Università degli Studi di Pisa

Facoltà di Scienze Matematiche Fisiche e Naturali
Dottorato di Ricerca in Informatica

Ph.D. Thesis

ON THE ALGEBRAIC APPROACH TO CONCURRENT TERM REWRITING

Fabio Gadducci

1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL:+39-50-887111. FAX: +39-50-887226

Bah! Kid talk!
No man is poor
who can do
what he **likes** to do
once in a while!
Carl Barks, as *$crooge McDuck*.

To my parents: it was more than paying for it...

Acknowledgements

It is not a secret that acknowledgments are the most difficult part to write, in a thesis. You are always afraid to forget some people, or to mention them in the wrong way.

So, let me start with the most classical opening, thanking my supervisor, Ugo Montanari. His constancy and insight have always won over my unsteadiness, and guided me during all this time. I also want to thank Andrea Corradini: working with him these years surely helped me to learn (and improve) my craft. Together with Ugo, they are the culprits for what you hold in your hands.

Besides them, there are a lot of friends in the Department of Computer Science that, with their discussions and friendship, helped me through these years.

First, my roommates. In particular, Gioia Ristori, Vladimiro Sassone and my Ph.D. collagues Corrado Priami, Paola Quaglia and Laura Semini. Among "those of my own age", let me at least mention also Stefano Guerrini, Alessio Guglielmi and Maria Chiara Meo. Then, our front-office: Gianluigi Ferrari, Andrea Masini and Francesca Rossi; thanks for all the times I came bothering you. My hidden counselor all these years, Simone Martini, and his wife Antonella. And many other people like Daniela Archieri, Paola Bruscoli, Marco Comini, Alessandra Di Pierro, Giorgio Ghelli, Giorgio Levi, Andrea Maggiolo-Schettini... Really many of them, and I beg pardon for all of those I forgot to mention: they know I'm grateful for all their help.

And what about all the international connections? Let me just name a few people. My "Italian" friends, Marcelo Fiore and Claudio Hermida: what a wonderful (not only categorically) summer. The Spanish section, José Meseguer and Narciso Martí-Oliet: thanks for all the interest and care you took for my work. The French rewriters, Claude Kirchner and Patrick Viry: I hope I learned something on what rewriting is, in the end.

There would be also many people outside the Department, but I think their presence here would be to a certain extent pointless (Hello, Mirko: finally some free time!). Let me just thank my family: they fully supported me in many ways all these years (Ciao, Crissie). And, last but not least, to Francesca: it is a long and winding road.

I keep my feeling buried inside a sea urchin.
Ezra Pound

Contents

1 Introduction **1**

2 Preliminaries on (Infinitary) Rewriting **7**

2.1 Varieties of Finite Algebras 7

2.2 Continuous Algebras 10

2.2.1 Explicit Construction of CT_Σ 11

2.2.2 Varieties of (Complete) Ordered Algebras 12

2.3 Term Rewriting 14

2.3.1 Parallel Rewriting 17

2.3.2 Infinite Parallel Rewriting 19

3 Some Notions of Category Theory **23**

3.1 Basic Definitions 23

3.2 Cartesianity as Enriched Monoidality 29

3.3 Cat-Enriched Categories 31

3.4 Internal Categories 35

3.4.1 Double-categories 36

3.5 Algebraic Theories 40

4 Rewriting Logic: Syntax and Semantics **43**

4.1 Rewriting Theories 44

4.2 Algebraic Semantics 47

4.3 $\mathcal{R}$-Systems 51

4.4 Functorial Models of Rewriting Theories 54

5 Consistency with Finitary Rewriting **59**
5.1 Consistency between Operational Semantics 60
5.2 Consistency between Abstract Semantics 65

6 On Concurrent Rewriting **71**
6.1 Permutation vs. Concurrency 72
6.2 Flat Models Form Prime Algebraic Domains 77
6.3 Cartesianity vs. Interchange 81

7 Dealing with Infinitary Rewriting **83**
7.1 Infinite Parallel Rewriting Logic 84
7.2 Consistency between Operational Semantics 86
7.3 Continuous Models of Term Rewriting 91

8 Typed Rewriting Logic **93**
8.1 Typed Rewriting Logic: Syntax 94
8.2 Typed Rewriting Logic: Functorial Semantics 96
8.3 An Application in Concurrency Theory 98
8.3.1 Process Description Algebras 99
8.3.2 Context Systems 103
8.4 Strategies for Rewriting 107
8.4.1 Recovering ELAN 108

9 Further Work: On Term *Graph* Rewriting **111**
9.1 S-Monoidal Theories 112
9.2 Some Results on Term Graphs 113

Bibliography **121**

Chapter 1

Introduction

As an autonomous research field, term rewriting can be actually dated to the introduction of the so-called *Knuth-Bendix completion procedure* [KB70]. The motivations for such an algorithm lie on the mid-Thirties work on *algebraic varieties* by Birkoff, where a *variety* is a class of algebras satisfying a given set of axioms. The deduction method to derive new equations from a given specification is known as *equational logic*: it simply puts on a formal ground the intuitive "substitution of equals" technique, allowing to derive new equations from older ones. The aim of the Knuth-Bendix algorithm is to transform a given input specification into a *convergent* rewriting system: a system such that rewriting is always terminating, and produces a unique possible result for any term, its *normal form*. In this system, the equations of the input specification are turned into *one-way* rules, such that two terms are equated in the input theory if and only if they have the same normal form.

It is not a mere understatement to say that, in these years, term rewriting has been *implicitly* analyzed only as a *technique*, in correspondence with its use as an efficient form of equational deduction. This explains the syntactical nature of the research in the field, between the two guidelines of *termination* (we must ensure that rewriting always terminates) and *Church-Rosser property* (we must ensure that rewriting produces a unique normal form). Such a choice has been sustained also by the interest of computer scientists for term rewriting, due for example to the introduction of *algebraic semantics* for program schemes (see e.g. [Gue81]).

Of course, if we are just interested in equational deduction, we can safely avoid dealing with the "details" of how the reduction process is actually implemented. However, this approach is too narrowing, especially so since term rewriting can be considered as a basic

computational paradigm, where terms are interpreted as states of an abstract machine, while rewriting rules are state-transforming functions.

In the classical set-theoretical approach to term rewriting, the basic notion is that of *sequence* of rewrites: to each term t a *derivation space* is associated, i.e., a class of "transitions" describing all the possible reductions starting from t. Derivation spaces represent a naïve operational model: each element of a space is interpreted as a sequence of "concrete" steps of an underlying, sequential machine dealing with data structures representing syntactical descriptions of terms. Such an operational description however is not sufficient if we want to take into account different representations for terms, able to recover notions like *sharing* of subterms, or if we want to deal with a distributed system, such that the reduction mechanism is implemented over a network of processors.

The use of suitable *graph expressions* to describe terms, handling explicitly the *sharing* of subexpressions, dates at least to the late Sixties [Wad71]. In this setting, a single reduction step (on a graph) can subsume a long sequence of (sequential) rewrites: "concrete" derivations that are different when dealing with the syntactical description of terms can represent the same derivation when this more complex data structure is used. When introducing *permutation equivalence* [Lev80], Lévy wanted to capture the fundamental uniqueness of such a reduction: permutation equivalence equates sequences of syntactical rewrites that correspond to the same contractions on graphs. Two reduction steps are then *compatible* if they are independent with respect to the modifications they induce on the data structure (i.e., if they act on disjoint portions of the graph).

The studies on the permutation equivalence paradigm in the term rewriting framework can be dated back at least to [Bou85]. Along this line of research, in [Mar91] the author proves the adequacy of the equational representation for describing graph reduction in the case of orthogonal term rewriting systems (informally, those such that all their rules have no subexpression in common). Moreover, a "lazy" reduction strategy (i.e., an actual algorithm) is provided that is "optimal", in the sense that it is able to perform the minimal number of graph reductions. The reduction strategy, however, implicitly assumes a sequential machine over which the reduction mechanism is implemented, or at least the presence of a (hidden) *synchronizing agent*. Furthermore, the equivalence does not describe faithfully the behaviour of the reduction mechanism over graphs, whenever not-orthogonal rules are taken into account.

When dealing with a distributed system, instead, we assume that the reduction process is implemented over a network of processors. From a theoretical point of view, the easiest choice is the *one-node/one-processor* architecture: terms are described as trees, and we

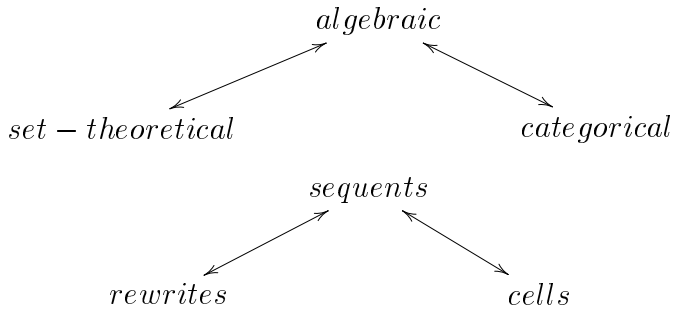
have a tree-like implementation schema, where processors are *loosely coupled* (i.e., the control is totally distributed) and each processor has knowledge only of the rewriting rules of the system, of the information of the actual node it is associated with, and of the nodes it is linked with (i.e, we have *fine-grain parallelism*). Such an architecture is well-suited for theoretical purposes (since we consider irrelevant the eventual inter-processor communications), even if a realistic implementation should find a suitable trade-off between fine and coarse-grain, distributed and centralized control: for a careful discussion of these topics, we refer to the works of the *Rewriting Rule Machine* project [GKM87, Gog90, LMR94]. For our purposes, however, we need just remark that many concrete derivations may correspond to the simultaneous execution of the same set of compatible reductions. Here compatibility means that they act on disjoint portion of the tree, or, in different terms, that they are "causally unrelated". A *concurrent semantics* then has to provide an equivalence over sequences of syntactical rewrites equating those that correspond to the same parallel reduction, while an optimal strategy means to provide a notion of compatibility allowing the simultaneous execution of as many single sequential reductions as possible.

An alternative approach to the semantics of term rewriting relies on the use of suitable *cat-enriched categories*, namely *2-categories* and *sesqui-categories*. A cat-enriched structure is given by a category such that each hom-set also forms a category: the class of morphisms (called *cells*) of these hom-categories are closed under certain composition operators, and are subject to suitable *coherence* axioms. Such algebraic structures are freely generated from a *c-computad*: a pair $\langle \mathbf{D}, S \rangle$, where $\mathbf{D}$ is a category whose arrows represent terms, and S is a collection of cells describing the rewriting rules. This way we obtain derivation spaces (the cells of the enriched structure) that are already subject to an equivalence relation, due to the coherence axioms: since there exists a correspondence between cells and sequences of rewrites, we get "for free" suitable equivalences over derivations. Moreover, the explicit description of terms and rewrites provided by the categorical framework is such that the resulting model is much closer to a possible implementation, since all the relevant properties of the involved operations (like *substitution*) that are sometimes overlooked in the classical description of reduction are to be made explicit in this representation. To a large extent, the categorical description underlines and clarifies the relevant aspects of the derivation mechanism, and may furthermore suggest useful refinements.

The set-theoretical and the categorical semantics of term rewriting can be related el-

egantly through the use of *Rewriting Logic* [Mes92], whose logical approach has its roots in a seminal paper by Meseguer and Montanari [MM90] on the semantics of *Petri Nets* [Rei85]. The idea of rewriting logic is to take a logical viewpoint, regarding a rewriting system $\mathcal{R}$ as a theory, and any rewriting - making use of rules in $\mathcal{R}$ - as a *sequent* entailed by the theory, where the entailment relation is defined inductively by a given set of deduction rules. Each of these rules can be considered as a general pattern for a *basic action* that can occur concurrently with many others, so that rewriting logic can be considered as a *logic of changes*, while different sets of rules mean different assumptions on the reduction mechanism. Derivations are then naturally equipped with an algebraic structure: a sequent is a triple $\alpha : s \to t$, where α is an element of a *proof term algebra* encoding the justifications of the rewrites of s into t. The underlying idea is that, if for a sequential system a set of states and a set of transitions between states can already be considered a faithful description of its behaviour, this is not true anymore for *concurrent* ones. More precisely, what allows us to recover faithfully a concurrent behaviour is the presence of an algebraic structure. If we are just interested in equational deduction, one can immediately recover the set-theoretical rewrite relation, simply ignoring the proof term of a sequent $\alpha : s \to t$. At the same time, one can easily provide different equivalences over sequences of rewrites, imposing suitable sets of axioms over proof terms.

