur free in B ($x \notin FV(B)$). In the "propositions-as-types" analogy the dependent product type $\Pi_{x:A}.B(x)$ represents the proposition $\forall x : A.\ B(x)$.

Environments are lists of pairs consisting of a variable and its type or kind.

The *LF* specification of a formal system is given by a particular environment, called *signature*, determined according to the following methodology. The *syntactic categories* of the abstract syntax language are encoded as type-variables, *syntactic constructors* are encoded as term-variables of the appropriate type, *judgements* are encoded as variables of "typed-valued function" kind, and *rules* are encoded as term-variables of the appropriate types. Hence, derivations of a given assertion are represented as terms of the type corresponding to the assertion in question, so that *checking* the correctness of a derivation amounts just to *type-checking*. Thus, *derivability* of an assertion corresponds to the *inhabitation* of the type corresponding to that assertion, i.e. the derivability of a term of that type.

Metalanguage variables play also the rôle of "object logic" variables so that *schemata* are represented as λ-abstractions and *instantiation* amounts to β-reduction. Hence *binding operators* are encoded as *higher order* variables, being viewed as ranging over schemata.

8.2.1 The LF-signature Σ_π of the π-Calculus

Syntactic Categories
The syntactic categories for names, labels and processes are encoded by the following types :

$$
\begin{array}{lcl}
name & : & Type \\
label & : & Type \\
proc & : & Type
\end{array}
$$

Syntactic Constructors

Syntactic constructors are defined for labels and processes. Functional types are used for constructors, e.g: the constructor *out* for output requires two names to generate a label and so it is represented by assigning to *out* type $name \rightarrow name \rightarrow label$.

- constructors for *label*:

$$
\begin{array}{lcl}
\tau & : & label \\
in & : & name \rightarrow name \rightarrow label \\
out & : & name \rightarrow name \rightarrow label
\end{array}
$$

- constructors for *proc*:

$$
\begin{array}{lcl}
0 & : & proc \\
\tau_{pref} & : & proc \rightarrow proc \\
in_{pref} & : & name \rightarrow (name \rightarrow proc) \rightarrow proc \\
out_{pref} & : & name \rightarrow name \rightarrow proc \rightarrow proc \\
\mid & : & proc \rightarrow proc \rightarrow proc \\
+ & : & proc \rightarrow proc \rightarrow proc \\
! & : & proc \rightarrow proc \\
\nu & : & (name \rightarrow proc) \rightarrow proc \\
[\ =\ ] & : & name \rightarrow name \rightarrow proc \rightarrow proc
\end{array}
$$

For sake of simplicity we do not consider the mismatch constructor $[x \neq y]$. Notice that labels are defined only for free transitions: we will see below how bound transitions, such as bound output, are represented by giving a higher order type to the corresponding transition judgment. We shall freely pretty-print $in(x, y)$ as xy and $out(x, y)$ as $\bar{x}y$.

Notice how π-calculus binders are expressed in LF making use of suitable abstractions. E.g.: in the case of the input prefix, in order to obtain a process, a name (the channel for the input) and an abstraction (i.e., a function from names to processes) are required. Similarly, restrictions are encoded using abstractions. This is the core of higher order syntax.

It is easy to define a translation function $\mathbf{t}$ from π-calculus processes, as defined in Section 8.1, to elements of type *proc*. Most of the π-calculus constructors are translated in the obvious way; the only interesting cases are input prefix and restrictions:
$\mathbf{t}(x(y).p) = in_{pref}(x, \lambda y : name.\ \mathbf{t}(p))$ $\qquad$ $\mathbf{t}((\nu y)p) = \nu(\lambda y : name.\ \mathbf{t}(p))$.
Accordingly, π-calculus processes will be often pretty-printed, following the notation of Section 8.1.

Judgments

We introduce two judgments for encoding the late transition, and two judgements for encoding the early transition:

$$
\begin{array}{lcl}
\longmapsto & : & proc \to label \to proc \to Type \\
\longmapsto\!\!\!\!\!\!\twoheadrightarrow & : & proc \to (name \to label) \to (name \to proc) \to Type \\
\longmapsto_e & : & proc \to label \to proc \to Type \\
\longmapsto\!\!\!\!\!\!\twoheadrightarrow_e & : & proc \to (name \to label) \to (name \to proc) \to Type
\end{array}
$$

Judgment $\longmapsto$ ($\longmapsto_e$) is used to encode the free transition assertion. Its meaning should be clear. We will often write $p \overset{\alpha}{\longmapsto} p'$ ($p \overset{\alpha}{\longmapsto}_e p'$) instead of $\longmapsto p\,\alpha\,p'$ ($\longmapsto_e\ p\,\alpha\,p'$). Judgment $\longmapsto\!\!\!\!\!\twoheadrightarrow$ ($\longmapsto\!\!\!\!\!\twoheadrightarrow_e$) is used to encode bound transitions. We will write $p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p'$ ($p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow}_e p'$) instead of $\longmapsto\!\!\!\!\!\twoheadrightarrow (p, \alpha, p')$ ($\longmapsto\!\!\!\!\!\twoheadrightarrow_e (p, \alpha, p')$). In this case, both the label and the target are abstractions (of a *label* and of a *proc* respectively).

Rules rules for $\longmapsto$ and $\longmapsto\!\!\!\!\!\twoheadrightarrow$ appear in Table 8.3 (rules SUM_f^r, SUM_b^r, PAR_f^r, PAR_b^r, COM^r and $CLOSE^r$ have been omitted), and rules for and $\longmapsto_e$ and $\longmapsto\!\!\!\!\!\twoheadrightarrow_e$ appear in Table 8.4: notice that some of the rules of Table 8.1 and Table 8.2 have two counterparts, according to whether they refer to free or bound transitions.

Some comments on the rules of Table 8.3 and Table 8.4 are in order.

- No explicit axiomatization of α-conversion is needed. Since π-calculus binders are explained in terms of abstractions, α-conversion is delegated to the metalanguage and hence it is for free.

- In rule IN, both the label and the target process are abstractions; the idea is that, when an input occurs, the target is applied to the effectively received name, so that this name replaces all the occurrences of the placeholder in the target process. This happens, for instance, when rule COM is applied: in this case, the target q_1 of the input transition is applied to the name z, which is the message of the output transition.

- Consider now rules RES: in Table 8.1 these rules has a side-condition; namely a transition of p with label α can be performed also by $(\nu x)p$ *provided* x does not appear among the names of α. This side-condition is expressed in the LF encoding by requiring that the process px perform the transition independently from the name x, which is chosen to instantiate abstraction p. In this way, the variable x, being bound, behaves as "generic" hence different from all the names in α.
  - Rule $OPEN$ is similar; in this case, however, according to Table 8.1, the value of the free output has to coincide with the name of the restriction. The free output becomes a bound output, so the restriction is removed in the target state, which then becomes an abstraction.

- In rule $CLOSE$ an input and a bound output are synchronized. In this case, the fact that the values of the two transitions should be identified

Table 8.3: LF rules

TAU	$: \Pi_{p:proc}.\ \tau.p \overset{\tau}{\longmapsto} p$
OUT	$: \Pi_{p:proc}\Pi_{x,y:name}.\ \bar{x}y.p \overset{\bar{x}y}{\longmapsto} p$
IN	$: \Pi_{p:name\to proc}\Pi_{x:name}.\ in_{pref}(x,p) \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p$
SUM^l_f	$: \Pi_{p,q,r:proc}\Pi_{\alpha:label}.\ p \overset{\alpha}{\longmapsto} r \to (p{+}q) \overset{\alpha}{\longmapsto} r$
SUM^l_b	$: \Pi_{p,q:proc}\Pi_{r:name\to proc}\Pi_{\alpha:name\to label}.\ p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} r \to (p{+}q) \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} r$
PAR^l_f	$: \Pi_{p,q,r:proc}\Pi_{\alpha:label}.\ p \overset{\alpha}{\longmapsto} r \to (p\vert q) \overset{\alpha}{\longmapsto} (r\vert q)$
PAR^l_b	$: \Pi_{p,q:proc}\Pi_{r:name\to proc}\Pi_{\alpha:name\to label}.\ p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} r \to$ $(p\vert q) \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} (\lambda x : name.\ (rx\vert q))$
COM^l	$: \Pi_{p,q,r:proc}\Pi_{s:name\to proc}\Pi_{x,y:name}.\ p \overset{\bar{x}y}{\longmapsto} r \to$ $q \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} s \to (p\vert q) \overset{\tau}{\longmapsto} (r\vert sy)$
RES_f	$: \Pi_{p,q:name\to proc}\Pi_{\alpha:label}.(\Pi_{x:name}.px \overset{\alpha}{\longmapsto} qx) \to \nu(p) \overset{\alpha}{\longmapsto} \nu(q)$
RES_b	$: \Pi_{p:name\to proc}\Pi_{q:name\to name\to proc}\Pi_{\alpha:name\to label}.$ $(\Pi_{x:name}.\ px \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} \lambda z : name.\ qzx) \to$ $\nu(p) \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} \lambda z : name.\ \nu(qz)$
$OPEN$	$: \Pi_{p,q:name\to proc}\Pi_{x:name}.$ $(\Pi_{y:name}.\ py \overset{\bar{x}y}{\longmapsto} qy) \to \nu(p) \overset{\lambda z:name.\ \bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q$
$CLOSE^l$	$: \Pi_{p,q:proc}\Pi_{r,s:name\to proc}\Pi_{x:name}.\ p \overset{\lambda z:name.\ \bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} r \to$ $q \overset{\lambda z:name.\ xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} s \to (p\vert q) \overset{\tau}{\longmapsto} \nu(\lambda z : name.\ (rz\vert sz))$
$BANG_f$	$: \Pi_{p,q:proc}\Pi_{\alpha:label}.\ p\vert !p \overset{\alpha}{\longmapsto} q \to\ !p \overset{\alpha}{\longmapsto} q$
$BANG_b$	$: \Pi_{p:proc}\Pi_{q:name\to proc}\Pi_{\alpha:name\to label}.\ p\vert !p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q \to\ !p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q$
$COND_f$	$: \Pi_{p,q:proc}\Pi_{\alpha:label}\Pi_{x:name}.\ p \overset{\alpha}{\longmapsto} q \to [x{=}x]p \overset{\alpha}{\longmapsto} q$
$COND_b$	$: \Pi_{p:proc}\Pi_{q:name\to proc}\Pi_{\alpha:name\to label}\Pi_{x:name}.$ $p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q \to [x{=}x]p \overset{\alpha}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q$

IN^e	:	$\Pi_{p:name\to proc}\Pi_{x:name}.\ in_{pref}(x,p) \overset{xz}{\longmapsto}_e pz$
COM_l^e	:	$\Pi_{p_1,p_2,q_1,q_2:proc}\Pi_{x,y:name}.\ p_1 \overset{xy}{\longmapsto}_e q_1 \to p_2 \overset{\bar{x}y}{\longmapsto}_e q_2 \to$ $p_1 \vert p_2 \overset{\tau}{\longmapsto}_e q_1 \vert q_2$
COM_r^e	:	$\Pi_{p_1,p_2,q_1,q_2:proc}\Pi_{x,y:name}.\ p_1 \overset{xy}{\longmapsto}_e q_1 \to p_2 \overset{\bar{x}y}{\longmapsto}_e q_2 \to$ $p_2 \vert p_1 \overset{\tau}{\longmapsto}_e q_2 \vert q_1$
$CLOSE_l^e$	:	$\Pi_{p_1,p_2:proc}\Pi_{q_1,q_2:name\to proc}\Pi_{x:name}.\ p_1 \overset{\lambda z:name.\ \bar{x}z}{\longmapsto\!\!\!\!\twoheadrightarrow}_e q_1 \to$ $\Pi_{y:name}.\ p_2 \overset{xy}{\longmapsto}_e q_2 y \to p_1 \vert p_2 \overset{\tau}{\longmapsto}_e \nu(\lambda z : name.\ (q_1 z \vert q_2 z))$
$CLOSE_r^e$	:	$\Pi_{p_1,p_2:proc}\Pi_{q_1,q_2:name\to proc}\Pi_{x:name}.\ p_1 \overset{\lambda z:name.\ \bar{x}z}{\longmapsto\!\!\!\!\twoheadrightarrow}_e q_1 \to$ $\Pi_{y:name}.\ p_2 \overset{xy}{\longmapsto}_e q_2 y \to p_2 \vert p_1 \overset{\tau}{\longmapsto}_e \nu(\lambda z : name.\ (q_2 z \vert q_1 z))$

Table 8.4: LF rules for early semantics

is obtained by combining the target processes $\lambda z : name.\ q_1 z$ and $\lambda z : name.\ q_2 z$ into $\lambda z : name.\ (q_1 z | q_2 z)$.