The aim of the thesis is twofold. As pointed out in [MOM91], rewriting logic establishes a generalization of the Curry-Howard correspondence, namely the *Lambeck-Lawvere correspondence* [Lam68, Lam69, Lam72], for term rewriting, as summed up in the following figure:



The first part of the thesis studies in depth the correspondence between the different sides of the analogy in the term rewriting setting. In Chapter 4 we introduce the paradigm of rewriting logic, using two different sets of deduction rules: the *full entailment* is a variant of the relation proposed by Meseguer in his paper [Mes92], while the *flat* entailment is our own, and was originally introduced in [CGM95]. Then we equip the two relations with

two different set of axioms over proof terms, obtaining respectively families of *abstract full* and *abstract flat* sequents, and we provide sound and complete categorical models for both entailment relations. In Chapter 5 we carefully relate these semantics with two equivalences induced over derivations in the set-theoretical setting, the well-known permutation equivalence and a different one we called *disjoint equivalence*. Elaborating on the results in [LM92], we prove the one-to-one correspondence between the families of permutation (disjoint) equivalent derivations, and the families of abstract full (flat) sequents. Finally, in Chapter 6 we study the algebraic properties of the models, with the aim of showing that the equivalences we introduced can be considered as providing a suitable concurrent semantics for the reduction mechanism. As already remarked in a series of papers in the early Seventies (see e.g. [Ben75]) on the semantics of the reduction process in *context free grammars*, and following a now common practice in the concurrency community, we say that a given equivalence expresses a *direct implementability* property (i.e., it is theoretically feasible to describe a suitable distributed network whose processors directly implement abstract reductions) only if the families of abstract sequents form with respect to the intuitive prefix ordering a *prime algebraic domain* (a model tightly related to the well-accepted formalism of *event structures* [Win89]). We prove that abstract flat sequents form such a domain; full ones, instead, do not (as could be also inferred from [Lan94]). Moreover, we provide an intuitive justification of this fact via a possible implementation schema of the reduction process over the one-node/one-processor distributed architecture.

Many rule-based formalisms in theoretical computer science are provided with satisfactory operational semantics, usually given in a set-theoretical setting. The overall goal of the second part of this thesis is to show that for a large class of formalisms a categorical semantics can be provided as well, on the assumption that "concurrency lies on the algebraic structure, modulo a suitable axiomatization". Moreover, as for term rewriting, a corresponding logic is defined that constitutes the "trait d'union" between the operational and algebraic semantics. This idea will be applied to the following cases.

1. To describe *infinitary parallel term rewriting* (Chapter 7), dealing with terms of infinite depth and (possibly) infinite sets of rewrites (all of them applied in parallel to a given term). The operational semantics was presented in [Cor93], while the corresponding rewriting logic and categorical semantics was introduced in [CG95]. The main definitions involve an infinitary rule (corresponding to an ω-completion) for the extended logic, and the use of CPO-*enrichment* for the categorical models.

2. To recast formalisms relying on the use of *side effects* in determining the actual

behaviour of a given system (Chapter 8), i.e., such that a transition relation is not sufficient anymore to describe the evolution of the system. Suitable examples are *process algebras* [Hoa85, Mil89], whose operational semantics is described by means of *labeled transition systems*, or generalizations of this paradigm like *context systems* [LX90] (Chapter 8). The associated logic introduces the notion of *typed sequent* (i.e., such that the algebra of proof terms is subject to suitable typing conditions), while the categorical model uses *double-categories* [GM95a, GM95b].

3. To implement *strategies* for term rewriting, i.e., to describe algorithms designed to choose deterministically which rewrite to execute at each point of the computation. Usually, a strategy is simply given by a function associating to each term a set of its subterms. In Chapter 8 we refer some work in progress, relying on the use of typed sequents, where the typing conditions now carry information on which rewrites can be performed.

4. To describe *term graph rewriting*. The classical operational semantics can be found in the introductory chapter of [EPS93], while in Chapter 9 we investigate on a possible algebraic (categorical) description for term graphs and term graph rewriting, showing some preliminary but quite enlightening results. The categorical model involves the use of *symmetric monoidal categories*, equipped with two transformations representing the explicit sharing and garbaging of terms.

From the semantic point of view, then, we establish a Lambek-Lawvere analogy for a large class of formalisms, usually equipped with a set-theoretical semantics, via suitable extensions of rewriting logic, used as a tool. Moreover, the categorical side is usually dealt with by reformulating the established categorical semantics for rewriting logic.

Chapter 2

Preliminaries on (Infinitary) Rewriting

The aim of this chapter is to recall the basic definitions of classical, *set-theoretical term rewriting*, that play an important röle in the thesis. The first two sections, however, are devoted to the introduction of the notions of *algebra*, *continuous algebra* and *variety*. For the finite case, any introductory book like [Gra79] provides a far more interesting background; for the continuous case, our exposition is based on classical papers like [ADJ77, Blo76].

2.1 Varieties of Finite Algebras

Definition 2.1 (Σ-Algebras) *Let Σ be a (one-sorted)* signature*, i.e., a ranked alphabet of operator symbols $\Sigma = \cup_{n\in\mathbb{N}}\Sigma_n$ (saying that f is of* arity *n for $f \in \Sigma_n$). A* Σ-algebra *is a pair $A = \langle |A|, \rho_A\rangle$ such that $|A|$ is a not-empty set (the* carrier*), and $\rho_A = \{f_A \mid f \in \Sigma\}$ is a family of functions such that for each $f \in \Sigma_n$, $f_A : |A|^n \to |A|$. Let A, B be two Σ-algebras: a* Σ-homomorphism *$\tau : A \to B$ is a function $\tau : |A| \to |B|$ preserving operators, i.e., such that for every $f \in \Sigma_n$, $\tau \circ f_A = f_B \circ \tau^n$: $\tau(f_A(a_1, \ldots, a_n)) = f_B(\tau(a_1), \ldots, \tau(a_n))$.* □

Since Σ-homomorphisms are closed under (functional) composition, and the identity function of the carrier is a Σ-homomorphism, the class **Σ-Alg** of Σ-algebras and Σ-homomorphisms forms a *category* (see Chapter 3). We recall some well-known results for the class of Σ-algebras.

Proposition 2.1 (Initial Algebra) *Let Σ be a signature. Then the category* **Σ-Alg** *admits an* initial algebra*, denoted T_Σ, such that for any Σ-algebra A, there exists a unique homomorphism $\tau_A : T_\Sigma \to A$.* □

The construction of T_Σ is well-known, and we will not repeat it here. Next proposition, however, states the key result for any proof carried out by "structural" induction over terms.

Proposition 2.2 (Unique Construction) *Let Σ be a signature, and T_Σ its initial algebra. Then for any $t \in T_\Sigma$ there exists a unique $n \in \mathbb{N}$, $f \in \Sigma_n$ and $t_i \in T_\Sigma$, $i = 1 \ldots n$ such that $t = f(t_1, \ldots, t_n)$.* □

We introduce now the notion of *freely generated* algebra over a set of *variables*.

Definition 2.2 (Freely Generated Algebras) *Let Σ be a signature, and X a set (whose elements will be called* variables*) such that $\Sigma \cap X = \emptyset$. We define the* free algebra over X *as the initial algebra of* **Σ(X)-Alg**, *denoted $T_\Sigma(X)$, where $\Sigma(X)$ is the signature obtained from Σ by adding the elements of X as constants.* □

The following results about *substitutions* and *derived operators* are also well-known.

Definition 2.3 (Substitutions) *Let Σ be a signature, and X, Y be two sets of variables. A* substitution *(from X to Y) is a function $\sigma : X \to T_\Sigma(Y)$ (used in postfix notation). A substitution σ from X to Y uniquely determines a Σ-homomorphism (also denoted by σ) from $T_\Sigma(X)$ to $T_\Sigma(Y)$, which extends σ as $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ for each $f \in \Sigma_n$, for all $n \in \mathbb{N}$. If X is finite, a substitution σ from X to Y can be represented as a finite set $\{x_1/t_1, \ldots, x_n/t_n\}$ with $t_i = x_i\sigma$ for all $1 \leq i \leq n$ and, given a term t such that $var(t) \subseteq \{x_1, \ldots, x_n\}$, we usually write $t(t_1, \ldots, t_n)$ for $t\sigma$.* □

Proposition 2.3 *Let Σ be a signature, A a Σ-algebra and X a set of variables. Then any function $h : X \to |A|$ (also called* evaluation*) can be uniquely extended to a Σ-homomorphism $\tau_h : T_\Sigma(X) \to A$.* □

The previous proposition states that each term $t \in T_\Sigma(\{x_1, \ldots, x_n\})$ actually defines an n-nary function, the *derived operator* t_A, for any Σ-algebra A: given any possible evaluation $h : \{x_1, \ldots, x_n\} \to A$, with $h(x_i) = a_i$ for $i = 1 \ldots n$, then $t_A(a_1, \ldots, a_n) = \tau_h(t)$. Note that we could define the derived operator as an m-ary function, for each $n \leq m$, whenever we have $\#|var(t)| = n$: the result, however, is not influenced by the auxiliary arguments.

Definition 2.4 (Equations) *Let Σ be a signature and X a set of variables. Then an* equation *over $\Sigma(X)$ is a pair $\langle t, s\rangle$ of elements of $T_\Sigma(X)$. An algebra A satisfies an equation $\langle t, s\rangle$ iff for any evaluation $h : \{x_1, \ldots, x_n\} \to A$, with $h(x_i) = a_i$ for $i = 1 \ldots n$, such that both $\tau_h(t)$ and $\tau_h(s)$ are defined, the identity $t_A(a_1, \ldots, a_n) = s_A(a_1, \ldots, a_n)$ holds.* ∎

A *variety* is the class of all algebras satisfying a given set E of equations. For a given set of axioms E defined over $\Sigma(X)$, let us indicate with $\cong_E$ the minimal congruence induced by the axioms in E over T_Σ, closed with respect to substitutions $X \to T_\Sigma$. The *quotient algebra* $T_{(\Sigma,E)} = T_\Sigma / \cong_E$ is defined as the Σ-algebra whose carrier consists the equivalence classes of elements of $|T_\Sigma|$ modulo the congruence $\cong_E$, while, for each operator $f \in \Sigma_n$, $f_{T_{(\Sigma,E)}} : \langle [t_1], \ldots, [t_n]\rangle \to [f(t_1, \ldots, t_n)]$ (where $[t_i]$ is the equivalence class associated to the term t_i).

Proposition 2.4 (Initiality in Varieties) *Let Σ be a signature, and E a set of axioms. Then $T_{(\Sigma,E)}$ is initial in the variety associated to E.* ∎

This result shows that, as for establishing identities holding in all the algebras of a given variety, it is sufficient to consider just elements of T_Σ. The next proposition states that, given a set of axioms E, the identities holding in $T_{(\Sigma,E)}$ can be inferred by a finite application of certain deduction rules: hence, proofs can be carried out by *structural induction* over terms of T_Σ.

Proposition 2.5 (Equational Logic) *Let Σ be a signature, X a set of variables and E a set of axioms defined over $\Sigma(X)$. Given two terms $t, s \in T_\Sigma$, the equality $t = s$ holds in $T_{(\Sigma,E)}$ iff it can be inferred by finite application of the following rules:*

- $$\frac{t \in T_\Sigma}{t = t};$$
- $$\frac{t(x_1, \ldots, x_n) = s(x_1, \ldots, x_n) \in E, t_i = s_i \text{ for } i = 1, \ldots, n}{t(t_1, \ldots, t_n) = s(s_1, \ldots, s_n)};$$
- $$\frac{f \in \Sigma_n, t_i = s_i \text{ for } i = 1, \ldots, n}{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)};$$
- $$\frac{s = t, t = u}{s = u}.$$

∎

A well-studied extension of the class of algebras we introduced are *partial algebras* over a given signature: pairs $\langle A, \rho_A \rangle$ where ρ_A characterizes a family of *partial* functions over the carrier $|A|$. Since we will only occasionally deal with these structures, we refer the reader to [Gra79]: next section will be devoted instead to introducing the notion of *continuous algebra*.

2.2 Continuous Algebras

In this section we consider algebras whose carriers form an ordered structure, namely a (ω-complete) partial order. Correspondingly, homomorphisms do not preserve only the algebraic structure, but are also order-preserving (e.g, monotone, continuous) functions. The importance of these algebras in computer science has been stressed by a great deal of work during the Seventies, mainly inside the algebraic semantics framework, for example in providing denotational semantics for functional languages, semantical models for flow diagrams and, in general, for formalisms dealing with unbounded computations. Let us first recall some definition about (complete) partial orders.

Definition 2.5 ((Complete) Partial Orders) *A* partial order *is a pair $\langle D, \leq \rangle$, where $\leq$ is a reflexive, antisymmetric and transitive binary relation over a set D. A partial order is* strict *if it has an element $-$ (called* bottom*) such that $- \leq d$ for all $d \in D$. A partial order is ω-complete (also* complete*) if it admits* least upper bounds *(*LUB*'s) for all ω-chains of elements. If $\{d_i\}_{i<\omega}$ is an ω-chain (i.e., $d_i \leq d_{i+1}$ for all $i < \omega$), we denote its* LUB *by $\bigsqcup_{i<\omega}\{d_i\}$. A* monotone function *$f : \langle D, \leq_D \rangle \to \langle D', \leq_{D'} \rangle$ between partial orders is a function $f : D \to D'$ which preserves the ordering relation, i.e., such that if $t \leq s$, then $f(t) \leq f(s)$; it is* strict *if moreover $f(-_D) = -_{D'}$. A* continuous function *$f : \langle D, \leq_D \rangle \to \langle D', \leq_{D'} \rangle$ between complete partial orders is a function $f : D \to D'$ which preserves* LUB*'s of ω-chains, i.e., $f(\bigsqcup_{i<\omega})\{d_i\} = \bigsqcup_{i<\omega}\{f(d_i)\}$.* ∎

The *product* of (complete) partial orders can be defined as the cartesian product of their underlying sets, while the ordering relation is defined pointwise. Then, most properties of Σ-algebras can be extended to take into account *(complete) ordered* algebras, i.e., algebras whose carriers are (complete) partial orders (CPO's from now on).

Definition 2.6 (The Category of (Complete) Ordered Σ-Algebras) *A* (complete) ordered Σ-algebra *is a pair $A = \langle \langle |A|, \leq_A \rangle, \rho_A \rangle$ such that $\langle |A|, \leq_A \rangle$ is a (complete) partial order, and $\rho_A = \{f_A \mid f \in \Sigma\}$ is a family of* monotone *(continuous) functions such that for*

every $f \in \Sigma_n$, $f_A : \langle |A|, \leq_A \rangle^n \to \langle |A|, \leq_A \rangle$. *If* A, B *are (complete) ordered* Σ*-algebras, an* ordered (continuous) Σ-homomorphism $\tau : A \to B$ *is a monotone (continuous) function* $\tau : \langle |A|, \leq_A \rangle \to \langle |B|, \leq_B \rangle$ *which preserves the operators.* ∎

Σ-OAlg and **Σ-COAlg** denote respectively the categories having (complete) ordered Σ-algebras as objects and ordered (continuous) Σ-homomorphisms as arrows. In both cases, the initial algebra FT_Σ is given by the rather uninteresting pair $\langle \langle |T_\Sigma|, = \rangle, \rho_{T_\Sigma} \rangle$: the algebra T_Σ equipped with the identity relation. A much more important class is represented by *continuous algebras.*

Definition 2.7 (The Category of Continuous Σ-Algebras) *A* continuous Σ-algebra *is a pair* $A = \langle \langle |A|, \leq_A \rangle, \rho_A \rangle$ *such that* $\langle |A|, \leq_A \rangle$ *is a strict* CPO, *and* $\rho_A = \{ f_A \mid f \in \Sigma \}$ *is a family of continuous functions such that for every* $f \in \Sigma_n$, $f_A : \langle |A|, \leq_A \rangle^n \to \langle |A|, \leq_A \rangle$. ∎

Σ-CAlg denotes the category having continuous Σ-algebras as objects and strict continuous Σ-homomorphisms as arrows. As shown in [ADJ77], also this class has an initial algebra, denoted CT_Σ: its elements are possibly infinite, possibly partial terms freely generated from Σ, and they form a strict CPO where the ordering relation is given by $t \leq t'$ iff t' is "more defined" than t.

2.2.1 Explicit Construction of CT_Σ

Since continuous algebras will be very important in this thesis, we give in this section an explicit construction of CT_Σ; definitions are borrowed from [ADJ77], with minor changes.

Definition 2.8 (Occurrences) *Let* ω^* *be the set of all finite strings of positive natural numbers; its elements are called* occurrences, *and the empty string is denoted by* λ. ∎

Definition 2.9 (Terms as Functions) *Let* Σ *be a signature and* X *a set of variables such that* $\Sigma \cap X = \emptyset$. *A* term *over* (Σ, X) *is a partial function* $t : \omega^* \to \Sigma \cup X$, *such that the domain of definition of* t, $\mathcal{O}(t)$, *satisfies (for* $w \in \omega^*$ *and* $i \in \omega$*):*

- $wi \in \mathcal{O}(t) \Rightarrow w \in \mathcal{O}(t)$;
- $wi \in \mathcal{O}(t) \Rightarrow t(w) \in \Sigma_n$ *for some* $n \geq i$.

$\mathcal{O}(t)$ *is called the* set of occurrences *of* t. *A term* t *is* total *if* $t(w) \in \Sigma_n \Rightarrow wi \in \mathcal{O}(t)$ *for all* $0 < i \leq n$. ∎

The set of terms over (Σ, X) is denoted by $CT_\Sigma(X)$ (CT_Σ stays for $CT_\Sigma(\emptyset)$). For finite, total terms, this description is equivalent to the usual representation of terms as operators applied to other terms. Partial terms are made total in this representation by introducing the undefined term $-$, which represents the empty function $- : \emptyset \to \Sigma \cup X$, always undefined. Thus, for example, if $x \in X$, $t = f(-, g(x))$ is the term such that $\mathcal{O}(t) = \{\lambda, 2, 2 \cdot 1\}$, $t(\lambda) = f \in \Sigma_2$, $t(2) = g \in \Sigma_1$, and $t(2 \cdot 1) = x \in X$.

$CT_\Sigma(X)$ forms a strict CPO with respect to the "approximation" relation. We say that t *approximates* t' (written $t \leq t'$) iff t is less defined than t' as partial function[1]. The least element of $CT_\Sigma(X)$ with respect to $\leq$ is clearly $-$. An ω-chain $\{t_i\}_{i<\omega}$ is an infinite sequence of terms $t_0 \leq t_1 \leq \ldots$. Every ω-chain $\{t_i\}_{i<\omega}$ in $CT_\Sigma(X)$ has a LUB $\bigcup_{i<\omega}\{t_i\}$ characterized as follows:

$$t = \bigcup_{i<\omega} \{t_i\} \quad \Leftrightarrow \quad \forall w \in \omega^* \,.\, \exists i < \omega \,.\, \forall j \geq i \,.\, t_j(w) = t(w).$$

Since each pair of terms has a greatest lower bound (GLB), $CT_\Sigma(X)$ is more than a strict CPO: it is actually an *ω-complete lower semilattice*. The following result was originally proved in [ADJ77].

Proposition 2.6 (Initiality for Continuous Algebras) *CT_Σ is initial in Σ-**CAlg**, and it is called the* continuous word algebra *over Σ.* ∎

2.2.2 Varieties of (Complete) Ordered Algebras

In this section we recall the basic results on varieties of ordered algebras that can be found in [Blo76].

Proposition 2.7 *Let Σ be a signature, A a (complete) ordered Σ-algebra and X a set of variables. Then any evaluation $h : X \to |A|$ can be uniquely extended to a ordered (continuous) Σ-homomorphism $\tau_h : FT_\Sigma(X) \to A$.* ∎

The previous result extends the notion of derived operator to ordered and continuous algebras. Each term $t \in FT_\Sigma(\{x_1, \ldots, x_n\})$ actually defines an n-ary function, the derived operator t_A, for any ordered (continuous) Σ-algebra A: given any possible evaluation $h : \{x_1, \ldots, x_n\} \to A$, with $h(x_i) = a_i$ for $i = 1 \ldots n$, then $t_A(a_1, \ldots, a_n) = \tau_h(t)$.

[1] Equivalently, the relation '$\leq$' can be defined as the minimal one such that $\perp \leq t$ for all t; $x \leq x$ for all $x \in X$, and $f(t_1, ..., t_n) \leq f(t'_1, ..., t'_n)$ if $t_1 \leq t'_1, \ldots, t_n \leq t'_n$, for all $f \in \Sigma_n$. This is not by chance: see next section.

Definition 2.10 (Inequalities) *Let Σ be a signature and X a set of variables. An* inequality *over $\Sigma(X)$ is a pair $\langle t, s\rangle$ of elements of $FT_\Sigma(X)$. An algebra A satisfies an inequality $\langle t, s\rangle$ iff for any evaluation $h : \{x_1, \ldots, x_n\} \to A$, with $h(x_i) = a_i$ for $i = 1 \ldots n$, the inequality $t_A(a_1, \ldots, a_n) \leq s_A(a_1, \ldots, a_n)$ holds.* ∎

A *variety* of ordered algebras is the class of all ordered algebras satisfying a given set I of inequalities. For a given set I of inequalities defined over $\Sigma(X)$, let us indicate with $\sqsubseteq_I$ the minimal preorder induced by the inequalities in I over T_Σ, closed with respect to substitutions $X \to T_\Sigma$, and $\cong_I$ the associated congruence. The *quotient ordered algebra* $FT_{(\Sigma,I)} = FT_\Sigma/\cong_I$ is defined as the ordered Σ-algebra whose carrier consists of the equivalence classes of elements of $|T_\Sigma|$ modulo the congruence $\cong_I$, while the associated partial order is $[a] \leq [b]$ iff $a \sqsubseteq_I b$; for each operator $f \in \Sigma_n$, $f_{FT_{(\Sigma,I)}} : \langle [t_1], \ldots, [t_n]\rangle \to [f(t_1, \ldots, t_n)]$ (where $[t_i]$ is the equivalence class associated to the term t_i).

Proposition 2.8 (Initiality in Varieties) *Let Σ be a signature, and I a set of inequalities. Then $FT_{(\Sigma,I)}$ is initial in the variety of ordered algebras associated to I.* ∎

This result shows that, as for establishing inequalities holding in all the algebras of a given variety, it is sufficient to consider just elements of FT_Σ. The next proposition states that, given a set of inequalities I, the inequalities holding in $FT_{(\Sigma,I)}$ can be inferred by finite application of certain deduction rules: hence, proofs can be carried out by *structural induction* over terms of FT_Σ.

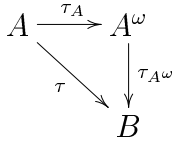
Proposition 2.9 (Logic of Inequalities) *Let Σ be a signature, X a set of variables and I a set of inequalities defined over $\Sigma(X)$. Given two terms $t, s \in T_\Sigma$, the inequality $t \leq s$ holds in $FT_{(\Sigma,I)}$ iff it can be inferred by a finite application of the following rules:*

- $$\frac{t \in T_\Sigma}{t \leq t};$$
- $$\frac{t(x_1, \ldots, x_n) \leq s(x_1, \ldots, x_n) \in I, t_i \leq s_i \text{ for } i = 1, \ldots, n}{t(t_1, \ldots, t_n) \leq s(s_1, \ldots, s_n)};$$
- $$\frac{f \in \Sigma_n, t_i \leq s_i \text{ for } i = 1, \ldots, n}{f(t_1, \ldots, t_n) \leq f(s_1, \ldots, s_n)};$$
- $$\frac{s \leq t, t \leq u}{s \leq u}.$$

∎

A fundamental result, associating to each ordered algebra a suitable *completion*, is given by the following proposition.

Proposition 2.10 (Minimal Extension) *Let Σ be a signature, and A be an ordered Σ-algebra, belonging to a given variety. There exists a complete ordered algebra A^ω in the same variety of A and an ordered homomorphism $\tau_A : A \to A^\omega$ such that, for any other complete ordered algebra B and ordered homomorphism $\tau : A \to B$, there exists a unique continuous homomorphism $\tau_{A^\omega} : A^\omega \to B$ such the following diagram commutes:*



□

Given an ordered algebra A, the previous result implies also that for establishing an inequality $t \leq t'$ for $t, t' \in A^\omega$, it is enough to reason inductively over (ω-chains of) elements in A. Roughly, A^ω is obtained from A just adding the LUB's of ω-chains, and equating those terms that are obtained as LUB's of chains whose components are equal. For example, with a signature $\Sigma = \{f, a\}$ and an axiom $Q = \{a \leq f(x)\}$, the inequalities $f^n(a) \leq f^m(a)$ holds in $FT_{(\Sigma,Q)}$ for all $0 \leq n \leq m < \omega$. Also, the ordering induced by Q implies that $(FT_{(\Sigma,Q)})^\omega$ contains also the infinite term $f^\omega(a) = \bigsqcup_{i<\omega}\{f^i(a)\}$. This property can be used to provide an alternative description of CT_Σ, in term of the completion of the initial ordered algebra of a given variety.