The operational semantics we have presented here has some advantages w.r.t. the ordinary one, deriving from the usage of functions to model bound transitions. First of all, one can eliminate those explicit side-conditions, which are needed in the ordinary semantics to enforce that the bound names be fresh, so as to avoid name clashing. In the *LF* presentation, the side conditions are implicit in the higher-order nature of the type encoding the rule. Bound names remain *bound* also when the transition is performed, since abstractions are used for the label and for the target process. As we will see, the names which have been abstracted in the targets of bound transitions will be instantiated in the definition of bisimulation. Another advantage is that name substitution is not required in the operational semantics (and in the definition of bisimulation): function applications are sufficient to represent all the required forms of name instantiation.

We did not deal with the mismatch operator, $[\ \neq\ ]$. We could have easily done so at the price of an increase in the complexity of the *LF* encoding of the π-calculus. In fact, in order to produce an adequate encoding of the mismatch operator, it is not enough to add an explicit judgement to express *name diversity* and the rule for mismatch. Because of technical reasons, we need to modify also all rules in Table 8.3 which have a premise with a locally bound name. The fact that such a name is different from all the ones which can possibly occur in the global parameters has to be explicitly asserted. Perhaps this increase in complexity is an indication that such an operator has not as an immediate semantics as the others.

Let Σ_π denote the list of variable declarations introduced above. The signature Σ_π is the LF presentation of the π-calculus. Throughout this chapter we shall assume that we have an infinite supply of different variables of type $name$, standing for the names in $\mathcal{N}$, and we shall work, even without explicit mention, in the environment Γ_π defined as follows: $\Gamma_\pi \cong \Sigma_\pi \cup \{x_i : name\}_{i\in\mathbf{N}}$. We shall denote with the name of the type (i.e. $name$, $label$, $proc$) the collection of all canonical (normal) well-typed terms of that type, derivable in the context Γ_π. Moreover, for $a \subseteq name$, we shall denote with $label_a$ the subset of $label$, consisting only of normal forms whose free variables appear in a.

In view of the remarks above, the set $fn(p)$ of free names of $p \in proc$ is therefore simply the set of *free variables* of type $name$, occurring in the canonical (normal) form of p.

It is possible to establish a precise correspondence between LF judgements derivable in the environment Γ_π and the corresponding concepts phrased in terms of the system in Section 8.1. This takes the form of an *Adequacy Theorem* for the encoding. Here, we do this just for late transitions. The proof of the following proposition is by induction on the structure of derivations (only if part) and on the structure of *normal forms* (if part).

Proposition 8.2.1 (Adequacy) *Let p_1, p_2 be π-calculus processes and let x, y be names in Γ_π; then:*

- $p_1 \xrightarrow{\tau} p_2$ *if and only if the type* $p_1 \xmapsto{\tau} p_2$ *is inhabited in* Γ_π.
- $p_1 \xrightarrow{\bar{x}y} p_2$ *if and only if the type* $p_1 \xmapsto{\bar{x}y} p_2$ *is inhabited in* Γ_π.
- $p_1 \xrightarrow{x(y)} p_2$ *if and only if the type* $p_1 \overset{\lambda y:name.\ xy}{\longmapsto\!\!\!\!\rightarrow} \lambda y : name.\ p_2$ *is inhabited in* Γ_π.
- $p_1 \xrightarrow{\bar{x}(y)} p_2$ *if and only if the type* $p_1 \overset{\lambda y:name.\ \bar{x}y}{\longmapsto\!\!\!\!\rightarrow} \lambda y : name.\ p_2$ *is inhabited in* Γ_π.

LF Encoding of Weak Transition Relations We give the LF encoding of the following weak transition relations on π-calculus, by suitably extending the LF signature Σ_π:

- the *weak late transition*, which arises by merging the late transition and the weak transition relation inducing the weak equivalence on process algebras (see Chapter 5, Definition 5.2.13);
- the *weak* late transition*, which arises by merging the late transition and the weak congruence transition relation inducing the weak congruence on process algebras (see Chapter 5, Definition 5.2.14).

For the weak late transition we introduce the following two judgements:

$$\begin{array}{lcl} \Longmapsto & : & proc \to label \to proc \to Type \\ \Longmapsto\!\!\!\!\rightarrow & : & proc \to (name \to label) \to (name \to proc) \to Type \end{array}$$

$weak1_\tau$	:	$\Pi_{p:proc}.\ p \overset{\tau}{\Longmapsto} p$
$weak2_\tau$	:	$\Pi_{p_1,p_2,p_3:proc}.\ p_1 \overset{\tau}{\Longmapsto} p_2 \to p_2 \overset{\tau}{\longmapsto} p_3 \to p_1 \overset{\tau}{\Longmapsto} p_3$
$weak_{out}$	:	$\Pi_{p_1,p_2,p_3,p_4:proc}\Pi_{x,y:name}.\ p_1 \overset{\tau}{\Longmapsto} p_2 \to p_2 \overset{\bar{x}y}{\longmapsto} p_3 \to p_3 \overset{\tau}{\Longmapsto} p_4 \to p_1 \overset{\bar{x}y}{\Longmapsto} p_4$
$weak_{in}$	:	$\Pi_{p_1,p_2:proc}\Pi_{p_3:name\to proc}\Pi_{x:name}.\ p_1 \overset{\tau}{\Longmapsto} p_2 \to p_2 \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\to} p_3 \to p_1 \overset{\lambda z:name.xz}{\Longmapsto\!\!\!\!\to} p_3$
$weak_{bout}$	:	$\Pi_{p_1,p_2:proc}\Pi_{p_3,p_4:name\to proc}\Pi_{x:name}.\ p_1 \overset{\tau}{\Longmapsto} p_2 \to p_2 \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\to} p_3 \to (\forall z : name.\ p_3 z \overset{\tau}{\Longmapsto} p_4 z) \to p_1 \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\!\to} p_4$

Table 8.5: LF rules for weak late semantics

For the weak* late transition we introduce the following two judgements:

$\Longmapsto_* \quad : \quad proc \to label \to proc \to Type$

$\Longmapsto\!\!\!\!\to_* \quad : \quad proc \to (name \to label) \to (name \to proc) \to Type$

The rules for $\Longmapsto$ and $\Longmapsto\!\!\!\!\to$ appear in Table 8.5, while the rules for $\Longmapsto_*$ and $\Longmapsto\!\!\!\!\to_*$ appear in Table 8.6.

As noticed in [San96] for the weak versions of late, early and open bisimulations, weak early and weak open transitions are obtained just allowing a possible empty sequence of τ's actions before and after a (possibly empty) strong transition. But for the weak late bisimulation, input transitions cannot be followed by τ's actions.

8.3 Bisimulations on π-Calculus

We now present the definition of various notions of bisimulation on the π-calculus, making use of the LF presentation of the π-calculus, Σ_π. Using Proposition 8.2.1, one can show that these are equivalent to the classical ones.

Definition 8.3.1 (Late Strong Bisimulation) *A symmetric relation $\mathcal{R} \subseteq proc \times proc$ is a* late strong bisimulation *if and only if, $\forall p, q : proc$, whenever $p\,\mathcal{R}\,q$ then:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p \overset{\alpha}{\longmapsto} p_1 \to (\exists q_1 : proc.\ q \overset{\alpha}{\longmapsto} q_1 \wedge p_1\,\mathcal{R}\,q_1)$,

$weak^*1_\alpha$	:	$\Pi_{p_1,p_2:proc}\Pi_{\alpha:label}.\ p_1 \overset{\alpha}{\longmapsto} p_2 \to p_1 \overset{\alpha}{\Longmapsto}_* p_2$
$weak^*2_\alpha$	:	$\Pi_{p_1:proc}\Pi_{p_2:name\to proc}\Pi_{\alpha:name\to label}.\ p_1 \overset{\alpha}{\longmapsto\!\!\!\twoheadrightarrow} p_2 \to p_1 \overset{\alpha}{\Longmapsto\!\!\!\twoheadrightarrow}_* p_2$
$weak^*1_\tau$	:	$\Pi_{p_1,p_2,p_3:proc}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\tau}{\longmapsto} p_3 \to p_1 \overset{\tau}{\Longmapsto}_* p_3$
$weak^*2_\tau$	:	$\Pi_{p_1,p_2,p_3:proc}.\ p_1 \overset{\tau}{\longmapsto} p_2 \to p_2 \overset{\tau}{\Longmapsto}_* p_3 \to p_1 \overset{\tau}{\Longmapsto}_* p_3$
$weak^*1_{out}$	:	$\Pi_{p_1,p_2,p_3:proc}\Pi_{x,y:name}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\bar{x}y}{\longmapsto} p_3 \to p_1 \overset{\bar{x}y}{\Longmapsto}_* p_3$
$weak^*2_{out}$	:	$\Pi_{p_1,p_2,p_3:proc}\Pi_{x,y:name}.\ p_1 \overset{\bar{x}y}{\longmapsto} p_2 \to p_2 \overset{\tau}{\Longmapsto}_* p_3 \to p_1 \overset{\bar{x}y}{\Longmapsto}_* p_3$
$weak^*3_{out}$	:	$\Pi_{p_1,p_2,p_3,p_4:proc}\Pi_{x,y:name}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\bar{x}y}{\longmapsto} p_3 \to p_3 \overset{\tau}{\Longmapsto}_* p_4 \to p_1 \overset{\bar{x}y}{\Longmapsto}_* p_4$
$weak^*{}_{in}$	:	$\Pi_{p_1,p_2:proc}\Pi_{p_3:name\to proc}\Pi_{x:name}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\lambda z:name.xz}{\longmapsto\!\!\!\twoheadrightarrow} p_3 \to p_1 \overset{\lambda z:name.xz}{\Longmapsto\!\!\!\twoheadrightarrow}_* p_3$
$weak^*1_{bout}$	:	$\Pi_{p_1,p_2:proc}\Pi_{p_3:name\to proc}\Pi_{x:name}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\twoheadrightarrow} p_3 \to p_1 \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\twoheadrightarrow}_* p_3$
$weak^*2_{bout}$	:	$\Pi_{p_1:proc}\Pi_{p_2,p_3:name\to proc}\Pi_{x:name}.\ p_1 \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\twoheadrightarrow} p_2 \to (\forall z:name.p_2z \overset{\tau}{\Longmapsto}_* p_3z) \to p_1 \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\twoheadrightarrow}_* p_3$
$weak^*3_{bout}$	:	$\Pi_{p_1,p_2:proc}\Pi_{p_3,p_4:name\to proc}\Pi_{x:name}.\ p_1 \overset{\tau}{\Longmapsto}_* p_2 \to p_2 \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\twoheadrightarrow} p_3 \to (\forall z:name.p_3z \overset{\tau}{\Longmapsto}_* p_4z) \to p_1 \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\twoheadrightarrow}_* p_4$

Table 8.6: LF rules for weak* late semantics

- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\rightarrow} p_1 \to$
$(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\rightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1) \to p_1 y\ \mathcal{R}\ q_1 y))$, *and*

- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\rightarrow} p_1 \to$
$(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\rightarrow} q_1 \wedge \forall y : name.\ p_1 y\ \mathcal{R}\ q_1 y)$.

Let $p, q : proc$; p is late strongly bisimilar *to q if there exists a late strong bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all late strong bisimulations (denoted by $\approx^l$) is a late strong bisimulation itself.*

Definition 8.3.2 (Early Strong Bisimulation) *A symmetric relation $\mathcal{R} \subseteq proc \times proc$ is an* early strong bisimulation *if and only if, $\forall p, q : proc$, whenever $p\ \mathcal{R}\ q$ then:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p \overset{\alpha}{\longmapsto} p_1 \to (\exists q_1 : proc.\ q \overset{\alpha}{\longmapsto} q_1 \wedge p_1\ \mathcal{R}\ q_1)$, *and*

- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\rightarrow} p_1 \to$
$(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\rightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1) \to p_1 y\ \mathcal{R}\ q_1 y))$.

Let $p, q : proc$; p is early strongly bisimilar *to q if there exists an early strong bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all early strong bisimulations (denoted by $\approx^e$) is an early strong bisimulation itself.*

It is well known that the equivalences $\approx^l$ and $\approx^e$ are not (higher order) congruences. The largest congruences included in $\approx^l$ and $\approx^e$ respectively, are obtained by allowing to instantiate the names in the processes to be compared, before starting to match their transitions. Let $\sigma : name \to name$ denote a substitution, and let $p\sigma$, for $p \in proc$, be the process obtained from p by substituting each free name x appearing in p with the name $\sigma(x)$, possibly renaming bounded names to avoid capture of free names.

Definition 8.3.3 (Late, Early Congruences)

- *Let $\approx^{lc} \subseteq proc \times proc$ be the* late congruence *defined by*

$$p \approx^{lc} q \iff \forall \sigma.\ p\sigma \approx^l q\sigma\ .$$

- *Let $\approx^{ec} \subseteq proc \times proc$ be the* early congruence *defined by*

$$p \approx^{ec} q \iff \forall \sigma.\ p\sigma \approx^e q\sigma\ .$$

Notice that the notions of late and early congruences have not coinductive definitions which allows for an immediate final description. The difficulty in capturing coinductively by finality late and early congruences lies in the fact that substitutions of names are applied only to the starting pair of processes. But according to the final semantics paradigm, any functor has to act uniformly on all the processes; and so there is no natural notion of "initial pair". At the price of a little awkwardness, however, we can proceed as follows. We split the type of names into two infinite subsets, e.g $name_{even}$ consisting of those names with an *even* index and $name_{odd}$, consisting of those names with an *odd* index. We assume that initial π-calculus processes have only names in $name_{odd}$. By considering processes only "up to" substitutions of the form $\sigma : name \to name_{even}$, which are *injective* on $name_{even}$, we can then sidestep the problem given by the absence of an "initial pair". Hence a coinductive characterization of late and early congruences can be given as follows:

Lemma 8.3.4 *Let p, q be π-calculus processes with names in $name_{odd}$.*

- $p \approx^{lc} q$ *if and only if the pair (p, q) belongs to a symmetric relation $\mathcal{R} \subseteq proc \times proc$ satisfying: whenever $p \; \mathcal{R} \; q$, then, for all substitutions $\sigma : name \to name_{even}$,* injective *on $name_{even}$,*
  - $\forall \alpha : label.\ \forall p_1 : proc.\ p\sigma \overset{\alpha}{\longmapsto} p_1 \to (\exists q_1 : proc.\ q\sigma \overset{\alpha}{\longmapsto} q_1 \wedge p_1 \; \mathcal{R} \; q_1)$,
  - $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
    $(\exists q_1 : name \to proc.\ q\sigma \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ p_1 y \; \mathcal{R} \; q_1 y)$, *and*
  - $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
    $(\exists q_1 : name \to proc.\ q\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1) \to p_1 y \; \mathcal{R} \; q_1 y))$.
- $p \approx^{ec} q$ *if and only if the pair (p, q) belongs to a symmetric relation relation $\mathcal{R} \subseteq proc \times proc$ satisfying: whenever $p \; \mathcal{R} \; q$, then, for all substitutions $\sigma : name \to name_{even}$,* injective *on $name_{even}$,*
  - $\forall \alpha : label.\ \forall p_1 : proc.\ p\sigma \overset{\alpha}{\longmapsto} p_1 \to (\exists q_1 : proc.\ q\sigma \overset{\alpha}{\longmapsto} q_1 \wedge p_1 \; \mathcal{R} \; q_1)$, *and*
  - $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
    $(\exists q_1 : name \to proc.\ q\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1) \to p_1 y \; \mathcal{R} \; q_1 y))$.

A notion of bisimulation which is a congruence is the *open bisimulation* introduced in [San96]s. The open bisimulation is obtained from the late bisimulation, comparing all the pairs of processes obtained from the initial pair of processes by applying to them all possible substitutions:

Definition 8.3.5 (Open Bisimulation) *A symmetric relation $\mathcal{R} \subseteq proc \times proc$ is an* open bisimulation *if and only if, $\forall p, q : proc$, whenever $p \; \mathcal{R} \; q$ then, for all substitutions $\sigma : name \to name$:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p\sigma \overset{\alpha}{\longmapsto} p_1 \to (\exists q_1 : proc.\ q\sigma \overset{\alpha}{\longmapsto} q_1 \wedge p_1 \; \mathcal{R} \; q_1)$,
- $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
  $(\exists q_1 : name \to proc.\ q\sigma \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ p_1 y \; \mathcal{R} \; q_1 y)$, *and*

- $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\!\twoheadrightarrow} p_1 \to$
  $(\exists q_1 : name \to proc.\ q\sigma \overset{\lambda z:name.\bar{x}z}{\longmapsto\!\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1) \to$
  $p_1 y\ \mathcal{R}\ q_1 y))$.

Let $p, q : proc$; p is open bisimilar *to q if there exists an open bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all open bisimulations (denoted by $\approx^o$) is an open bisimulation itself.*

Now we discuss the LF representation of the weak versions of the bisimulations and congruences listed above. We have all the notions of bisimulations which arise by combining the notions of *weak* bisimulation and congruence on process algebras with the late bisimulation and congruence. Here we present the LF representation only for the late versions. The other cases can be dealt with similarly.

Merging weak bisimulation on process algebras and late bisimulation, we get the notion of *weak late bisimulation*, which is defined in terms of the weak late transition (see Table 8.5).

Definition 8.3.6 (Weak Late Bisimulation) *A* weak late bisimulation *is a symmetric relation $\mathcal{R} \subseteq proc \times proc$ such that, $\forall p, q : proc$, whenever $p\ \mathcal{R}\ q$ then:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p \overset{\alpha}{\Longmapsto} p_1 \to (\exists q_1 : proc.\ q \overset{\alpha}{\Longmapsto} q_1 \wedge p_1\ \mathcal{R}\ q_1)$,
- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
  $(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1)) \to$
  $p_1 y\ \mathcal{R}\ q_1 y))$ *, and*
- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.xz}{\Longmapsto\!\!\!\!\!\twoheadrightarrow} p_1 \to$
  $(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.xz}{\Longmapsto\!\!\!\!\!\twoheadrightarrow} q_1 \wedge \forall y : name\ p_1 y\ \mathcal{R}\ q_1 y)$.

Let $p, q : proc$. Then p is weak late bisimilar *to q if there exists a weak late bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all weak late bisimulations (denoted by $\approx^{wl}$) is a weak late bisimulation itself.*

Merging weak congruence on process algebras and late bisimulation, we get the notion of *weak late ground congruence*, which is defined in terms of the weak* late transition (see Table 8.6).

Definition 8.3.7 (Weak Late Ground Congruence) *A symmetric relation $\mathcal{R} \subseteq proc \times proc$ is a* weak* late bisimulation *if and only if, $\forall p, q : proc$, whenever $p\ \mathcal{R}\ q$ then:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p \overset{\alpha}{\Longmapsto}_* p_1 \to (\exists q_1 : proc.\ q \overset{\alpha}{\Longmapsto}_* q_1 \wedge p_1\ \mathcal{R}\ q_1)$,
- $\forall x : name.\ \forall p_1 : name \to proc.\ p \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\!\!\twoheadrightarrow_*} p_1 \to$
  $(\exists q_1 : name \to proc.\ q \overset{\lambda z:name.\bar{x}z}{\Longmapsto\!\!\!\!\!\twoheadrightarrow_*} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1)) \to$
  $p_1 y\ \mathcal{R}\ q_1 y))$ *, and*

- $\forall x : name.\ \forall p_1 : name \to proc.\ p \stackrel{\lambda z:name.xz}{\Longmapsto_*} p_1 \to$ $(\exists q_1 : name \to proc.\ q \stackrel{\lambda z:name.xz}{\Longmapsto_*} q_1 \wedge \forall y : name\ p_1 y\ \mathcal{R}\ q_1 y)$.

Let p, q : proc. Then p is weak* late bisimilar *to q if there exists a weak* late bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all weak* late bisimulations, called* weak late ground congruence *(denoted by $\approx^{wlgc}$) is a weak* late bisimulation itself.*

Finally, merging weak congruence on process algebras and late congruence, we get the notion of *weak late congruence*, which again is defined in terms of the weak* late transition.

Definition 8.3.8 (Weak Late Congruence) *Let p, q be processes with names in $name_{odd}$. A symmetric relation $\mathcal{R} \subseteq proc \times proc$ is a* weak** late bisimulation *if and only if, $\forall p, q : proc$, whenever $p\ \mathcal{R}\ q$ then, for all substitutions $\sigma : name \to name_{even}$,* injective *on $name_{even}$,:*

- $\forall \alpha : label.\ \forall p_1 : proc.\ p\sigma \stackrel{\alpha}{\Longmapsto}_* p_1 \to (\exists q_1 : proc.\ q\sigma \stackrel{\alpha}{\Longmapsto}_* q_1 \wedge p_1\ \mathcal{R}\ q_1)$,
- $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \stackrel{\lambda z:name.\bar{x}z}{\Longmapsto_*} p_1 \to$ $(\exists q_1 : name \to proc.\ q\sigma \stackrel{\lambda z:name.\bar{x}z}{\Longmapsto_*} q_1 \wedge \forall y : name.\ (y \notin fn(p_1), fn(q_1)) \to$ $p_1 y\ \mathcal{R}\ q_1 y))$, *and*
- $\forall x : name.\ \forall p_1 : name \to proc.\ p\sigma \stackrel{\lambda z:name.xz}{\Longmapsto_*} p_1 \to$ $(\exists q_1 : name \to proc.\ q\sigma \stackrel{\lambda z:name.xz}{\Longmapsto_*} q_1 \wedge \forall y : name\ p_1 y\ \mathcal{R}\ q_1 y)$.

Let p, q : proc. Then p is late weak** late bisimilar *to q if there exists a weak** late bisimulation $\mathcal{R} \subseteq proc \times proc$ such that $p\ \mathcal{R}\ q$. The union of all weak** late bisimulations, called* weak late congruence *(denoted by $\approx^{wlc}$) is a weak** late bisimulation itself.*

8.4 Final Semantics

In defining the functor and the syntactical coalgebra for the π-calculus, we cannot re-use directly the standard technique for process algebras of Chapter 5. In fact, in the definition of π-calculus bisimulations, when two bound outputs are compared, we require that the names which are used to instantiate the placeholder in the pair of target processes be fresh for both processes (conditions $y \notin fn(p_1), fn(q_1)$ in Definitions 8.3.1, 8.3.2, 8.3.5, 8.3.6, 8.3.7, 8.3.8, and in Lemma 8.3.4). This is unproblematic when dealing with pairs of processes. In the co-algebraic approach, however, the arrow f, corresponding to the syntactical coalgebra, should describe the transitions of each process independently from its possible partners. One cannot use simply, as names for the bound outputs of p, the names which do not appear in p, otherwise equivalent processes with different sets of free names would give rise to different sets of transitions, and hence would receive different final semantics.