Proposition 2.11 *Let $\Sigma(-)$ be a signature Σ extended with a new constant $-$, and let us consider the inequality $Q = \{- \leq x\}$. Then CT_Σ is isomorphic to $(FT_{\Sigma(\perp)}/\cong_Q)^\omega$.*
□

2.3 Term Rewriting

We introduce now term rewriting over CT_Σ (implicitly describing also term rewriting over T_Σ). A term rewriting system (briefly, TRS) over a set of variables X is a (labeled) set of rules, i.e., of pairs of finite, total terms in CT_Σ. A rule can be applied to a term t if its left-hand side matches a subterm of t, and the result is the term t where the matched subterm is replaced by a suitable instantiation of the right-hand side of the rule. Note that restricting our attention only to CT_Σ, instead of $CT_\Sigma(Y)$ for a given set Y of variables, is by no means limiting the generality.

Definition 2.11 (Term Rewriting Systems) *Let X be a set of variables. A* term rewriting system *$\mathcal{R}$ (over X) is a tuple (Σ, L, R), where Σ is a signature, L is a set of labels, and R is a function $R : L \to T_\Sigma(X) \times T_\Sigma(X)$, such that for all $d \in L$, if $R(d) = \langle l, r\rangle$ then $var(r) \subseteq var(l) \subseteq X$ and l is not a variable.* ∎

Given a term rewriting system (also TRS) $\mathcal{R}$, we usually write $d : l \to r \in R$ if $d \in L$ and $R(d) = \langle l, r\rangle$; to make explicit the variables contained in a rule, we write $d(x_1, \ldots, x_n) : l(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n) \in R$ where $\{x_1, \ldots, x_n\} = var(l)$.

Now we instantiate the notion of substitution to the case of continuous algebras, and we introduce some basic operations on terms that are needed to define the notion of rewriting over continuous algebras. We recall that these definitions subsume the finitary case, in the sense that they coincide with the classical definitions, when we restrict our attention to terms in T_Σ.

Definition 2.12 (Substitutions over Continuous Algebras) *Let X and Y be two sets of variables. A* continuous substitution *from X to Y (just substitution from now on) is a function $\sigma : X \to CT_\Sigma(Y)$ (used in postfix notation). A substitution σ from X to Y uniquely determines a strict continuous Σ-homomorphism (also denoted by σ) from $CT_\Sigma(X)$ to $CT_\Sigma(Y)$, which extends σ as follows:*

- $-\sigma = -$;
- $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$;
- $(\bigcup_{i<\omega}\{t_i\})\,\sigma = \bigcup_{i<\omega}\{t_i\sigma\}$.

If X is finite, a substitution σ from X to Y is described as a finite set $\{x_1/t_1, \ldots, x_n/t_n\}$ with $t_i = x_i\sigma$ for all $1 \leq i \leq n$ and, given a finite term t such that $var(t) \subseteq \{x_1, \ldots, x_n\}$, we usually write $t(t_1, \ldots, t_n)$ for $t\sigma$. ∎

Definition 2.13 (Subterm Selection) *Given an occurrence $w \in \omega^*$ and a term $t \in CT_\Sigma(X)$, the* subterm *of t at (occurrence) w is the term t/w defined as $t/w(u) = t(wu)$ for all $u \in \omega^*$. In the alternative representation of terms, t/w is equivalently defined by the following clauses:*

$$t/w = \begin{cases} t & \text{if } w = \lambda; \\ t_i/w' & \text{if } w = iw' \text{ and } t = f(t_1, \ldots, t_i, \ldots, t_n); \\ \bigcup_{i<\omega}\{t_i/w\} & \text{if } t = (\bigcup_{i<\omega}\{t_i\}); \\ - & \text{otherwise.} \end{cases}$$

It is easy to check that $t/w = -$ iff $w \notin \mathcal{O}(t)$. ∎

We recall now some definitions useful in the following.

Definition 2.14 (On Occurrences) *Let u, v be occurrences: we write $u \leq v$ if u is a prefix of v. We say that they are* disjoint *(denoted $u|v$) if neither $u \leq v$ nor $v \leq u$. The* length *of an occurrence w, denoted $|w|$, is defined as $|\lambda| = 0$ and $|wi| = |w| + 1$ for $w \in \omega^*$ and $i \in \omega$. The* depth *of a term t is defined only if t is finite; in this case, $depth(t) = max\{|w| \mid w \in \mathcal{O}(t)\}$. We will denote by $\mathcal{O}_x(t)$ the set of occurrences of the variables x in t, i.e., $\mathcal{O}_x(t) = \{v \in \mathcal{O}(t) \mid t(v) = x\}$, and with $\mathcal{O}_X(t)$ the set of occurrences of all the variables in X. A term t is* linear *if no variable occurs more than once in it, i.e., if $v, u \in \mathcal{O}_X(t)$ and $v \neq u$ implies $t(v) \neq t(u)$.* ∎

Definition 2.15 (Subterm Replacement) *Given terms $t, s \in CT_\Sigma$ and an occurrence $w \in \omega^*$, the* replacement *of s in t at (occurrence) w, denoted $t[w \leftarrow s]$, is the term defined as $t[w \leftarrow s](u) = t(u)$ if $w \not\leq u$ or $t/w = -$, and $t[w \leftarrow s](wu) = s(u)$ otherwise. Equivalently, subterm replacement can be defined as follows:*

- $t[w \leftarrow s] = t$ *if $t/w = -$ (i.e., if $w \notin \mathcal{O}(t)$);*
- $t[\lambda \leftarrow s] = s$ *if $t \neq -$;*
- $f(t_1, \ldots, t_n)[iw \leftarrow s] = f(t_1, \ldots, t_i[w \leftarrow s], \ldots, t_n)$ *if $i \leq n$;*
- $(\bigcup_{i<\omega}\{t_i\})\,[w \leftarrow s] = \bigcup_{i<\omega}\{t_i[w \leftarrow s]\}$.

The first clause also implies that $-[w \leftarrow s] = -$ for all w, s (even if $w = \lambda$). ∎

A particular relevance is due to *orthogonal* TRS's.

Definition 2.16 (Orthogonal TRS's) *Let $\mathcal{R}$ be a* TRS. *A rule $\langle l, r\rangle$ is left-linear if l is linear. $\mathcal{R}$ is* orthogonal *if all its rules are left-linear and* non-overlapping, *that is, the left-hand side of each rule does not unify with a non-variable subterm of any other rule in $\mathcal{R}$, or with a proper, non-variable subterm of itself.* ∎

The set-theoretical approach to term rewriting relies on the notion of *redex* (for *reducible expression*). A redex is just a pair $\Delta = (w, d)$ where w is the occurrence of the root of the subterm matching the left-hand side of the rule d.

Definition 2.17 (Redexes and Derivations) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *A* redex *Δ of t is a pair $\Delta = (w, d)$ where $w \in \omega^*$ is an occurrence, $d : l \to r \in R$ is a rule, and there exists a substitution $\sigma : var(l) \to CT_\Sigma$ such that $t/w = l\sigma$. The result of its* application *is $s = t[w \leftarrow r\sigma]$. We also write $t \to_\Delta s$, and we say that t* rewrites to *s (via Δ). We say that there is a* derivation *from t to t' if there are redexes $\Delta_1, \ldots, \Delta_n$ such that $t \to_{\Delta_1} t_1 \to_{\Delta_2} \ldots \to_{\Delta_n} t_n = t'$.* ∎

2.3.1 Parallel Rewriting

Sequential term rewriting can be generalized to parallel term rewriting by allowing for the simultaneous application of two or more redexes to a term. Clearly, the result of such a parallel rewriting must be well defined, and should be related in some way to the result obtained by applying the redexes in any order. The definitions below summarize those in [Bou85] (see also [LM92, Cor93]). Intuitively, *finite* parallel rewriting can be defined easily by exploiting the confluence of sets of (pairwise) *compatible* redexes. The parallel reduction of a finite set of such redexes is defined simply as any *complete development* of them: any such development ends with the same term, so the result is well-defined. Note however that, given two redexes of a term, the reduction of one of them can transform the other in various ways: the second redex can be destroyed, can be left intact, or can be copied a number of times; the situation is captured by the definition of *residual*.

Definition 2.18 (Compatible Redexes) *Let $\Delta = (w, d : l \to r)$ and $\Delta' = (w', d' : l' \to r')$ be two redexes in a term t. They are* disjoint *(and we write $\Delta || \Delta'$) if so are the respective occurrences (i.e., $w|w'$), or if they are equal. They are* compatible *(and we write $\Delta ||_c \Delta'$) if they are disjoint, or if $wu_x \leq w'$, where l/u_x is a variable, or if $w'u'_x \leq w$, where l'/u'_x is a variable.* ∎

Note that the definition is rather involved, due to the presence of *critical pairs* between the rules. This affects also the definition of residual: it is defined only for compatible redexes, since the reduction of a given redex in a critical pair "destroys" the possibility to reduce the other. Moreover, it is fundamental to take into account only *left-linear* TRS's: i.e., such that all the left-hand sides of the deduction rules are linear. Otherwise, it wouldn't be true that the reduction of a redex does not destroy the possibility to execute a compatible one: as an example, let us consider the TRS $\mathcal{Z} = \{d : f(x, x) \to g(x), d' : a \to b\}$: the redexes $\Delta = (\lambda, d)$ and $\Delta' = (1, d')$ of $t = f(a, a)$ are compatible according to Definition 2.18, but the reduction of Δ' forbids to reduce Δ (until also $(2, d')$ is executed).

If we take into account disjoint rewrites only, instead, left-linearity can be dropped. In the rest of the section we implicitly assume that, when dealing with *compatibility*, all the TRS's are left-linear; this restriction is not applied when dealing with *disjointness*.

Definition 2.19 (Residuals) *Let $\Delta = (w, d)$ and $\Delta' = (w', d' : l' \to r')$ be two compatible redexes in a term t. The* set of residuals of Δ by Δ', *denoted by $\Delta \backslash \Delta'$, is defined as:*

$$\Delta\backslash\Delta' = \begin{cases} \emptyset & \text{if } \Delta = \Delta'; \\ \{\Delta\} & \text{if } w \not> w'; \\ \{(w'w_x u, d) \mid r'/w_x = l'/v_x\} & \text{if } w = w'v_x u \text{ and } l'/v_x \text{ is a variable.} \end{cases}$$

□

Note that $\Delta\backslash\Delta'$ can actually be a set of redexes, whenever the rule d' is not right-linear. As an example, let us consider the TRS $\mathcal{Z}' = \{d : f(x) \to g(x, x), d' : a \to b\}$ and the compatible redexes $\Delta = (1, d')$, $\Delta' = (\lambda, d)$: then $\Delta/\Delta' = \{(1, d'), (2, d')\}$. When two redexes Δ, Δ' are disjoint, the third case never happens.

Proposition 2.12 (Reduction Preserves Compatibility) *Let $\Phi \cup \{\Delta\}$ be a finite set of pairwise compatible redexes of t, such that $t \to_\Delta s$. Then the set $\Phi\backslash\Delta$ of residuals of Φ by Δ, defined as the union of $\Delta'\backslash\Delta$ for all $\Delta' \in \Phi$, is still compatible. Moreover, each $\Delta'\backslash\Delta$ is a redex in s.* □

The previous result obviously holds for sets of disjoint redexes, and it allows to extend the definition of residual to include also sequences of reductions.

Definition 2.20 (Residual of a Sequence) *Let Φ be a finite set of pairwise compatible redexes of t and $\rho = (t \to_{\Delta_1} t_1 \ldots \to_{\Delta_n} t_n)$ be a reduction sequence, such that $\Phi \cup \{\Delta_1\}$ is compatible. Then $\Phi\backslash\rho$ is defined as Φ if $n = 0$, and as $(\Phi\backslash\Delta_1)\backslash\rho'$, where $\rho' = (t_1 \to_{\Delta_2} t_2 \ldots \to_{\Delta_n} t_n)$, otherwise.* □

Note that the residual of a sequence is not always defined. It is necessary that, at each step i, the redex Δ_{i+1} is compatible with $((\ldots((\Phi\backslash\Delta_1)\backslash\Delta_2)\ldots)\backslash\Delta_i)$.

Definition 2.21 (Complete Development) *Let Φ be a finite set of pairwise compatible redexes of t. A* development of Φ *is a reduction sequence such that after each initial segment ρ, the next reduced redex is an element of $\Phi\backslash\rho$. A* complete development of Φ *is a development ρ such that $\Phi\backslash\rho = \emptyset$.* □

The complete development of a set of compatible redexes is well-defined, due to the result stated in Proposition 2.12. The following result was originally proved in [Bou85].