The solution we propose consists in associating to processes sets of names (of possible partners). We work with pairs of the form (a, p), where p is a process and $a \in \mathcal{P}_{fin}(name)$ is an estimate of the finite set of free names currently available for interaction: i.e., $fn(p) \subseteq a$. When two processes p and q are compared, it is sufficient to choose $a \in \mathcal{P}_{fin}(name)$ so that $fn(p), fn(q) \subseteq a$ and to compare (a, p) and (a, q). This idea of associating the set of free names to a π-calculus process has been used also in [FMQ95, Qua96] to obtain a characterization of π-calculus based on ordinary transition systems and bisimulations.

In the sequel we fix a function $e : \mathcal{P}_{fin}(name) \to name$ such that, for all $a \subseteq name$, the value x_a of e on a is not in a. We shall denote by a^+ the finite set of names $a \cup \{x_a\}$.

The following functors will be used for giving final descriptions for the various notions of bisimulations of Section 8.3.

Definition 8.4.1 *Let $\mathcal{S}$ be the set of all substitutions $\sigma : name \to name$, and let $\mathcal{S}^-$ be the set of all substitutions $\sigma : name \to name_{even}$, which are injective on $name_{even}$.*

1. *Let $F : Class^*(U) \to Class^*(U)$ be the functor:*

   $$F(X) = \bigcup_{a \in \mathcal{P}_{fin}(name)} \{a\} \times (\mathcal{P}_{fin}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}) \cup \{\perp\}).$$

   The definition of F on arrows is canonical.

2. *Let $F_{\mathcal{S}} : Class^*(U) \to Class^*(U)$ be the functor:*

   $$F_{\mathcal{S}}(X) = \{\perp\} \cup (\bigcup_{a \in \mathcal{P}_{fin}(name)} \{a\} \times \mathcal{P}_{fin}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}))^{\mathcal{S}}.$$

   The definition of $F_{\mathcal{S}}$ on arrows is canonical.

3. *Let $F_{\mathcal{S}^-} : Class^*(U) \to Class^*(U)$ be the functor:*

   $$F_{\mathcal{S}^-}(X) = \\ \{\perp\} \cup (\textstyle\bigcup_{a \in \mathcal{P}_{fin}(name)} \{a\} \times \mathcal{P}_{fin}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}))^{\mathcal{S}^-}.$$

   The definition of $F_{\mathcal{S}^-}$ on arrows is canonical.

4. *Let $F^w : Class^*(U) \to Class^*(U)$ be the functor:*

   $$F^w(X) = \\ \textstyle\bigcup_{a \in \mathcal{P}_{fin}(name)} \{a\} \times (\mathcal{P}_{<2^{\aleph_0}}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}) \cup \{\perp\}).$$

   The definition of F^w on arrows is canonical.

5. *Let $F^w_{\mathcal{S}^-} : Class^*(U) \to Class^*(U)$ be the functor:*

   $$F^w_{\mathcal{S}^-}(X) = \\ \{\perp\} \cup (\textstyle\bigcup_{a \in \mathcal{P}_{fin}(name)} \{a\} \times \mathcal{P}_{<2^{\aleph_0}}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}))^{\mathcal{S}^-}.$$

   The definition of $F^w_{\mathcal{S}^-}$ on arrows is canonical.

In particular, the functor F above is used for modeling late and early bisimulations, the functor $F_{\mathcal{S}}$ is used for open bisimulation, the functor $F_{\mathcal{S}^-}$ for late (early) congruences, the functor F^w for weak late (early) bisimulation and weak late (early) ground congruence, and the functor $F^w_{\mathcal{S}^-}$ for weak late (early) congruence. Notice that the only difference between the functors for weak bisimulations and weak congruences and those for strong bisimulations and congruences lies in the powerset constructor. In fact, for weak bisimulations and congruences, we have to take into account the fact that weak processes can be infinitely branching. As already pointed out for the weak semantics of process algebras (see Chapter 5), this is not at all problematic in our purely set-theoretic setting, contrary to what happens in other semantical settings based on the notion of *continuity* (e.g. domain theory or complete metric spaces). Clearly much smaller categories than $Class^*(U)$ are sufficient to carry out our constructions. For instance, we could have used the category of *sets of hereditarily finite hypersets*, in dealing with strong bisimulations and congruences, and the category of *sets of hereditarily countable hypersets*, in dealing with weak bisimulations and congruences.

Proposition 8.4.2 *The functors F, $F_{\mathcal{S}}$, $F_{\mathcal{S}^-}$, F^w, and $F^w_{\mathcal{S}^-}$ have final coalgebras, denoted by (U_F, id_{U_F}), $(U_{F_{\mathcal{S}}}, id_{U_{F_{\mathcal{S}}}})$, $(U_{F_{\mathcal{S}^-}}, id_{U_{F_{\mathcal{S}^-}}})$, $(U_{F^w}, id_{U_{F^w}})$, and $(U_{F^w_{\mathcal{S}}}, id_{U_{F^w_{\mathcal{S}}}})$ respectively.*

8.4.1 Final Description of Late Bisimulation

.

Definition 8.4.3 *Let $(\mathcal{P}_{fin}(name) \times proc, f_l)$ be the F-coalgebra defined as follows:*

$$f_l(a,p) = \begin{cases} (a, \{(\alpha, (a, p_1)) \mid p \overset{\alpha}{\longmapsto} p_1\} \cup \\ \quad \{(\overline{y}x_a, (a^+, p_1 x_a)) \mid p \overset{\lambda z:name.\ \overline{y}z}{\longmapsto\!\!\!\!\!\rightarrow} p_1\} \cup \\ \quad \{\lambda x \in a^+. \\ \qquad \textit{if } x \in a \textit{ then } (yx, (a, p_1 x)) \\ \qquad\quad \textit{else } (yx_a, (a^+, p_1 x_a)) \mid p \overset{\lambda z:name.\ yz}{\longmapsto\!\!\!\!\!\rightarrow} p_1\}) & \textit{if } fn(p) \subseteq a \\ (a, \bot) & \textit{if } fn(p) \not\subseteq a \end{cases}$$

According to the definition of the functor F, the function f_l associates to each pair $(a,p) \in \mathcal{P}_{fin}(name) \times proc$ a pair (a, A). The component A is $\bot$, whenever a is not a correct estimate of the free names of p. Otherwise, $A \subseteq (label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}$ represents the transitions that the process p can perform. In the case of τ-actions, output, and bounded output, A is a subset of $\mathcal{P}_{fin}(name) \times proc$. In the case of input transitions, A is a function defined on a suitable finite set of names. This is arranged in such a way that all the names in a and the fresh name x_a are used as possible input values. In the case of bound output transitions, just name x_a is used. A single name, namely x_a, is chosen when a fresh name is required in a bound transition. This is not restrictive, since a single fresh name is sufficient for checking bisimulation equivalence. In this way, moreover, the function f_l maps each process (a, p) to a finite set A.

In the following lemma we characterize F-bisimulations on the coalgebra of processes defined above:

Lemma 8.4.4 *An F-bisimulation on the F-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_l)$ is a symmetric equivalence $\mathcal{R} \subseteq (\mathcal{P}_{fin}(name) \times proc) \times (\mathcal{P}_{fin}(name) \times proc)$, such that $(a, p)\ \mathcal{R}\ (a', q)$ implies $a = a'$ and one of the two conditions:*

- $fn(p) \not\subseteq a\ \wedge\ fn(q) \not\subseteq a$,
- $fn(p) \subseteq a\ \wedge\ fn(q) \subseteq a\ \wedge$ $[\forall \alpha : label.\ \forall p_1 : proc.\ p \overset{\alpha}{\longmapsto} p_1 \rightarrow (\exists q_1 : proc.\ q \overset{\alpha}{\longmapsto} q_1 \wedge (a, p_1)\ \mathcal{R}\ (a, q_1))] \wedge$ $[\forall x : name.\ \forall p_1 : name \rightarrow proc.\ p \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\to} p_1 \rightarrow (\exists q_1 : name \rightarrow proc.\ q \overset{\lambda z:name.xz}{\longmapsto\!\!\!\!\to} q_1 \wedge$ $\forall x \in a.\ (a, p_1 x)\ \mathcal{R}\ (a, q_1 x))] \wedge$ $[\forall \alpha : name \rightarrow label.\ \forall p_1 : name \rightarrow proc.\ p \overset{\alpha}{\longmapsto\!\!\!\!\to} p_1 \rightarrow (\exists q_1 : name \rightarrow proc.\ q \overset{\alpha}{\longmapsto\!\!\!\!\to} q_1 \wedge$ $(a^+, p_1 x_a)\ \mathcal{R}\ (a^+, q_1 x_a))]$.

Theorem 8.4.5 *Let $p, q \in proc$ be such that $fn(p), fn(q) \subseteq a$. Then $p \approx^l q$ if and only if there exists an F-bisimulation, $\mathcal{R}$, on the F-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_l)$ such that $(a, p)\ \mathcal{R}\ (a, q)$.*

The following proposition characterizes the greatest F-bisimulation by finality, and hence, by Theorem 8.4.5, also late bisimulation on π-calculus processes.

Proposition 8.4.6 *The equivalence induced by the unique morphism $\mathcal{M}^F_{(\mathcal{P}_{fin}(name) \times proc, f_l)} : (\mathcal{P}_{fin}(name) \times proc, f_l) \rightarrow (U_F, id_{U_F})$ coincides with the union, $\sim^F_{(\mathcal{P}_{fin}(name) \times proc, f_l)}$, of all F-bisimulations on the F-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_l)$.*

Another Final Description of Late Bisimulation

The functor F of Definition 8.4.1 is by no means the unique possible. Many other kinds of functors can be used just as well to provide a final account of strong bisimilarity. We chose such an F because we feel that it is simple and yet perspicuous.

By way of example we shall present an interesting alternative, which eliminates, at the price of an increase in complexity, the somewhat unpleasant fact that F yields only indirectly the semantics of a process. We use the notation of definitions 8.4.1 and 8.4.3.

Definition 8.4.7 *Let $G : Class^*(U) \rightarrow Class^*(U)$ be the functor:*

$$G(X) = [F(\mathcal{P}_{fin}(name) \times X)^{\mathcal{P}_{fin}(name)}]_{/\simeq}.$$

Where $[F(\mathcal{P}_{fin}(name) \times X)^{\mathcal{P}_{fin}(name)}]_{/\simeq}$ *denotes the set of equivalence classes of elements of* $F(\mathcal{P}_{fin}(name) \times X)^{\mathcal{P}_{fin}(name)}$ *modulo the equivalence:*

$$g \simeq g' \Longleftrightarrow \exists a \in \mathcal{P}_{fin}(name).\ \forall b \in \mathcal{P}_{fin}(name).\ a \subseteq b \Longrightarrow g(b) = g'(b).$$

The definition of G *on arrows is canonical.*

Definition 8.4.8 *Let* $(proc, g_l)$ *be the* G*-coalgebra defined as follows:*

$$g_l(p) = [\lambda a \in \mathcal{P}_{fin}(name).(a, f_l(a,p))]_{\simeq}.$$

Using standard arguments, one can show that the functor G is well-behaved and that the following proposition holds

Proposition 8.4.9 *Let* $p, q \in proc$. *Then*

$\mathcal{M}^G_{(proc,g_l)}(p) = \mathcal{M}^G_{(proc,g_l)}(q) \Longleftrightarrow$

$\exists a \in \mathcal{P}_{fin}(name).\ \mathcal{M}^F_{(\mathcal{P}_{fin}(name)\times proc, f_l)}((a,p)) = \mathcal{M}^F_{(\mathcal{P}_{fin}(name)\times proc, f_l)}((a,q))$,

where $\mathcal{M}^G_{(proc,g_l)}$ *is the unique map from the* G*-coalgebra* $(proc, g_l)$ *into the final* G*-coalgebra.*

8.4.2 Final Description of Early Bisimulation

In order to give a final description to early bisimulation, we endow π-calcolus processes with the following structure of F-coalgebra:

Definition 8.4.10 *Let* $(\mathcal{P}_{fin}(name) \times proc, f_e)$ *be the* F*-coalgebra defined as follows:*

$$f_e(a,p) = \begin{cases} (a, \{(\alpha, (a, p_1)) \mid p \overset{\alpha}{\longmapsto} p_1\} \cup & \\ \quad \{(\overline{y}x_a, (a^+, p_1 x_a)) \mid p \overset{\lambda z:name.\ \overline{y}z}{\longmapsto\!\!\!\!\!\rightarrow} p_1\}) & \text{if } fn(p) \subseteq a \\ (a, \bot) & \text{if } fn(p) \not\subseteq a \end{cases}$$

Similar results to those for late bisimulation hold also for the early case. In particular, also early bisimulation can receive an alternative final description via the functor G.

8.4.3 Final Description of Open Bisimulation

In order to give a final description to open bisimulation, we endow π-calculus processes with the following structure of $F_{\mathcal{S}}$-coalgebra:

Definition 8.4.11 *Let* $(\mathcal{P}_{fin}(name) \times proc, f_o)$ *be the* $F_{\mathcal{S}}$*-coalgebra defined as follows:*

$$f_o(a,p) = \begin{cases} \bot & \text{if } fn(p) \not\subseteq a \\ \lambda\sigma \in \mathcal{S}.\ f_l(\sigma^*(a), p\sigma) & \text{if } fn(p) \subseteq a \end{cases}$$

where $\sigma^*(a) = \{\sigma(x) \mid x \in a\}$.

Another Final Description of Open Bisimulation

There is a simple alternative to the functor $F_{\mathcal{S}}$ for giving the final description of the open bisimulation, i.e. the functor $G_{\mathcal{S}}$, which, like the functor G for late bisimulation, gives immediately the semantics of a process:

Definition 8.4.12 *Let $G_{\mathcal{S}} : Class^*(U) \to Class^*(U)$ be the functor:*

$G_{\mathcal{S}}(X) =$

$$\bigcup_{a\in\mathcal{P}_{fin}(name)} \{a\} \times (\mathcal{P}_{fin}((label_{a^+} \times X) \cup (label_{a^+} \times X)^{a^+}) \cup \{\perp\})]^{\mathcal{P}_{fin}(name)\times\mathcal{S}} \; .$$

The definition of $G_{\mathcal{S}}$ on arrows is canonical.

Definition 8.4.13 *Let $(proc, g_o)$ be the $G_{\mathcal{S}}$-coalgebra defined as follows:*

$$g_o(p) = \lambda(a,\sigma) \in \mathcal{P}_{fin}(name) \times \mathcal{S}.\ \mathit{if}\ (cod(\sigma) \subseteq a)\ \mathit{then}\ f_l(a, p\sigma)\ \mathit{else}\ \perp \;,$$

where $cod(\sigma)$ denotes the codomain of σ.

Proposition 8.4.14 *Let $p, q \in proc$. Then*

$$p \approx^e q \iff \mathcal{M}^{G_{\mathcal{S}}}_{(proc,g_o)}(p) = \mathcal{M}^{G_{\mathcal{S}}}_{(proc,g_o)}(q) \;,$$

where $\mathcal{M}^{G_{\mathcal{S}}}_{(proc,g_o)}$ is the unique map from the $G_{\mathcal{S}}$-coalgebra $(proc, g_o)$ into the final $G_{\mathcal{S}}$-coalgebra.

8.4.4 Final Description of Late Congruence

We assume that for all $a \in \mathcal{P}_{fin}(name)$, $x_a \in name_{even}$.

Definition 8.4.15 *Let $(\mathcal{P}_{fin}(name) \times proc, f_{lc})$ be the $F_{\mathcal{S}^-}$-coalgebra defined as follows:*

$$f_{lc}(a,p) = \begin{cases} \perp & \mathit{if}\ fn(p) \not\subseteq a \\ \lambda\sigma \in \mathcal{S}^-.\ f(\sigma^*(a), p\sigma) & \mathit{if}\ fn(p) \subseteq a \end{cases}$$

where $\sigma^*(a) = \{\sigma(x) \mid x \in a\}$.

Proposition 8.4.16 *Let $p, q \in proc$ be such that $fn(p), fn(q) \subseteq a \cap name_{odd}$, and let $\mathcal{M}^{F_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{lc})}$ be the map from the $F_{\mathcal{S}^-}$-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_{lc})$ into the final $F_{\mathcal{S}^-}$-coalgebra. Then*

$$p \approx_{lc} q \;\Leftrightarrow\; \mathcal{M}^{F_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{lc})}((a,p)) = \mathcal{M}^{F_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{lc})}((a,q)).$$

8.4.5 Final Description of Weak Late Bisimulation

Definition 8.4.17 *Let $(\mathcal{P}_{fin}(name) \times proc, f_{wl})$ be the F^w-coalgebra defined as follows:*

$$f_{wl}(a,p) = \begin{cases} (a, \{(\alpha,(a,p_1)) \mid p \overset{\alpha}{\Longmapsto} p_1\} \cup \\ \quad \{(\overline{y}x_a,(a^+,p_1x_a)) \mid p \overset{\lambda z:name.\ \overline{y}z}{\Longmapsto} p_1) \cup \\ \quad \{\lambda x : a^+. \\ \qquad \textit{if } x \in a \textit{ then } (yx_i,(a,p_1x_i)) \\ \qquad\quad \textit{else } (yx_a,(a^+,p_1x_a)) \mid p \overset{\lambda z:name.\ yz}{\Longmapsto} p_1\}) & \textit{if } fn(p) \subseteq a \\ (n,\perp) & \textit{if } fn(p) \not\subseteq a \end{cases}$$

Theorem 8.4.18 *Let $p,q \in proc$ be such that $fn(p), fn(q) \subseteq a$. Then $p \approx^{wl} q$ if and only if there exists an F^w-bisimulation on the F^w-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_{wl})$ such that $(a,p)\ \mathcal{R}\ (a,q)$.*

8.4.6 Final Description of Weak Late Ground Congruence

Definition 8.4.19 *Let $(\mathcal{P}_{fin}(name) \times proc, f_{wlgc})$ be the F^w-coalgebra defined as follows:*

$$f_{wlgc}(a,p) = \begin{cases} (a, \{(\alpha,(a,p_1)) \mid p \overset{\alpha}{\Longmapsto}_* p_1\} \cup \\ \quad \{(\overline{y}x_a,(a^+,p_1x_a)) \mid p \overset{\lambda z:name.\ \overline{y}z}{\Longmapsto}_* p_1) \cup \\ \quad \{\lambda x : a^+. \\ \qquad \textit{if } x \in a \textit{ then } (yx_i,(a,p_1x_i)) \\ \qquad\quad \textit{else } (yx_a,(a^+,p_1x_a)) \mid p \overset{\lambda z:name.\ yz}{\Longmapsto}_* p_1\}) & \textit{if } fn(p) \subseteq a \\ (n,\perp) & \textit{if } fn(p) \not\subseteq a \end{cases}$$

8.4.7 Final Description of Weak Late Congruence

We assume that for all $a \in \mathcal{P}_{fin}(name)$, $x_a \in name_{even}$.

Definition 8.4.20 *Let $(\mathcal{P}_{fin}(name) \times proc, f_{wlc})$ be the $F^w_{\mathcal{S}^-}$-coalgebra defined as follows:*

$$f_{wlc}(a,p) = \begin{cases} \perp & \textit{if } fn(p) \not\subseteq a \\ \lambda\sigma \in \mathcal{S}^-.\ f_{wlgc}(\sigma^*(a), p\sigma) & \textit{if } fn(p) \subseteq a \end{cases}$$

where $\sigma^*(a) = \{\sigma(x) \mid x \in a\}$.

Proposition 8.4.21 *Let $p,q \in proc$ such that $fn(p), fn(q) \subseteq a \cap name_{odd}$ and let $\mathcal{M}^{F^w_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{wlc})}$ be the map from the $F^w_{\mathcal{S}^-}$-coalgebra $(\mathcal{P}_{fin}(name) \times proc, f_{wlc})$ into the final $F^w_{\mathcal{S}^-}$-coalgebra. Then*

$$p \approx_{wlc} q \;\Leftrightarrow\; \mathcal{M}^{F^w_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{wlc})}((a,p)) = \mathcal{M}^{F^w_{\mathcal{S}^-}}_{(\mathcal{P}_{fin}(name)\times proc, f_{wlc})}((a,q)).$$

8.5 Applications

As an application of the final semantics apparatus, that we have erected so far, we give a semantical proof of the law $p|q \sim q|p$. Of course this proof ultimately corresponds to the usual proof by coinduction, but the machinery of final semantics allows to express it in a more structured way. In our setting this amounts to showing that $\forall a \in \mathcal{P}_{fin}(name).\ \mathcal{M}_F(\langle a, p|q\rangle) = \mathcal{M}_F(\langle a, q|p\rangle)$. To this end, because of the particular nature of the π-calculus, we will have to show more in general that

$$\begin{array}{ll}\forall a \in \mathcal{P}_{fin}(name). & \mathcal{M}_F(\langle a, \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n|qz_1 \cdots z_n)\cdots))\rangle) = \\ & \mathcal{M}_F(\langle a, \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ qz_1 \cdots z_n|pz_1 \cdots z_n)\cdots))\rangle)\end{array}$$

holds for all naturals $n \in \mathbf{N}$ and $p, q \in name^n \to proc$, where $name^0 \to proc = proc$ and $name^{n+1} \to proc = (name^n \to proc)^{name}$. In the case $n = 0$ this reduces to $\forall a \in \mathcal{P}_{fin}(name).\ \mathcal{M}_F(\langle a, p|q\rangle) = \mathcal{M}_F(\langle a, q|p\rangle)$.

We will achieve our goal exploiting the finality of (X_F, ι_F) among F-coalgebras. First we introduce a suitable F-coalgebra (B, β), and two functions $g_1, g_2 : B \to \mathcal{P}_{fin}(name) \times proc$, such that for suitable elements $u_{a,n,p,q}$ in B,
$g_1(u_{a,n,p,q}) = \langle a, \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n|qz_1 \cdots z_n)\cdots))\rangle$ while
$g_2(u_{a,n,p,q}) = \langle a, \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ qz_1 \cdots z_n|pz_1 \cdots z_n)\cdots))\rangle$.

We show then that both $\mathcal{M}_F \circ g_1$ and $\mathcal{M}_F \circ g_2$ are F-coalgebra morphisms from (B, β) to the final F-coalgebra (X_F, ι_F). Therefore they coincide by finality.