Proposition 2.13 *All complete developments ρ and ρ' of a finite set of pairwise compatible redexes Φ in a term t are finite, and end with the same term. Moreover, for each redex Δ of t, compatible with those in Φ, it holds $\Delta\backslash\rho = \Delta\backslash\rho'$. Therefore we can safely denote by $\Delta\backslash\Phi$ the residuals of Δ by any complete development of Φ (and similarly replacing Δ with a finite set of compatible redexes Φ' of t).* □

Exploiting this result, we define the parallel reduction of a finite set of compatible redexes as any complete development of them.

Definition 2.22 (Parallel Redex Reduction) *A* parallel redex Φ *of a term t is a finite set of pairwise compatible redexes in t. We write $t \to_\Phi t'$ and say that there is a* parallel reduction *from t to t' if there exists a complete development $t \to_{\Delta_1} t_1 \ldots \to_{\Delta_n} t'$ of Φ.* □

Obviously, all the results can be lifted to disjoint sets of redexes: we indicate this case as *disjoint reduction.* Despite its straightforward definition, disjoint reduction will play a fundamental röle in our analysis of the *concurrency* of the reduction process (see Chapter 6). Instead, parallel reduction will be pivotal when defining *infinitary term rewriting* (next section and Chapter 7).

Definition 2.23 (Disjoint Redex Reduction) *A* disjoint redex Φ *of a term t is a finite set of pairwise disjoint redexes in t. We write $t \to_\Phi t'$ and say that there is a* disjoint reduction *from t to t' if there exists a complete development $t \to_{\Delta_1} t_1 \ldots \to_{\Delta_n} t'$ of Φ.* □

2.3.2 Infinite Parallel Rewriting

Parallel rewriting allows to reduce a finite set of redexes of a term in a single, parallel step. If we consider an infinite term, there might be infinitely many distinct redexes in it: since the simultaneous rewriting of any finite subset of those redexes is well-defined, by a continuity argument one would expect that also the simultaneous rewriting of infinitely many redexes in an infinite term can be properly defined. Note however that, since in the finite case the parallel application of a redex Φ is defined as the sequential application of

all the contained redexes, a naïve extension to infinity could not work, because it would correspond to an infinite sequence of reductions. We present here a definition which makes use of a suitable limit construction: for details we refer to [Cor93, CD96]. Note however that, for the sake of simplicity, we restrict our attention to orthogonal TRS's. In fact, for a given orthogonal TRS the residual operation is total, since any two redexes Δ, Δ' are the same or do not overlap: then, any set of redexes is compatible.

Given an infinite parallel redex Φ (i.e., an infinite set of redexes) of a term t, we consider a chain of finite approximations of t, $t_0 \leq t_1 \leq t_2 \ldots$ such that their limit is t. For each $i < \omega$, let Φ_i be the finite subset of Φ containing all and only those redexes of t which are also redexes of t_i, and call s_i the result of the parallel reduction of Φ_i, i.e., $t_i \to_{\Phi_i} s_i$. Then the crucial fact is that the sequence of terms $s_0, s_1, s_2, \ldots$ defined in this way forms a chain: by definition we say that there is an infinite parallel reduction from t to $s = \bigcup_{i<\omega} s_i$ via Φ, written $t \to_\Phi s$. Here is the formal definition.

Definition 2.24 (Infinite Parallel Redex Reduction) *Given an infinite parallel redex Φ of a term t, let $t_0 \leq t_1 \leq t_2 \ldots$ be any chain of finite approximations of t, such that for each $i < \omega$, every redex $(w, d) \in \Phi$ is either a redex of t_i or $t_i(w) = -$ (that is, the image of the left-hand side of every redex in Φ is either all in t_i, or it is outside, but does not "cross the boundary"). Let Φ_i be the subset of all redexes in Φ which are also redexes of t_i, and let s_i be the result of the (finite) parallel reduction of t_i via Φ_i (i.e., $t_i \to_{\Phi_i} s_i$). Then we say that there is an* (infinite) parallel reduction *from t to $s \stackrel{def}{=} \bigcup_{i<\omega}\{s_i\}$ via Φ, and we write $t \to_\Phi s$.* □

Let us consider the TRS $\mathcal{V} = \{d : f(x) \to g(x), d' : a \to b\}$. Then the infinite parallel redex $\{(1^*, d)\}$ can be applied to the infinite term $t = f^\omega = \bigcup_{i<\omega}\{f^i(-)\}$: a suitable chain of finite approximations is given by $t_i = f^i(-)$, and the associated subset Φ is $\{(1^j, d) \mid j \leq i\}$. Then $t_i \to_{\Phi_i} g^i(-)$, and $t \to_\Phi g^\omega$. Next result (originally proved in [Cor93]) states that the reduction of an infinite, parallel redex is a well-given definition.

Proposition 2.14 (Infinite Parallel Redex Reduction is Well-Defined) *In the hypotheses of Definition 2.24:*

1. *for each $i < \omega$, $s_i \leq s_{i+1}$; i.e., $\{s_i\}_{i<\omega}$ is a chain.*
2. *Definition 2.24 is well-given; i.e., the result of the infinite parallel reduction of t via Φ does not depend on the choice of the chain approximating t, provided that it satisfies the required conditions.* □

Finally, we shall need the following easy result, stating the compatibility of finite and infinite parallel reduction.

Proposition 2.15 (Strong Confluence of Parallel Reduction) *Let $\mathcal{R}$ be an orthogonal* TRS. *Then parallel reduction is strongly confluent, i.e., if $t' \ {}_{\Phi'}\!\leftarrow t \rightarrow_{\Phi} t''$ for (eventually infinite) sets Φ, Φ' of redexes, then there exist t''', Ψ, Ψ' such that $t' \rightarrow_{\Psi'} t''' \ {}_{\Psi}\!\leftarrow t''$. Hence, parallel reduction is confluent.* ∎

Chapter 3

Some Notions of Category Theory

In the first section we briefly recall some basic concepts of category theory; except for Section 3.2 (where the notion of *s-monoidal category* is an original one) we refer the interested reader to [ML71]. The rest of the chapter is devoted to provide an introduction to categorical structures like *2-categories* and *algebraic theories*, that are not so common in the computer science community.

3.1 Basic Definitions

We first introduce the underlying notion of *graph*.

Definition 3.1 (Graphs) *A* graph G *is a 4-tuple* $\langle O_G, A_G, \delta_0, \delta_1 \rangle$ *where* O_G, A_G *are classes whose elements are called respectively* objects *and* arrows *(ranged over by* $a, b, \ldots$ *and* $f, g, \ldots$*), and* $\delta_0, \delta_1 : A_G \to O_G$ *are functions, called respectively* source *and* target.
□

A graph is *small* if its arrows form a set; it is *locally small* if for each pair of objects a, b, the *hom-set* $G[a, b]$ (i.e., the class of arrows from a to b) forms a set. Let G_1, G_2 be two graphs. A *graph morphism* $\tau : G_1 \to G_2$ is a couple of functions $\tau_A : A_1 \to A_2$, $\tau_O : O_1 \to O_2$ preserving source and target. The *product* $G_1 \times G_2$ is the graph $\langle O_1 \times O_2, A_1 \times A_2, \delta_0', \delta_1' \rangle$: its components are given by the cartesian product of the underlying classes, while δ_0', δ_1' are defined pointwise. A graph *with pairing* is a graph G such that its class of objects form a monoid $\langle O_G, \otimes, 1 \rangle$: $\otimes : O_G \times O_G \to O_G$ is an associative function, and 1 is a distinguished element such that $a \otimes 1 = 1 \otimes a = a$ for all $a \in O_G$. A *reflexive* graph is a graph G equipped with a function $id_G : O_G \to A_G$ such that $\delta_0(id_G(a)) = \delta_1(id_G(a)) = a$

for all $a \in O_G$. The set of *composable arrows* $A \times_0 A$ of a graph G is given by the subset of $A \times A$ satisfying $\{\langle f, g\rangle \mid \delta_1(f) = \delta_0(g)\}$.

A *category* is obtained simply enriching the structure of a graph, in order to describe *composition* of arrows.

Definition 3.2 *A category* $\mathbf{C}$ *is a 6-tuple* $\langle O_C, A_C, \delta_0, \delta_1, id_C, ;_C \rangle$ *(we forget the subscript when there is no ambiguity) where* $G_C = \langle O_C, A_C, \delta_0, \delta_1, id_C\rangle$ *is a reflexive graph, and* $;_C$ (composition) *is a function* $;_C : A_C \times_0 A_C \to A_C$ *satisfying:*

- $\delta_0(f ;_C g) = \delta_0(f)$ *and* $\delta_1(f ;_C g) = \delta_1(g)$*, for* $f, g \in A_C$*;*
- $;_C$ *is associative, i.e.,* $(f ;_C g) ;_C h = f ;_C (g ;_C h)$ *for* $f, g, h \in A_C$*;*
- $f ;_C id_C(b) = id_C(a) ;_C f = f$*, for* $f \in A_C$*,* $\delta_0(f) = a$*,* $\delta_1(f) = b$*.* ∎

All the various classes of algebras introduced in the previous chapter form categories, since the identity function is a continuous Σ-homomorphism and (ordered, continuous) Σ-homomorphisms are closed with respect to functional composition. The paradigmatic example of category is **Set**, the category of sets and functions. A category is *discrete* if all its arrows are identity arrows: interesting examples are $\mathbf{0}$, the empty category, and $\mathbf{1}$, the category with one object and one arrow.

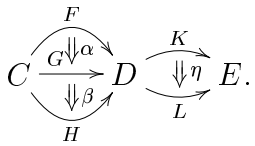
Definition 3.3 (Functors and Natural Transformations) *Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be two categories. A* functor $F : \mathbf{C} \to \mathbf{D}$ *is a graph morphism* $\langle F_O, F_A\rangle$ *preserving identities and composition:*

- $F_A(id_C(a)) = id_D(F_O(a))$*, for* $a \in O_C$*;*
- $F_A(f ;_C g) = F_A(f) ;_D F_A(g)$*, for* $f, g \in A_C$*.*

Given two functors $F, G : \mathbf{C} \to \mathbf{D}$*, a* transformation $\eta : F \Rightarrow G$ *is a function* $\alpha : O_C \to A_D$ *such that* $\eta(a) \in \mathbf{D}[F_O(a), G_O(a)]$*. A transformation* $\eta : F \Rightarrow G : \mathbf{C} \to \mathbf{D}$ *is* natural *if the identity* $F_A(f) ;_D \eta(b) = \eta(a) ;_D G_A(f)$ *holds for all* $f : a \to b \in A_C$*; or, equivalently, if the following diagram commutes:*

$$\begin{array}{ccc} a & & F(a) \xrightarrow{\eta(a)} G(a) \\ {\scriptstyle f}\downarrow & & {\scriptstyle F(f)}\downarrow \qquad\qquad \downarrow{\scriptstyle G(f)} \\ b & & F(b) \xrightarrow[\eta(b)]{} G(b) \end{array}$$

Now let us consider the situation denoted by the following transformations:



The vertical composition $\alpha \cdot \beta : F \Rightarrow H : \mathbf{C} \to \mathbf{D}$ *is defined as* $(\alpha \cdot \beta)(a) = \alpha(a);_D \beta(a)$ *for every* $a \in O_C$.

The right composition $\alpha *_R K : FK \Rightarrow GK : \mathbf{C} \to \mathbf{E}$ *is defined as* $(\alpha *_R K)(a) = K_A(\alpha(a))$ *for every* $a \in C_O$; *the* left composition $(F *_L \eta) : FK \Rightarrow FL : \mathbf{C} \to \mathbf{E}$ *is defined as* $(F *_L \eta)(a) = \eta(F_O(a))$ *for every* $a \in O_C$.

If α, η *are natural, then the* horizontal composition $\alpha * \eta : FK \Rightarrow GL : \mathbf{C} \to \mathbf{E}$ *is defined as* $(\alpha * \eta)(a) = K_A(\alpha(a));_E \eta(G_O(a)) = \eta(F_O(a));_E L_A(\alpha(a))$ *for every* $a \in O_C$. ∎

A subcategory $\mathbf{D}$ of $\mathbf{C}$ is a category whose arrows and objects form subclasses of those in $\mathbf{C}$ or, equivalently, if there exists an *inclusion functor* $In : \mathbf{D} \hookrightarrow \mathbf{C}$ such that the underlying graph morphism $In_G : G_D \to G_C$ is the identity function on arrows and nodes. An inclusion functor is *full* if, for all objects $a, b \in \mathbf{D}$, $\mathbf{D}[a, b] = \mathbf{C}[a, b]$. Given categories $\mathbf{C}$ and $\mathbf{D}$, the *product category* $\mathbf{C} \times \mathbf{D}$ has as underlying graph the product $G_C \times G_D$, while composition and identity are defined pointwise.

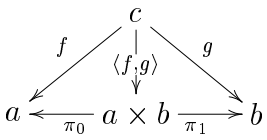
Definition 3.4 (Functor Category) *Let* $\mathbf{C}$ *and* $\mathbf{D}$ *be two locally small categories. The* functor category $[\mathbf{C} \to \mathbf{D}]$ *is defined*[1] *as follows:*

- *objects of* $[\mathbf{C} \to \mathbf{D}]$ *are functors* $F : \mathbf{C} \to \mathbf{D}$;
- *an arrow* $\alpha : F \Rightarrow G$ *between two parallel functors* $F, G : \mathbf{C} \to \mathbf{D}$ *is a* natural transformation. ∎

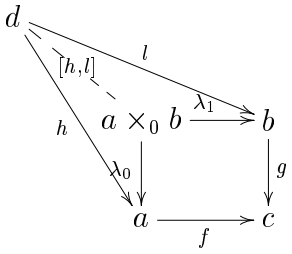
One of the main features of categories are *universal constructions*: they allow to characterize some elements as those satisfying "in a unique way" suitable properties.

[1] In order for the functor category $[\mathbf{C} \to \mathbf{D}]$ to be defined, $\mathbf{C}$ and $\mathbf{D}$ *must be* locally small. We will not deal with foundational issues, assuming that our categories are locally small whenever this requirement is necessary.

Definition 3.5 (Product, Terminal Object, Pullback) *Let* $\mathbf{C}$ *be a category. An object* 1 *in* $\mathbf{C}$ *is* terminal *if for every object* a *in* $\mathbf{C}$ *there exists a unique arrow* $!_a : a \to 1$*, while an object* 0 *is* initial *if for every object* a *in* $\mathbf{C}$ *there exists a unique arrow* $I_a : 0 \to a$ *The* product *of a pair* $\langle a, b\rangle$ *of objects in* $\mathbf{C}$ *is a triple* $\pi_{a,b} = \langle a \times b, \pi_0 : a \times b \to a, \pi_1 : a \times b \to b\rangle$ *such that, given a pair of arrows* $\langle f : c \to a, g : c \to b\rangle$*, there is a unique arrow* $\langle f, g\rangle : c \to a \times b$ *satisfying* $\langle f, g\rangle ;_C \pi_0 = f$ *and* $\langle f, g\rangle ;_C \pi_1 = g$*; or, equivalently, making the following diagram commute:*



The pullback *of* a *(or* along a*)* *pair of arrows* $\langle f : a \to c, g : b \to c\rangle$ *is a triple* $pb_{f,g} = \langle a \times_0 b, \lambda_0 : a \times_0 b \to a, a \times_0 b \to b\rangle$ *verifying* $\lambda_0 ;_C f = \lambda_1 ;_C g$ *and such that, given a pair of arrows* $\langle h : d \to a, l : d \to b\rangle$ *satisfying* $h ;_C f = l ;_C g$*, there is a unique arrow* $[h, l] : d \to a \times_0 b$ *satisfying* $[h, l]; \lambda_0 = h$ *and* $[h, l]; \lambda_1 = l$*; or, equivalently, making the following diagram commute:*



□

Usually, the *universal property* characterizes an object only *up-to-isomorphism*: for example, there may exist many elements satisfying the terminal object property, but they are isomorphic by a canonical isomorphism. In particular, if both 1 and $1'$ satisfy the universal property of terminal object in a category $\mathbf{C}$, then $!_{1'} ;_C !'_1 = id_1$ and $!'_1 ;_C !_{1'} = id_{1'}$. A universal construction is *strict* if it is uniquely determined, in the sense that the canonical isomorphism is actually an identity: for the terminal object, e.g., we require $!_{1'} = !'_1 = id_1$. More generally, we say that a universal construction is *on-the-nose* if we characterize each class of elements satisfying the conditions of the universal construction by a canonical representative.

Definition 3.6 (Cartesian Category) *A category* $\mathbf{C}$ *is* cartesian *if it has terminal object* 1 *and product* $\pi_{a,b}$ *for each pair of objects* a, b*. A functor* $F : \mathbf{C} \to \mathbf{D}$ *between*

cartesian categories is cartesian *if it preserves products and terminal objects; i.e., if the canonical morphisms* $\langle F(\pi_0), F(\pi_1)\rangle : F(a \times_C b) \to F(a) \times_D F(b)$ *and* $!_{F(1_C)} : F(1_C) \to 1_D$ *are isomorphisms.* ∎

Definition 3.7 (Monoidal Categories) *A* monoidal category $\mathbf{C}$ *is a 6-tuple of the kind* $\langle \mathbf{C}_0, \otimes, e, \alpha, \eta_l, \eta_r\rangle$*, where* $\mathbf{C}_0$ *is a category,* $e \in \mathbf{C}_0$ *and* $\otimes : \mathbf{C}_0 \times \mathbf{C}_0 \to \mathbf{C}_0$ *are functors and* $\alpha : (-\otimes =) \otimes + \Rightarrow - \otimes (= \otimes +)$*,* $\eta_l : e \otimes - \Rightarrow -$*,* $\eta_r : - \otimes e \Rightarrow -$ *are natural isomorphisms, satisfying the following coherence axioms:*

$$\begin{array}{ccccc} a \otimes (b \otimes (c \otimes d)) & \xrightarrow{\alpha(a,b,c\otimes d)} & (a \otimes b) \otimes (c \otimes d) & \xrightarrow{\alpha(a\otimes b,c,d)} & ((a \otimes b) \otimes c) \otimes d \\ \downarrow{\scriptstyle a\otimes\alpha(b,c,d)} & & & & \uparrow{\scriptstyle \alpha(a,b,c)\otimes d} \\ a \otimes ((b \otimes c) \otimes d) & & \xrightarrow[\alpha(a,b\otimes c,d)]{} & & (a \otimes (b \otimes c)) \otimes d \end{array}$$

$$\begin{array}{ccc} a \otimes (e \otimes b) & \xrightarrow{\alpha(a,e,b)} & (a \otimes e) \otimes b \\ & {\scriptstyle a\otimes\eta_l(b)}\searrow & \downarrow{\scriptstyle \eta_r(a)\otimes b} \\ & & a \otimes b \end{array}$$

(where for the sake of readability we indicate the identity of an object with the object itself). *A* symmetric monoidal category *is a 7-tuple* $\langle \mathbf{C}_0, \otimes, e, \alpha, \eta_l, \eta_r, \rho\rangle$ *where* $\langle \mathbf{C}_0, \otimes, e, \alpha, \eta_l, \eta_r\rangle$ *is a monoidal category, and* $\rho : - \otimes + \Rightarrow + \otimes -$ *is a natural isomorphism satisfying the coherence axioms*

$$\begin{array}{ccccc} a \otimes (b \otimes c) & \xrightarrow{\alpha(a,b,c)} & (a \otimes b) \otimes c & \xrightarrow{\rho(a\otimes b,c)} & c \otimes (a \otimes b) \\ \downarrow{\scriptstyle a\otimes\rho(b,c)} & & & & \downarrow{\scriptstyle \alpha(c,a,b)} \\ a \otimes (c \otimes b) & \xrightarrow[\alpha(a,c,b)]{} & (a \otimes c) \otimes b & \xrightarrow[\rho(a,c)\otimes b]{} & (c \otimes a) \otimes b \end{array}$$

$$\begin{array}{ccc} e \otimes a & \xrightarrow{\rho(e,a)} & a \otimes e \\ & {\scriptstyle \eta_l(a)}\searrow & \downarrow{\scriptstyle \eta_r(a)} \\ & & a \end{array} \qquad \begin{array}{ccc} a \otimes b & \xrightarrow{\rho(a,b)} & b \otimes a \\ & {\scriptstyle a\otimes b}\searrow & \downarrow{\scriptstyle \rho(b,a)} \\ & & a \otimes b \end{array}$$

A monoidal functor $F : \mathbf{C} \to \mathbf{C}'$ *is a triple* $\langle F_0, \nu, \phi\rangle$ *where* $F_0 : \mathbf{C}_0 \to \mathbf{C}_1$ *is a functor, and* $\nu : F_0(e) \Rightarrow e'$ *and* $\phi : F_0(- \otimes +) \Rightarrow F_0(-) \otimes' F_0(+)$ *are natural isomorphisms, satisfying the axioms:*

$$\begin{array}{ccccc} F(a\otimes(b\otimes c)) & \xrightarrow{\phi(a,b\otimes c)} & F(a)\otimes' F(b\otimes c) & \xrightarrow{F(a)\otimes'\phi(b,c)} & F(a)\otimes'(F(b)\otimes' F(c)) \\ \downarrow{\scriptstyle F(\alpha(a,b,c))} & & & & \downarrow{\scriptstyle \alpha'(F(a),F(b),F(c))} \\ F((a\otimes b)\otimes c) & \xrightarrow[\phi(a\otimes b,c)]{} & F(a\otimes b)\otimes' F(c) & \xrightarrow[\phi(a,b)\otimes' F(c)]{} & (F(a)\otimes' F(b))\otimes' F(c) \end{array}$$

$$\begin{array}{ccc} F(e\otimes a) & \xrightarrow{\phi(e,a)} & F(e)\otimes' F(a) \\ \downarrow{\scriptstyle F(\eta_l(a))} & & \downarrow{\scriptstyle \nu\otimes' F(a)} \\ F(a) & \xleftarrow{\eta_l'(F(a))} & e'\otimes' F(a) \end{array} \qquad \begin{array}{ccc} F(a\otimes e) & \xrightarrow{\phi(a,e)} & F(a)\otimes' F(e) \\ \downarrow{\scriptstyle F(\eta_r(a))} & & \downarrow{\scriptstyle F(a)\otimes'\nu} \\ F(a) & \xleftarrow{\eta_r'(F(a))} & F(a)\otimes' e' \end{array}$$

(where we omitted subscripts for the sake of readability). A monoidal functor is symmetric *if moreover:*

$$\begin{array}{ccc} F(a\otimes b) & \xrightarrow{\phi(a,b)} & F(a)\otimes' F(b) \\ \downarrow{\scriptstyle F(\rho(a,b))} & & \downarrow{\scriptstyle \rho'(F(a),F(b))} \\ F(b\otimes a) & \xrightarrow[\phi(b,a)]{} & F(b)\otimes' F(a) \end{array}$$

A (symmetric) monoidal (natural) transformation $\beta : F \Rightarrow G$ *between (symmetric) monoidal functors F, G is a (natural) transformation satisfying:*

$$\begin{array}{ccc} F(a\otimes b) & \xrightarrow{\phi(a,b)} & F(a)\otimes' F(b) \\ \downarrow{\scriptstyle \beta(a\otimes b)} & & \downarrow{\scriptstyle \beta(a)\otimes\beta(b)} \\ G(a\otimes b) & \xrightarrow[\phi'(a,b)]{} & G(a)\otimes' G(b) \end{array} \qquad \begin{array}{ccc} F(e) & \xrightarrow{\beta(e)} & G(e) \\ & {\scriptstyle \nu}\searrow & \downarrow{\scriptstyle \nu'} \\ & & e' \end{array}$$

□

All the coherence properties required for the monoidal version of the definition of functor and (natural) transformation can be summed up simply saying that, in the richer context, all the underlying definitions must preserve also the new relevant structure (and this fact will be fundamental in the next sections). In the following, a (symmetric) monoidal functor is usually indicated with the underlying (symmetric) functor whenever the associated natural isomorphisms are identities. Moreover, we will denote as *strict monoidal* all those monoidal categories such that the associated natural isomorphisms α, η_l, η_r are identities; note that in this case most of the coherence axioms collapse.