Here are the details. Let:

$$\begin{array}{l}B = \mathcal{P}_{fin}(name) \times (\bigcup_{n\in\mathbf{N}}\{n\} \times (name^n \to proc) \times (name^n \to proc));\\ g_1(\langle a, \langle n, p, q\rangle\rangle) = \langle a, \nu(\lambda z_1(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n|qz_1 \cdots z_n)\cdots))\rangle;\\ g_2(\langle a, \langle n, p, q\rangle\rangle) = \langle a, \nu(\lambda z_1(\cdots \nu(\lambda z_n.\ qz_1 \cdots z_n|pz_1 \cdots z_n)\cdots))\rangle.\end{array}$$

The definition of β is rather involved, since it has to take into account the different kinds of transitions:

$$\beta(\langle a, \langle n, p, q\rangle\rangle) = \begin{cases} \langle a, \perp\rangle & \text{if } fn(p), fn(q) \not\subseteq a \\ \langle a, A_{par} \cup A_{open} \cup A_{close}\rangle & \text{if } fn(p), fn(q) \subseteq a \end{cases}$$

where:

$$\begin{array}{rcl} A_{par} & = & \{\langle \alpha, \langle a, \langle n, r, s\rangle\rangle\rangle \mid \\ & & \quad \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n|qz_1 \cdots z_n)\cdots)) \stackrel{\alpha}{\longmapsto} \\ & & \quad \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ rz_1 \cdots z_n|sz_1 \cdots z_n)\cdots))\} \\ & \cup & \{\langle \bar{y}x_a, \langle a^+, \langle n, rx_a, sx_a\rangle\rangle\rangle \mid \\ & & \quad \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n|qz_1 \cdots z_n)\cdots)) \stackrel{\lambda x.\bar{y}x}{\longmapsto} \\ & & \quad \lambda x.\nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ rxz_1 \cdots z_n|sxz_1 \cdots z_n)\cdots))\} \\ & \cup & \{\lambda x \in a^+.\ \text{if } x \in a \text{ then } \langle yx, \langle a, \langle n, rx, sx\rangle\rangle\rangle \\ & & \quad \text{else } \langle yx_a, \langle a^+, \langle n, rx_a, sx_a\rangle\rangle\rangle \mid \end{array}$$

$$\begin{array}{rcl}
 & & \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n | qz_1 \cdots z_n)\cdots)) \overset{\lambda x.yx}{\longmapsto\!\!\!\!\rightarrow} \\
 & & \lambda x.\nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ rxz_1 \cdots z_n | sxz_1 \cdots z_n)\cdots))\}, \\
A_{open} & = & \{\langle \bar{y}x_a, \langle a^+, \langle n-1, rx_a, sx_a\rangle\rangle\rangle \mid \\
 & & \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n | qz_1 \cdots z_n)\cdots)) \overset{\lambda x:name.\ \bar{y}x}{\longmapsto\!\!\!\!\rightarrow} \\
 & & \quad \lambda x.\nu(\lambda z_1.(\cdots \nu(\lambda z_{n-1}.\ rxz_1 \cdots z_{n-1} | sxz_1 \cdots z_{n-1})\cdots))\}, \\
A_{close} & = & \{\langle \tau, \langle a, \langle n+1, r, s\rangle\rangle\rangle \mid \\
 & & \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\ pz_1 \cdots z_n | qz_1 \cdots z_n)\cdots)) \overset{\tau}{\longmapsto} \\
 & & \quad \nu(\lambda z_1.(\cdots \nu(\lambda z_n.\nu(\lambda x.\ rz_1 \cdots z_n x | sz_1 \cdots z_n x))\cdots))\}.
\end{array}$$

It is interesting to point out that the categorical methodology used above, i.e. defining suitable coalgebras and exploiting uniqueness of the final morphism, can be uniformly applied to other coinductive proofs of properties of processes. This methodology is also quite general. For instance it could be used to define coinductively a semantical operator of parallel composition.

Chapter 9

Concluding Remarks

In this thesis we think that we have shown a possible way of understanding, and hence manipulating and reasoning rigorously on, those infinite and circular objects and those recursively defined notions, which arise in computation theory through some kind of *maximal fixed point* definition. Classical examples of such objects are elements of *lazy* (*coinductive*) types, e.g. *streams* and *processes*. Classical examples of such notions are *observational equivalences* of computational objects exhibiting infinite behaviours, such as *untyped functions* and *processes*.

In Part I we have outlined how to utilize and make sense of maximal fixed points from three different perspectives: the set-theoretical, the categorical and the logical. We started, in Chapter 2, from the purely set-theoretical account of *coinduction principles* and *coinductive functions*. We gave a categorical generalization of this account in Chapter 3 in terms of *final coalgebras*, building on the work of Rutten and Turi. And finally, in Chapter 4, we discussed briefly possible logical axiomatizations of largest bisimulation equivalences on syntactical objects.

This theoretical investigation was put to use in Part II of the thesis, where we illustrated the broad applicability of *final semantics* to a wide range of programming languages. Here we developed in depth both syntactical and semantical techniques for giving coinductive descriptions of observational (contextual) equivalences, thereby providing a number of (often new) coinduction principles. We have mainly utilized categories of non-wellfounded sets. This part of the thesis can be viewed as an illustration of the generality of this mathematically elementary approach to semantics.

Our work, however, is far from having reached a conclusion.

We have left a number of challenging open problems. We recall some of the most intriguing ones:

- how to incorporate into the final semantics paradigm mixed functors à la Freyd ([Fre90, Fre92]);

- how to derive purely syntactical mixed induction coinduction principles, see Section 2.1;

- characterize those lambda theories which admit an applicative coinduction principle, see Section 7.5;

- investigate the problem of defining semantical counterparts to higher order syntactic operators on the final coalgebras used in Chapters 5, 6, 7, 8; more in general, develop general conditions for establishing compositionality of the final semantics, along the lines of [Tur96], which can encompass also such complex functors as those which we use in Part II.

In more than one occasion we have only developed specific examples where, instead, it would have been nice to have a general theory. For instance, in Chapter 4 we considered only regular guarded non-deterministic processes and regular binary trees, rather than a larger class of coinductive types. In Section 3.4, we presented a procedure for deriving set-theoretical coinduction principles from categorical ones for a particular algebra of functors. A more general characterization of when it is the case that a given set-theoretical coinduction principle can be construed categorically, would be welcome here.

On other occasions we have only scratched the surface of a given problem and only hinted to a possible general theory, without developing it to a satisfactory extent. For example, in Section 2.1 we have discussed many generalized forms of set-theoretical coinduction, but a theory of generalized coinductive schemata must be underlying these instances. In Section 3.3 we gave categorical accounts of some forms of set-theoretical coinduction principles, but we did not cover all the diversity of set-theoretical examples that we have mentioned in Chapter 2.

Many of the themes that we have dealt with in this thesis have been considered from different perspectives also by other authors in the literature.

Here is a list of related work, presenting approaches alternative to the ones we have explored, which would be extremely interesting to compare to ours. The classification is done by topic:

- Barwise and Moss ([BM96]) presented an approach to circular objects based on hypersets, developing some sort of "category-free" final semantics;

- Pitts ([Pit96a]), and Hermida and Jacobs ([HJ95]) presented categorical accounts of coinduction and coinductive types, based on independent categorical understandings of relations and relational structures;

- Montanari and coauthors ([DMV90, MS89]) introduced independent set-theoretical descriptions of bisimulations, in connection with the problem of finding canonical representatives for bisimulation equivalences; Joyal, Nielsen and Winskel ([JNW94]) introduced a categorical definition of bisimulation based on open maps;

- there exist the traditional domain-based approaches to coinductive types based on partial orders ([Plo85]) or metric spaces ([Bre97a]). But there exist also new frameworks for reasoning on infinite objects, such as *hyperuniverses* ([FH96, FHL94, FHL95]), which try to combine metric spaces and hypersets;

- there exists at lest two well established formal approaches to infinite and circular objects in intuitionistic type theories. The first, based on approximations, is due to Constable and coworkers, ([MPC86]); the second and more general, due to Coquand and Giménez [Coq94, Gim94, Gim95], is based on infinitely regressing proofs and the *guarded induction principle*.

One should try to carry out comparisons also with those alternative approaches to infinite or circular computational objects which are seemingly unrelated to ours. Here we have in mind Martin-Löf's constructive mathematics of infinity [Mar88], and possible "non-standard" approaches to coinductive types in the style of "non-standard" analysis, where actual infinites can be manipulated. Some work in this latter direction seems to have been done by Goto [Got87].

We should also consider recent advances in the theory of Final Semantics, e.g [Jac97, RV97, Tur97], so as recent uses of coalgebras in other contexts. Coalgebras have been used in the specification of (object-based) systems ([Rei95, Jac96, Cor97]), and, very recently, in the world of Goguen's hidden algebras ([Cir?]).

It would be also extremely important to explore to what extent and how fruitfully, other work based on coinduction, e.g. [CG94, RP95, Gor94, Gor95], could be rephrased in our setting.

Certainly we should also experiment more with the coinduction principles that we have introduced, in formal development environments such as COQ, [CCF95]. Some work along these lines appears in [HMS98]. But there are challenging open problems on operational equivalences of programs, in various programming languages, which wait to be addressed formally, see e.g. [HL95].

Finally, we feel that we are in the position of drawing some general considerations on what Final Semantics can do for us, or it seems not to be able to do cope with, yet.

Certainly, we have presented quite a broad spectrum of languages which can receive *interesting*, *fully abstract* final semantics: first and higher order process algebras, λ-calculi, π-calculi. And this often in many alternative ways. Final Semantics seems, indeed, a rather general technique for providing semantics. Probably some other important examples should still need to be considered: e.g. Algol-like languages or functional languages extended with primitives for manipulating effects (see [CG94, Gor94]), and/or parallel features (see e.g. [AMST93]). But we feel that an observational equivalence should be quite perverse, not to be ultimately characterizable coinductively with the machinery of Final Semantics.

On the other hand, Final Semantics seems to be rather weak on the side of the computational or program logics that it is capable of suggesting. Domain theoretic approaches, both in the ordered (see e.g. [GS90]) and in the metric case (see e.g. [BZ82, AM89]), provide naturally a notion of computable property (e.g. Scott open, compact subset), which one can use as the basis of an exogenous program logic. On the other hand, in the elementary set-theoretical setting that we have worked in, no class of properties naturally stands out. Final Semantics is a precious methodological tool, which can provide us with a principled way of finding bisimulations for the coinduction principle, which lies at the basis of the Final Semantics itself. Summing up, in our view Final Semantics using hypersets is one of the most elementary general frameworks for making sense of a wide variety of constructions and proof principles on infinite and circular objects. Final Semantics, however, does not improve substantially our understanding of the ultimately mysterious nature of infinity.

Bibliography

[Abr91] S.Abramsky. Domain Theory in logical form, *Annals of Pure and Applied Logic* **51**, 1991, 1–77.

[AO93] S.Abramsky, L.Ong. Full Abstraction in the Lazy Lambda Calculus, *Information and Computation* **105**(2), 1993, 159–267.

[Acz88] P.Aczel. *Non-well-founded sets*, CSLI Lecture Notes **14**, Stanford 1988.

[Acz93] P.Aczel. *Final Universes of Processes*, *MFPS'93* Conference Proceedings, Brookes et al. eds., Springer LNCS **802**, New Orleans 1993.

[AM89] P.Aczel, N.Mendler. A Final Coalgebra Theorem, in *Category Theory and Computer Science*, D.H.Pitt et al. eds., Springer LNCS **389**, 1989, 357–365.

[AMST93] G.Agha, I.Mason, S.Smith, C.Talcott. A Foundation for Actor Computation, *J. Functional Programming* **1** (1), 1993, 1– 68.

[AC93] R.Amadio, L.Cardelli. *Subtyping recursive types*, *ACM Transactions on Programming Languages and Systems (TOPLAS)* **15**(4), 1993, 575–631.

[AR89] P.America, J.J.M.M.Rutten. Solving domain equations in a category of complete metric spaces, *J. Comp. Sys. Sci.* **39**, 1989, 343–375.

[BB93] J.de Bakker, F.van Breugel. Topological models for higher order control flow, *MFPS'93* Conference Proceedings, Brookes et al. eds., Springer LNCS **802**, New Orleans 1993, 122–142.

[BB9?] J.de Bakker, F.van Breugel. From Banach to Milner: Metric Semantics for Second Order Communication and Concurrency, to appear in G.Plotkin, C.Stirling, M.Tofte, eds., *Language and Metalanguage*, Foundations of Computing Series, The MIT Press.