Definition 3.8 (Adjoint Functors) *Let* $\mathbf{C}$, $\mathbf{D}$ *be categories. An* adjunction *between* $\mathbf{C}$ *and* $\mathbf{D}$ *is a triple* $\langle F, G, \phi\rangle$ *where* $F : \mathbf{C} \to \mathbf{D}$, $G : \mathbf{D} \to \mathbf{C}$ *are functors, and* ϕ *is a function which assigns to each pair of objects* $c \in \mathbf{C}, d \in \mathbf{D}$ *a bijection* $\phi_{c,d} : \mathbf{C}[c, G(d)] \cong \mathbf{D}[F(c), d]$

which is natural both in c and d, i.e., such that for all $f : c' \to c, g : d \to d'$*, the following diagrams commute:*

$$\begin{array}{ccc} \mathbf{D}[F(c),d] & \xrightarrow{\phi_{c,d}} & \mathbf{C}[c,G(d)] \\ {\scriptstyle F(f);\perp}\downarrow & & \downarrow{\scriptstyle f;\perp} \\ \mathbf{D}[F(c'),d] & \xrightarrow[\phi_{c',d}]{} & \mathbf{C}[c',G(d)] \end{array} \qquad \begin{array}{ccc} \mathbf{D}[F(c),d] & \xrightarrow{\phi_{c,d}} & \mathbf{C}[c,G(d)] \\ {\scriptstyle \perp;g}\downarrow & & \downarrow{\scriptstyle \perp;G(g)} \\ \mathbf{D}[F(c),d'] & \xrightarrow[\phi_{c,d'}]{} & \mathbf{C}[c,G(d')] \end{array}$$

We say that F *is a* left-adjoint *of* G *(*G *is a* right-adjoint *of* F*) and we write* $F \triangleleft G$*. We indicate as* unit *of the adjunction the natural transformation* $\rho : Id_C \Rightarrow G(F(-))$*, associating to each object* $c \in \mathbf{C}$ *the arrow associated by* $\phi_{c,F(c)}$ *to* $id(F(c))$*; and we indicate as* co-unit *of the adjunction the natural transformation* $\eta : F(G(-)) \Rightarrow Id_D$*, associating to each object* $c \in \mathbf{C}$ *the arrow associated by* $\phi_{G(d),d}$ *to* $id(G(d))$*.* ∎

We say that a category $\mathbf{C}$ is *reflective* inside a category $\mathbf{D}$ if $\mathbf{C}$ is a sub-category of $\mathbf{D}$, the inclusion functor has a left-adjoint and the co-unit is a natural isomorphism. A very particular case is represented by *forgetful* functors, i.e., those functors that "forget" part of the structure of the source category; they usually have a left-adjoint that simply "adds" the relevant structure with a "free" construction. An intuitive example is represented by the inclusion functor of $\mathbf{Cat}$ in $\mathbf{Gr}$, the category of small graphs and graph-morphisms: the left-adjoint simply adds the identity function and the composition function, requiring they satisfy the axioms of categories. A relevant example is the adjunction arising from the inclusion functors $\mathbf{Cpo}_S \hookrightarrow \mathbf{Cpo} \hookrightarrow \mathbf{Set}$, where $\mathbf{Cpo}$ is the category of small CPO's and continuous functors, and $\mathbf{Cpo}_S$ is the category of small, strict CPO's and strict continuous functors. The following result relates initial objects and adjoint functors.

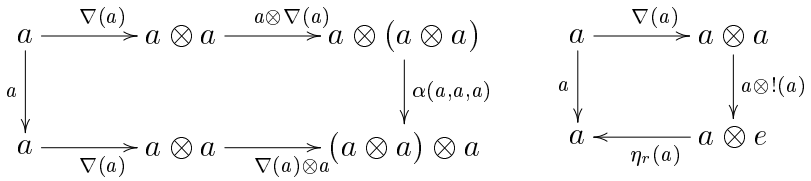
Proposition 3.1 *Let* $F : \mathbf{C} \to \mathbf{D}$ *be a functor with a left-adjoint* G*. If* 0 *is initial in* $\mathbf{D}$*, then* $G(0)$ *is initial in* $\mathbf{C}$*.* ∎

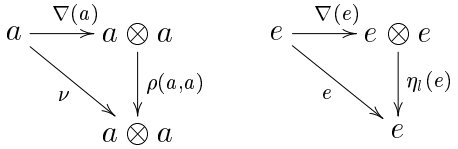
3.2 Cartesianity as Enriched Monoidality

In the latest years there has been some interest in getting suitable equational characterization of cartesian categories (see e.g. [Bur91, Laf95]). In this section we try to recast the previous results in a more general framework; our formalism is indebted to [Jac93].

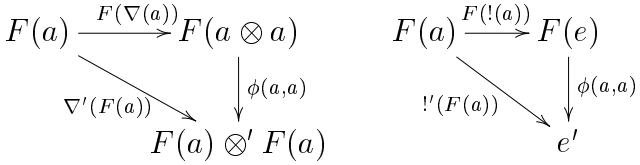
Definition 3.9 (S-Monoidal Categories) *A* s-monoidal category *is a 9-tuple of the kind* $\langle \mathbf{C}_0, \otimes, e, \alpha, \eta_l, \eta_r, \rho, \nabla, ! \rangle$*, where* $\langle \mathbf{C}_0, \otimes, e, \alpha, \eta_l, \eta_r, \rho \rangle$ *is a symmetric monoidal cate-*

gory, and $\nabla : Id \Rightarrow \langle - \otimes +, id, - \otimes \rho(+,-) \otimes - \rangle$ *and* $! : Id \Rightarrow e$ *are symmetric monoidal transformations satisfying the coherence axioms:*





A s-monoidal functor *is a symmetric monoidal functor such that the following diagram commutes:*



An s-monoidal (natural) transformation *between s-monoidal functors is a symmetric (natural) transformation.* ∎

The monoidality of the transformations amounts to say that $!(a \otimes b) = !(a) \otimes !(b)$ and $\nabla(a \otimes b) = (\nabla(a) \otimes \nabla(b)); (a \otimes \rho(a,b) \otimes b)$. As we remarked before, different versions of the following results became part of the categorical "folklore" in the past years.

Proposition 3.2 (Cartesianity as Enriched Monoidality) *Let* $\mathbf{D}$ *be an s-monoidal category. If both the associated transformations* ∇ *and* $!$ *are natural, then* $\mathbf{D}$ *is a cartesian category.* ∎

In the thesis, using a non-standard notation, we say that a triple $\mathbf{D} = \langle \mathbf{C}, \nabla, ! \rangle$ where $\mathbf{C}$ is a symmetric monoidal category and both $\nabla, !$ are natural is a cartesian category *with chosen products*; it is a cartesian category with *finite* products if the underlying monoidal category $\mathbf{C}$ is strict monoidal. A chosen functor $F : \mathbf{D} \to \mathbf{E}$ between cartesian categories with (finite) chosen products is a s-monoidal functor between the underlying (strict) s-monoidal categories; a chosen functor is also cartesian.

Both **Set** and **Cat** can be equipped with a finite products structure: for each set S (category $\mathbf{C}$), the n-th product is given by the set S^n (category $\mathbf{C}^n$) where the objects are n-tuples of objects in S, and the functions are defined pointwise.

In the rest of the chapter (actually, in the whole thesis), particular importance will be played by the following adjunction:

$$\begin{array}{ccccc} \mathbf{GR}_p & \rightleftarrows & \mathbf{SSM\text{-}Cat} & \rightleftarrows & \mathbf{SM\text{-}Cat} \\ & & \uparrow\downarrow & & \uparrow\downarrow \\ & & \mathbf{FC\text{-}Cat} & \leftrightarrows & \mathbf{CC\text{-}Cat} \end{array}$$

relating the category of small graphs with pairing $\mathbf{GR}_p$ (and graph morphisms preserving pairing), the category of (strict) s-monoidal small categories **SM-Cat** (**SSM-Cat**) and s-monoidal functors, and the category of cartesian small categories with chosen (finite) products **CC-Cat** (**FC-Cat**) and chosen functors.

3.3 Cat-Enriched Categories

Given a monoidal category $\mathbf{V}$, a $\mathbf{V}$-*category* (or a *category enriched over* $\mathbf{V}$) is basically just a category such that, for any two objects a, b, the hom-set $\mathbf{C}[a, b]$, i.e., the class of arrows from a to b is an object of $\mathbf{V}$; moreover, this hom-objects satisfy suitable coherence axioms. In particular, a *cat-enriched* category is a category $\mathbf{C}$ such that, given any two objects a, b, the hom-set $\mathbf{C}[a, b]$ is a category. Both *2-categories* and *sesqui-categories* are particular examples of cat-enriched categories[2], and they admit also a naïve description: for 2-categories, the classical reference is [KS74].

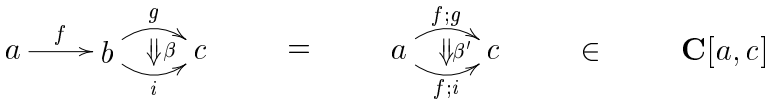
Let us fix some notation. An arrow of the category $\mathbf{C}[a, b]$, a *cell*, is denoted as $\alpha : f \Rightarrow g : a \to b$, with source $\delta_0(\alpha) = f$, target $\delta_1(\alpha) = g$ and where $f, g : a \to b$; or graphically, as

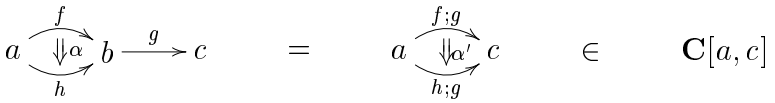
$$a \overset{f}{\underset{g}{\Downarrow \alpha}} b.$$

The following definition is adapted from [Ste92].

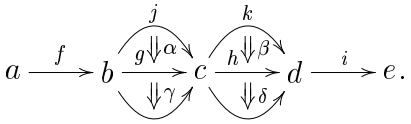
[2]We are quite informal here, since both 2-categories and sesqui-categories are categories enriched over **Cat**, in the sense of [Kel82], but the tensor product used is different in the two cases (see [Ste94]). For 2-categories is the usual cartesian product, while for sesqui is the so-called *funny tensor*: we refer the reader to [Str92] for a comprehensive introduction.

Definition 3.10 (2-Categories and Sesqui-Categories) *Let* $\mathbf{C}$ *be a category such that each hom-set* $\mathbf{C}[a,b]$ *also forms a category. Moreover, let us assume that for each triple* a,b,c *of objects there are two composition functions* $*_L$ *and* $*_R$ *such that, given* $\alpha : f \Rightarrow h : a \to b$ *and* $\beta : g \Rightarrow i : b \to c$*, then* $\beta' = f *_L \beta \in \mathbf{C}[a,c]$ *and* $\alpha' = \alpha *_R g \in \mathbf{C}[a,c]$*. Graphically,*





where ; *denotes composition inside* $\mathbf{C}$*. Let us consider the situation denoted by the following cells:*



A 2-category $\underline{\mathbf{C}}_2$ *(or simply* $\underline{\mathbf{C}}$*) is a category* $\mathbf{C}$ *(called the* underlying *category) with a structure as the one defined above, such that the composition functions are subject to the following equations:*

(1) $id_c *_L \beta = \beta$; (2) $(f;g) *_L \beta = f *_L (g *_L \beta)$;

(3) $g *_L id_h = id_{g;h}$; (4) $g *_L (\beta \cdot \delta) = (g *_L \beta) \cdot (g *_L \delta)$;

(5) $\alpha *_R id_c = \alpha$; (6) $\alpha *_R (h;i) = (\alpha *_R h) *_R i$;

(7) $id_g *_R h = id_{g;h}$; (8) $(\alpha \cdot \gamma) *_R h = (\alpha *_R h) \cdot (\gamma *_R h)$;

(9) $(f *_L \alpha) *_R h = f *_L (\alpha *_R h)$;

(10) $(j *_L \beta) \cdot (\alpha *_R h) = (\alpha *_R k) \cdot (g *_L \beta)$.

where $\cdot$ *denotes composition inside hom-categories. A* sesqui-category $\underline{\mathbf{C}}_S$ *(or simply* $\underline{\mathbf{C}}$*:*

usually, there is no ambiguity) is a category $\mathbf{C}$ *with a structure subject to equations (1)-(9).* □

Note also that, thanks to axiom (10), it is possible to define the notion of *horizontal* composition of cells, so that the given definition for 2-category is then equivalent to the classical one (see [KS74])

$$\alpha * \beta = (j *_L \beta) \cdot (\alpha *_R h) = (\alpha *_R k) \cdot (g *_L \beta).$$

Then, 2-categories $\underline{\mathbf{C}}$ also have another underlying category, $\mathbf{C}_h$, with the same objects as $\mathbf{C}$ and 2-cells as arrows. The generalized version of axiom 10 is the so-called *interchange rule*, stating that, whenever both sides are defined, then

$$(\alpha * \beta) \cdot (\gamma * \delta) = (\alpha * \gamma) \cdot (\beta * \delta).$$

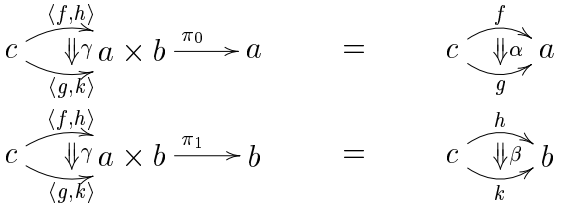
For the sake of simplicity, in the following some of the definitions will be ambiguously given for *enriched* categories: the corresponding definition for sesqui- and 2-categories can be obtained simply putting the correct prefix instead of "enriched".