[BV96] J.de Bakker, E.de Vink. *Control Flow Semantics*, Foundations of Computing Series, The MIT Press, Cambridge, 1996.

[BZ82] J.W.de Bakker, J.I.Zucker. Processes and denotational semantics of concurrency, *Info. Contr.* **54**, 1982, 70–120.

[Bar84] H.Barendregt. *The Lambda Calculus, its Syntax and Semantics,* North Holland, Amsterdam, 1984.

[BCD83] H.Barendregt, M.Coppo, M.Dezani-Ciancaglini. A filter lambda-model and the completeness of the type assignment, *J. of Symbolic Logic* **48**(4), 1983, 931–940.

[Bar93] M.Barr. Terminal coalgebras in well-founded set theory, *TCS* **114**, 1993, 299–315.

[Bar94] M.Barr. Additions and corrections to "Terminal coalgebras in well-founded set theory", *TCS* **124**, 1994, 189–192.

[Bar91] J.Barwise. Hypersets, *The Mathematical Intelligencer* **13**(4), 1991, 31–41.

[BM96] J.Barwise, L.Moss. *Vicious circle: On the mathematics of non-wellfounded phenomena*, CSLI Publications, Stanford, 1996.

[BIM88] B.Bloom, S.Istrail, A.Meyer. *Bisimulation can't be traced: Preliminary report*, *POPL'88* Conference Proceedings, 1988, 229–239.

[BH97] M.Brandt, F.Henglein. *Coinductive axiomatization of recursive type equality and subtyping*, *TLCA'97* Conference Proceedings, P.de Groote, R.Hindley, eds., Springer LNCS **1210**, 1997, 63–81.

[Bre94] F.van Breugel. *Topological Models in Comparative Semantics*, PhD thesis, Vrije University, Amsterdam 1994.

[Bre97] F.van Breugel. *Terminal Metric Spaces of Finitely Branching and Image Finite Linear Processes*, Draft, July 1997.

[Bre97a] F.van Breugel. Comparative Metric Semantics for Programming Languages, Draft, 1997.

[Cir?] C.Cirstea. *Coalgebra semantics for hidden algebras: parametrized objects and inheritance*, Proc. of the 13th *WADT* Workshop, Springer LNCS, to appear.

[CDHL82] M.Coppo, M.Dezani, F.Honsell, G.Longo. Extended Type Structures and Filter Lambda Models, *Logic Colloquium'82* Conf. Proc., G.Longo et al. eds., North Holland, 1983.

[CDZ87] M.Coppo, M.Dezani, M.Zacchi. Type Theories, Normal Forms and D_∞-Lambda-Models, *Information and Computation* **72**(2), 1987, 85–116.

[Coq94] T.Coquand. Infinite Objects in Type Theory, Types for Proofs and Programs *TYPES-93*, Springer LNCS **806**, 1994, 62–78.

[Cor97] A.Corradini. A complete calculus for equational deduction in coalgebraic specification, Technical Report **SEN-R9723**, CWI, Amsterdam, 1997.

[CCF95] C.Cornes *et al.*. The Coq Proof Assistant Reference Manual, Version 5.10 INRIA, Rocquencourt, 1995.

[CG94] R.Crole, A.D.Gordon. A sound metalogical semantics for input/output effects, *CSL'94* Conference Proceedings, Springer LNCS, 1994.

[DMV90] R.De Nicola, U.Montanari, F.Vaandrager. Back and forth bisimulations, Report **CS-R9021**, CWI, 1990.

[EHR92] L.Egidi, F.Honsell, S.Ronchi Della Rocca. Operational, denotational and logical descriptions: a case study, *Fundamenta Informaticae* **16**(2), 1992, 149–169.

[Fef90] S.Feferman. Polymorphic typed lambda-calculi in a type-free axiomatic framework, *Contemporary Mathematics* **106**, 1990, 101–136.

[FMQ95] G.-L.Ferrari, U.Montanari, P.Quaglia. The weak late π-calculus semantics as observation equivalence, *CONCUR'95*, Springer LNCS **962**, Berlin 1995.

[Fio96] M.Fiore. A Coinduction Principle for Recursive Data Types Based on Bisimulation, *Information and Computation* **127**, 1996, 186–198.

[FMS96] M.Fiore, E.Moggi, D.Sangiorgi. A fully-abstract model for the π-calculus, *LICS'96* Conference Proc., IEEE, 1996.

[FH83] M.Forti, F.Honsell. Set theory with free construction principles, *Ann. Scuola Norm. Sup. Pisa*, Cl. Sci. (4) **10**, 1983, 493– 522.

[FH96] M.Forti, F.Honsell. A general construction of Hyperuniverses, *TCS* **156**, 1996.

[FHL94] M.Forti, F.Honsell, M.Lenisa. Processes and Hyperuniverses, *MFCS'94* Conference Proceedings, I.Prívara et al. eds., Springer LNCS **841**, Berlin 1994, 352–361.

[FHL95] M.Forti, F.Honsell, M.Lenisa. Axiomatic Characterizations of Hyperuniverses and Applications, *Annals of the New York Academy of Sciences* **80**, Papers on General Topology and Applications, Eleventh Summer Conference at the University of Southern Maine, Gorham (Maine), New York 1996, 140–163.

[Fre90] P.Freyd. Recursive types reduced to inductive types, 5th *LICS* Conference Proceedings, IEEE Computer Society Press, 1990, 498–505.

[Fre92] P.Freyd. Algebraically complete categories, *Category Theory '90*, A.Carboni et al. eds, Springer LNM **1144**, Como 1992, 95–104.

[FS90] P.Freyd, A.Scedrov. *Categories, Allegories*, North-Holland, Amsterdam, 1990.

[Geu91] H.Geuvers. Inductive and Coinductive types with Iteration and recursion, Draft, 1991.

[Gim94] E.Giménez. Codifying guarded definitions with recursive schemes, Technical Report **95-07**, Ecole Normale Sup. de Lyon, December 1994.

[Gim95] E.Giménez. Implementation of co-inductive types in Coq: An Experiment with the Alternating Bit Protocol, *BRA* Workshop Types for Proofs and Programs, Turin, June 1995.

[Gla90] R.J.van Glabbeek. The linear time – branching time spectrum, *CONCUR'90* Conference Proceedings, J.C.M.Baeten et al eds., Springer LNCS **458**, 1993, 278–297.

[GW89] R.J.van Glabbeek, W.P.Weijland. Branching Time and Abstraction in Bisimulation Semantics, *IFIP'89* Conference Proceedings, G.X.Ritter ed, Elsevier Publishers B.V. (North Holland), 1989, 613–618.

[Gor94] A.D.Gordon. *Functional Programming and Input/Output*, Cambridge University Press 1994.

[Gor95] A.D.Gordon. Bisimilarity as a Theory of Functional Programming, to appear in *MFPS'95* Conference Proceedings, Electronic Notes in Computer Science **1**, 1995.

[Got87] S.Goto. Non-standard normalization, US-Japan Workshop, Honolulu, May 1987.

[GV92] J.F.Groote, F.Vaandrager. Structured Operational Semantics and Bisimulation as a Congruence, *Information and Computation* **100**, 1992, 202–260.

[GS90] C.Gunter, D.Scott. Semantic Domains, *Handbook of Theoretical Computer Science,* J.van Leeuwen ed., Elsevier Science Publishers B.V., 1990, 634–674.

[HHP93] R.Harper, F.Honsell, G.Plotkin. A framework for defining logics, *JACM* **40**(1), 1993, 143–184.

[HJ95] C.Hermida, B.Jacobs. Induction and Coinduction via Subset Types and Quotient Types, Proceedings of the Joint *CLICS-TYPES*, Workshop on Categories and Type Theory, P.Dybjer, R. Pollack, eds., Report **85**, Göteborg University, May1995.

[Hin80] R.Hinnion. Contraction de structures et application à NFU, *C.R. Acad. Sc.*, Paris, T.**290**(80) Sèr.A, 677–680, 1980.

[HL93] F.Honsell, M.Lenisa. Some Results on Restricted λ-calculi, *MFCS'93* Conference Proceedings, A.Borzyszkowski et al. eds., Springer LNCS **711**, 1993, 84–104.

[HL95] F.Honsell, M.Lenisa. Final Semantics for Untyped Lambda Calculus, *TLCA'95* Conf. Proc., M.Dezani, G.Plotkin eds., Springer LNCS **902**, Berlin 1995, 249–265.

[HL9?] F.Honsell, M.Lenisa. Semantical Analysis of Perpetual Strategies in λ-calculus, to appear in *TCS*.

[HL98] F.Honsell, M.Lenisa. A New Coinduction Principle for λ-calculi, Draft, January 1998.

[HMS98] F.Honsell, M.Miculan, I.Scagnetto. Encoding Process Calculi in Constructive Type Theory, Draft, January 1998.

[HLMP98] F.Honsell, M.Lenisa, U.Montanari, M.Pistore. Final Semantics for the π-calculus, to appear in Proc. of *PROCOMET'98*.

[HR92] F.Honsell, S.Ronchi Della Rocca. An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus, *J. of Computer and System Sciences* **45**(1), 1992, 49–75.

[How89] D.Howe. Equality in Lazy Computation Systems, 4th *LICS* Conference Proceedings, IEEE Computer Society Press, 1989, 198–203.

[How96] D.Howe. Proving Congruence of Bisimulation in Functional Programming Languages, *Information and Computation* **124**(2), 1996, 103–112.

[Hyl75] M.Hyland. A survey of some useful partial order relations on terms of the lambda-calculus, C.Böhm ed., Springer LNCS **37**, Berlin 1975, 83–95.

[Jac96] B.Jacobs. Objects and Classes, co-algebraically, *Object-Orientation with Parallelism and Book Persistence*, B.Freitag et al. eds., Kluwer Academic Publishers, 1996.

[Jac97] B.Jacobs. Behaviour-refinement of object-oriented specifications with coinductive correctness proofs, *TAPSOFT'97*, M.Bidoit, et. al. eds., Springer LNCS **1214**, 1997, 787–802.

[JNW94] A.Joyal, M.Nielsen, G. Winskel. Bisimulation from Open Maps, BRICS Report **RS-94-7**, 1994.

[Len96] M.Lenisa. Final Semantics for a Higher Order Concurrent Language, *CAAP'96*, H.Kirchner et. al. eds., Springer LNCS **1059**, 1996, 102–118.

[Len97] M.Lenisa. Semantic Techniques for Deriving Coinductive Characterizations of Observational Equivalences for λ-calculi, *TLCA'97* Conference Proceedings, P.de Groote, R.Hindley, eds., Springer LNCS **1210**, 1997, 63–81.

[Len97a] M.Lenisa. A Uniform Syntactical Method for Proving Coinduction Principles in λ-calculi, *TAPSOFT'97*, M.Bidoit, et. al. eds., Springer LNCS **1214**, Berlin 1997, 248–266.

[MST?] I.Mason, S.Smith, C.Talcott. From Operational Semantics to Domain Theory, *Information and Computation* to appear.

[Mar84] P.Martin-Löf. *Intuitionistic Type Theory*, Bibliopolis, 1984.

[Mar88] P.Martin-Löf. Mathematics of Infinity, in *COLOG-88* (P.Martin-Löf et al. eds.), Springer LNCS **417**, Berlin 1988.

[MPC86] N.P.Mendler, P.Panangaden, R.L.Constable. Infinite Objects in Type Theory, 1th *LICS* Conference Proceedings, IEEE Computer Society Press, 1986, 249–255.

[Mil83] R.Milner. Calculi for synchrony and asynchrony, *TCS* **25**, 1983, 267–310.

[Mil84] R.Milner. A complete inference system for a class of regular behaviours, *J. of Computer and System Sciences* **28**, 1984, 39–466.

[Mil89] R.Milner. Operational and Algebraic Semantics of Concurrent Processes, *Handbook of Theoretical Computer Science*, J. van Leeuwen ed., 1990, 1203–1242.

[Mil93] R.Milner. The polyadic π-calculus: a tutorial, *Logic and Algebra of Specification*, NATO ASI Series F **94**, Springer Verlag, 1993.

[MPW92] R.Milner, J.Parrow, D.Walker. A calculus of mobile processes, *Inf. and Comp.* **100**, 1992, 1–77.

bibitem[MPW93]MPW93 R.Milner, J.Parrow, D.Walker. Modal logic for mobile processes, *TCS* **114**, 1993, 149–171.

[MS92] U.Montanari, V.Sassone. Dynamic Congruence vs Progressing Bisimulation for CCS, *Fundamenta Informaticae* **16**(2), 1992, 171–196.

[MS89] U.Montanari, M.Sgamma. Canonical Representatives for Observational Equivalences Classes, *Resolution of Equations in Algebraic Structures* **1**, 1989, 293–319.

[Par81] D.M.R.Park. Concurrency and automata on infinite sequences, *MFPS'93*, P.Deussen ed., Springer LNCS **104**, 1981, 167–183.

[Pau93] L.C.Paulson. Co-induction and co-recursion in higher-order logic, Technical Report **304**, University of Cambridge Computer Laboratory, 1993.

[Per91] P.Pérez. An extensional partial combinatory algebra based on λ-terms, *MFCS'91*, Springer LNCS **520**, 1996.

[Pit94] A.M.Pitts. Computational Adequacy via 'Mixed' Inductive Definitions, *MFPS'93*, Brookes et al. eds., Springer LNCS **802**, 1994, 72–82.

[Pit95] A.M.Pitts. Operationally-Based Theories of Program Equivalence, notes for the Summer School on *Semantics and Logics of Computation*, Cambridge, 25–29 September 1995.

[Pit96] A.M.Pitts. *A Note on Logical Relations Between Semantics and Syntax*, *L. J. of the IGPL* **5**(4), 1997, 589–601.

[Pit96a] A.M.Pitts. Relational Properties of Domains, *Information and Computation* **127**, 1996, 66–90.

[Plo77] G.Plotkin. Call-by-name, call-by-value and the lambda calculus, *TCS* **1**, 1977, 125–159.

[Plo85] G.Plotkin. Types and Partial Functions, Post-Graduate Lecture Notes, Department of Computer Science, University of Edinburgh, 1985.

[Qua96] P.Quaglia. The π-calculus with explicit substitutions, PhD. thesis **TD-9/99**, University of Pisa, 1996.

[Rei95] H.Reichel. An approach to object semantics based on terminal coalgebras, *Mathematical Structure in Computer Science* **5**, 1995, 129–152.

[RP95] E.Ritter, A.M.Pitts. A fully abstract translation between a λ-calculus with reference types and Standard ML, *TLCA '95*, M.Dezani, G.Plotkin eds., Springer LNCS **902**, Berlin 1995, 397–413.

[Rut92] J.J.M.M.Rutten. Processes as terms: non-wellfounded models for bisimulation, Math. Struct. Comp. Sci. **2**(3), 1992, 257–275.

[Rut96] J.J.M.M.Rutten. Universal coalgebra: a theory of systems, Report **CS-R9652**, CWI, Amsterdam, 1996.

[RT93] J.J.M.M.Rutten, D.Turi. On the Foundations of Final Semantics: Non-Standard Sets, Metric Spaces, Partial Orders, *REX* Conference Proceedings, J.de Bakker et al. eds., Springer LNCS **666**, 1993, 477–530.

[RT94] J.J.M.M.Rutten, D.Turi. Initial algebra and final coalgebra semantics for concurrency, Report **CS-R9409**, CWI, Amsterdam, January 1994.

[RV97] J.J.M.M.Rutten, E.de Vink, Bisimulation for probabilistic transition systems: a coalgebraic approach (extended abstract), *ICALP'97* Conference Proceedings, P.Degano et al, eds., Springer LNCS **1256**, 1997, 460–470.

[San92] D.Sangiorgi. *Expressing Mobility in Process Algebras: First-Oder and Higher-Order Paradigms*, PhD thesis **CST-99-93**, Department of Computer Science, University of Edinburgh, 1992.

[San95] D.Sangiorgi. On the proof method for bisimulation, *MFCS'95* Conference Proceedings, J.Wiedermann et al. eds., Springer LNCS **969**, 1995, 479–488.

[San96] D.Sangiorgi. A theory of bisimulation for the π-calculus, *Acta Informatica* **33**(1), 1996, 22–69.

[Sca97] I.Scagnetto. π-calcolo poliadico in teoria dei tipi coinduttivi, Tesi di Laurea, Università di Udine, 1997.

[SP82] M.Smyth, G.Plotkin. The category-theoretic solution of recursive domain equations, *SIAM J. of Computing* **11**(4), 1982, 761–783.

[Sta96] I.Stark. A fully abstract domain model for the π-calculus, *LICS'96* Conference Proceedings, IEEE Press, 1996.

[Tal90] C.Talcott. A Theory for Program and Data type Specification, *TCS*, Disco90 Special Issue, 1990.

[Tur96] D.Turi. *Functorial Operational Semantics and its Denotational Dual*, PhD thesis, CWI, 1996.

[Tur97] D.Turi. Categorical Modeling of Structural Operational Rules, to appear in *CTCS'97* Conference Proceedings, Springer LNCS.

[TJ93] D.Turi, B.Jacobs. On final Semantics for applicative and non-deterministic languages, Fifth Biennial Meeting on *Category Theory and Computer Science*, Amsterdam, 1993.

[Wad76] C.Wadsworth. The relations between computational and denotational properties for Scott's D_∞-models of the λ-calculus, *SIAM* J. of Computing **5**(3), 1976, 488–521.

[Wal95] D.Walker. Objects in the π-calculus, *Information and Computation* **116**, 1995, 253–271.

Appendix A

Categorical Definitions

In this appendix, we recall some categorical definitions used in Chapter 3. For more details see e.g. [FS90].

Throughout this appendix we work in a category $\mathcal{C}$. The objects of $\mathcal{C}$ are ranged over by A, B, C, etc., while the morphisms are ranged over by a, b, c, etc..

Definition A.0.1 (Pullback, Kernel Pair) *Let $f : A \to C$ and $g : B \to C$.*

- *A* weak pullback *of f and g is a triple $(P, p_1 : P \to A, p_2 : P \to C)$ such that the following diagram commutes*

$$\begin{tikzcd}
 & P \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
A \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\
 & C &
\end{tikzcd}$$

- *A* pullback *of f and g is a weak pullback such that, for any other weak pullback $(P', p'_1 : P' \to A, p'_2 : P' \to C)$, there exists a unique morphism $h : P' \to P$ such that the following diagram commutes*

$$\begin{tikzcd}
 & P' \arrow[d, "h"] \arrow[ddl, "p'_1"'] \arrow[ddr, "p'_2"] & \\
 & P \arrow[dl, "p_1"] \arrow[dr, "p_2"'] & \\
A \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\
 & C &
\end{tikzcd}$$

- *A* kernel pair *is a pullback for a morphism $f : A \to C$ with itself.*

Definition A.0.2 (Equalizer) *Let $f, g : A \to B$. An* equalizer *of f and g is a pair $(E, e : E \to A)$ such that*

1. *$f \circ e = g \circ e$, i.e.* $E \xrightarrow{e} A \overset{f}{\underset{g}{\rightrightarrows}} B$

2. *for all $h : C \to A$ such that*
$$\begin{tikzcd} & C \arrow[d, "h"] & \\ E \arrow[r, "e"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \end{tikzcd}$$
there exists a unique $k : C \to E$ making the following diagram commute
$$\begin{tikzcd} & C \arrow[dl, "k"'] \arrow[d, "h"] & \\ E \arrow[r, "e"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \end{tikzcd}$$

Definition A.0.3 (Binary Product) *A* binary product *of A and B is a triple $A \xleftarrow{l} A \times B \xrightarrow{r} B$ such that, for all $f : C \to A$, $g : C \to B$, there exists a unique $< f, g >: C \to A \times B$ making the following diagram commute*

$$\begin{tikzcd} A & A \times B \arrow[l, "l"'] \arrow[r, "r"] & B \\ & C \arrow[ul, "f"] \arrow[u, "<f,g>"] \arrow[ur, "g"'] & \end{tikzcd}$$

Following [FS90], we define *cartesian categories* as follows:

Definition A.0.4 (Cartesian Category) *A* cartesian category *is a category with finite products and equalizers.*[1]

Definition A.0.5 (Bicartesian Category) *A* bicartesian category *is a category which is both cartesian and cocartesian, the latter meaning that the opposite category is cartesian.*

Definition A.0.6 (Image)
- *A subobject of B, $i : B' \rightarrowtail B$,*[2] allows *$f : A \to B$ if and only if there exists $h : A \to B'$ such that the following diagram commutes*

[1] Notice that in the standard definition of cartesian category the requirement on equalizers is omitted.

[2] We use $\rightarrowtail$ to denote monic morphisms.

$$\begin{tikzcd} & A \arrow[dl, "h"'] \arrow[d, "f"] \\ B' \arrow[r, tail, "i"] & B \end{tikzcd}$$

- *The* image *of $f : A \to B$, if it exists, is the smallest subobject that allows f.*

Definition A.0.7 (Cover) *A morphism $c : A \to B$ is a* cover, *denoted by $c : A \twoheadrightarrow B$, if its image is entire, i.e. for all $f : A \to C$ and $i : C \rightarrowtail B$ there exists $g : B \to C$ such that the following diagram commutes*

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[d, "c"] \\ C \arrow[r, tail, shift left, "i"] & B \arrow[l, shift left, "g"] \end{tikzcd}$$

Definition A.0.8 (Regular Category) *A* regular category *is a category with images and in which pullbacks tranfer covers, i.e., if $(P, p_1 : P \to A, p_2 : P \to B)$ is a pullback of a cover $c : A \twoheadrightarrow C$ and a morphism $g : B \to C$, then also p_2 is a cover, and moreover there exist a cover $c' : B \twoheadrightarrow D$ and a mono $i : D \rightarrowtail C$ which make the following diagram commute*

$$\begin{tikzcd} P \arrow[r, two heads, "p_2"] \arrow[dd, "p_1"'] & B \arrow[dd, "g"] \arrow[dr, two heads, "c'"] & \\ & & D \arrow[dl, tail, "i"] \\ A \arrow[r, two heads, "c"] & C & \end{tikzcd}$$

In the definition below we introduce the notion of *AC regular category. AC* stands for the Axiom of Choice. In particular, the Axiom of Choice asserts that the regular category *Set* is an *AC* regular category.

Definition A.0.9 (*AC* regular category)

- *A relation R on an object A is* entire *if the identity relation on A, $Id_{A\times A}$, is such that $Id_{A\times A} \leq R^{-1} \circ R$.*

- *An object is* choice *is every entire relation targeted at it contains the graph of a map.*

- *An* AC regular category *is a regular category if every object is choice.*

Definition A.0.10 (Inverse Image) *Let* $f : A \to B$. $i : A_1 \mapsto A$ *is an* inverse image *of* $j : B_1 \mapsto B$ *if there exists* $h : A_1 \to B_1$ *such that* $(A_1, h : A_1 \to B_1, i : A_1 \mapsto A)$ *is a pullback of* $j : B_1 \mapsto B$ *and* $f : A \to B$.

Definition A.0.11 (Pre-logos) *A* pre-logos *is a regular category such that*

- *for all* A, *the family of all subobjects of* A, $Sub(A)$, *is a lattice;*
- *for all* $f : A \to B$, *the function* $f^\sharp : Sub(B) \to Sub(A)$, *which associates to each object in* $Sub(B)$ *its inverse image is a lattice homomorphism.*