Definition 3.11 (Enriched Functors and (Natural) Transformations) *Let* $\underline{\mathbf{C}}$, $\underline{\mathbf{D}}$ *be two enriched categories. An* enriched functor $F : C \to D$ *is a triple* $\langle F_O, F_A, F_C \rangle$ *of functions, mapping objects to objects, arrows to arrows and cells to cells, respectively, preserving identities and compositions of all kinds. Let* $F, G : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ *be two parallel enriched functors: an* enriched transformation $\eta : F \Rightarrow G$ *is a function* $A_C \to O_D$ *assigning to each object* $a \in \mathbf{C}$ *an arrow* $\eta_a : F_O(a) \to G_O(a) \in \underline{\mathbf{D}}$*; it is* natural *if moreover for any cell* $\alpha : f \Rightarrow g : a \to b \in \underline{\mathbf{C}}$, $\eta_a *_L F_C(\alpha) = G_C(\alpha) *_L \eta_b$. *If* $\underline{\mathbf{C}}$ *and* $\underline{\mathbf{D}}$ *are two enriched categories, the* enriched functor category $[\underline{\mathbf{C}} \to \underline{\mathbf{D}}]$ *has enriched functors as objects and enriched natural transformations as arrows.* □

Let **2-Cat** (**S-Cat**) be the category of 2-categories (sesqui-categories) and 2-functors (sesqui-functors): there exists an inclusion functor U_i : **2-Cat** $\to$ **S-Cat**, whose left-adjoint F_i simply quotients the structure of a sesqui-category with respect to axiom (10), and such that **2-Cat** is *reflective* inside **S-Cat**. The paradigmatic example of 2-category is $\underline{\mathbf{Cat}}$: objects are small categories, arrows are functors, cells are natural transformations. For sesqui-categories, the paradigmatic example is $\underline{\mathbf{Cat}}_S$: differently from $\underline{\mathbf{Cat}}$, cells are just transformations, i.e., they are not required to satisfy the naturality condition.

To define universal constructions over enriched categories, we need to require suitable properties to hold for cells, instead of just for arrows.

Definition 3.12 (Cartesian Enriched Categories) *Let* $\underline{\mathbf{C}}$ *be an enriched category such that the underlying category* $\mathbf{C}$ *is cartesian. We say that* $\underline{\mathbf{C}}$ *has* enriched products *if for every pair* $\alpha : f \Rightarrow g : c \to a$ *and* $\beta : h \Rightarrow k : c \to b$ *of cells, there exists a unique cell* $\gamma = \langle \alpha, \beta \rangle : \langle f, h \rangle \Rightarrow \langle g, k \rangle : c \to a \times b$ *satisfying* $\gamma *_R \pi_0 = \alpha$ *and* $\gamma *_R \pi_1 = \beta$. *Graphically,*



$\underline{\mathbf{C}}$ *has* terminal enriched object *if for every cell* $\alpha : f \Rightarrow g : c \to a$ *we have* $\alpha *_R !_c = !_a$, *where* 0 *is terminal in* $\mathbf{C}$ *and* $!_a : a \to 0$, $!_c : c \to 0$. ∎

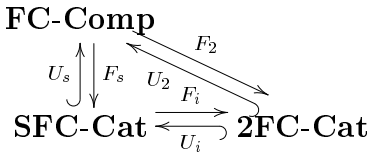
Again, products and terminal objects are defined up-to-isomorphism: we say that, if the underlying cartesian category $\mathbf{D} = \langle \mathbf{C}, \nabla, ! \rangle$ has finite products, the cartesian enriched category $\underline{\mathbf{D}} = \langle \underline{\mathbf{C}}, \nabla, ! \rangle$ has finite enriched products, where the monoidal structure of $\underline{\mathbf{C}}$ has the intuitive definition (and it preserves the one on $\mathbf{C}$), while $\nabla, !$ are natural transformations in **Cat**. An enriched functor $F = \langle F_O, F_A, F_C \rangle : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ is *cartesian* if for all objects a, b the canonical maps $\langle F_A(\pi_0), F_A(\pi_1) \rangle : F_O(a \times b) \to F_O(a) \times F_O(b)$ and $!_{F(0_C)} : F_O(0_C) \to 0_D$ are isomorphisms. A *chosen* functor $F : \mathbf{D} \to \mathbf{E}$ between cartesian enriched categories with (finite) chosen products is a symmetric monoidal enriched functor between the underlying symmetric (strict) monoidal enriched categories, preserving the additional structure: it is also cartesian in the previously defined sense.

Let **2C-Cat** (**SC-Cat**) be the category of cartesian 2-categories (sesqui-categories) and 2-functors (s-functors) with chosen products, and let **2FC-Cat** (**SFC-Cat**) be its counterpart with finite products: the pair of functors $\langle U_i, F_i \rangle$ introduced above still forms an adjunction when restricted to the appropriate sub-categories.

We will be interested in a finitary presentation of an enriched structure, i.e., in a set of generators such that a cat-enriched category can be obtained freely composing cells. An appropriate structure for that is represented by *c-computads*.

Definition 3.13 (C-Computads) *A* c-computad *is a pair* $\langle \mathbf{C}, S \rangle$, *where* $\mathbf{C}$ *is a category and* S *is a set of cells, each of which has a pair of parallel arrows of* $\mathbf{C}$ *as* source *and* target, *respectively. Given the c-computads* $\langle \mathbf{C}, S \rangle$ *and* $\langle \mathbf{C}', S' \rangle$, *a* c-morphism *is a pair* $\langle F, h \rangle$ *such that* $F : \mathbf{C} \to \mathbf{C}'$ *is a functor,* $h : S \to S'$ *is a function, and for every cell* $\alpha : f \Rightarrow g \in S$ *we have* $h(\alpha) : F(f) \Rightarrow F(g) \in S'$. ∎

A c-computad $\langle \mathbf{C}, S \rangle$ is cartesian (with finite products) if so is $\mathbf{C}$, while a c-morphism $\langle F, h \rangle$ preserves products and terminal object (is chosen) if F does so (is chosen). Let **FC-Comp** be the category of cartesian c-computads with finite products and chosen c-morphisms: there exists an obvious forgetful functor $U_2 : \mathbf{2FC\text{-}Cat} \to \mathbf{FC\text{-}Comp}$ which forgets the composition of cells, with left-adjoint F_2. This adjoint composes the cells of a c-computad in all the possible ways, both horizontally and vertically, imposing further equalities in order to satisfy the axioms of a 2-category (and preserving finite products on the underlying category). The main result of [Pow90] assures us that this *pasting* operation is well-defined, i.e., that it is indipendent from the order in which cells are composed. There exists also a similar adjunction pair $\langle U_s, F_s \rangle$ between **SFC-Cat** and **FC-Comp**, such that the following diagram commutes:



3.4 Internal Categories

Basically, an *internal category* of $\mathbf{C}$ (see [BW90]; also denoted a *cat-object* in $\mathbf{C}$) $\mathbf{D}_C$ is just a category such that its classes of arrows and objects are objects of $\mathbf{C}$, while its composition, identity, source and target morphisms are arrows in $\mathbf{C}$. An *internal functor* $F : \mathbf{D}_C \to \mathbf{E}_C$ between categories internal to $\mathbf{C}$ is a pair of arrows $F_A : A_{D_C} \to A_{E_C}, F_O : O_{D_C} \to O_{E_C}$ between, respectively, the objects in $\mathbf{C}$ representing the classes of objects of $\mathbf{D}_C$ and $\mathbf{E}_C$, and the objects in $\mathbf{C}$ representing the classes of arrows of $\mathbf{D}_C$ and $\mathbf{E}_C$. As an example, a category internal to **Set**, the category of small sets and functions, is just a locally small category $\mathbf{C}$. An *internal transformation* $\eta : F \Rightarrow G : \mathbf{D}_C \to \mathbf{E}_C$ is an arrow $O_{D_C} \to A_{E_C}$, satisfying the requirement $\eta ;_C \delta_0 = F_O$ and $\eta ;_C \delta_1 = G_O$, where δ_0, δ_1 are arrows in $\mathbf{C}$ representing the source and target functions in $\mathbf{E}_C$, while $;_C$ is the composition function in $\mathbf{C}$.

Actually, the definition is slightly more involved: it is necessary, in order to correctly define categories internal to $\mathbf{C}$, that $\mathbf{C}$ has pullbacks, so that for the category $\mathbf{D}_C$ the composition operator $;_{D_C}$ is defined as an arrow in $\mathbf{C}$ from the pullback $A_{D_C} \times_0 A_{D_C}$ (along $\langle (\delta_1)_{D_C}, (\delta_0)_{D_C} \rangle$) to A_{D_C}. Then we can define the *vertical composition* of two internal transformations as $\eta : F \Rightarrow G, \gamma : G \Rightarrow H$ as the arrow $[\eta, \gamma];_C (;_{E_C})$, or informally as $\eta ;_{E_C} \gamma$. We have *naturality* as the axiom $[F_A, (\delta_1)_{E_C} ;_C \eta];_C (;_{E_C}) =$

$[(\delta_0)_{E_C};_C \eta, G_A];_C (;_{E_C})$; and so on. An interesting example is already given by a simple category like **Cpo**, the category of small CPO's and continuous functions. For the sake of readability, we will refer to categories, functors and (natural) transformations internal to **Cpo** simply prefixing the adjective *continuous*.

Definition 3.14 (Continuous Categories) *A* continuous category $\mathbf{C}$ *is a tuple of the kind* $(O_C, A_C, id_C, \delta_0, \delta_1, ;_C)$*, such that:*

- O_C*,* A_C *are small* CPO*'s;*
- $id_C : O_C \to A_C$ *and* $\delta_0, \delta_1 : A_C \to O_C$ *are continuous functions;*
- $;_C : A_C \times_0 A_C \to A_C$ *is a continuous function, where* $A_C \times_0 A_C$ *is the pullback along* $\langle \delta_1, \delta_0 \rangle$*.*

Moreover, these functions satisfy the conditions holding for the corresponding structures in an ordinary category. A continuous functor is just a pair of continuous functions preserving identities and composition. A continuous category $\mathbf{C}$ *is* strict *if all its components are strict.* □

We can analogously extend the definitions of functor and natural transformation. Also the *horizontal* and *vertical* composition of continuous natural transformations are the intuitive extensions of those defined on **Cat**, and allows us to define the internalization of $\underline{\mathbf{Cat}}$ inside **Cpo**. We indicate with $\underline{\mathbf{Cat}(\mathbf{Cpo})}$ the 2-category such that its objects are continuous categories, its arrows are continuous functors, and its 2-cells are continuous natural transformations. An interesting sub-2-category is obtained simply restricting the class of objects to strictly continuous category: the resulting 2-category is denoted $\underline{\mathbf{Cat}(\mathbf{Cpo})}_S$.

3.4.1 Double-categories

A 2-category $\underline{\mathbf{C}}$ can be described as a internal object to **Cat**, such that the categories $\mathbf{O}_C$ and $\mathbf{A}_C$ have the same set of objects. In general, an internal object to **Cat** is a *double-category* (see [BE74]; also [DP93] for some recent results on pasting). It represents an intuitive generalization of a 2-category, and it admits the following *naïve* presentation, adapted from [KS74].

Definition 3.15 (Double-Categories) *A double-category* $\underline{\mathbf{C}}$ *(shortly, d-category) consists of a collection* $\{a, b, c, \ldots\}$ *of* objects*, or 0-cells, a collection* $\{f, g, h, \ldots\}$ *of* horizontal arrows*, or horizontal 1-cells, a collection* $\{x, y, z, \ldots\}$ *of* vertical arrows*, or vertical 1-cells, and a collection* $\{\alpha, \beta, \gamma, \ldots\}$ *of* double-cells*, also d-cells. Objects and horizontal arrows form a category, the* horizontal 1-category $\mathbf{C}$*, with identity* id_a *for each object* a*; also objects and vertical arrows form a category, the* vertical 1-category*, with identity* id^a *for each object* a*. D-cells are assigned* horizontal source *and* target *(which are vertical 1-cells), written as* $\alpha : x \Rightarrow_h y$*, and* vertical source *and* target *(which are horizontal 1-cells), written as* $\alpha : f \Rightarrow_v g$*; furthermore, these 1-cells must be* compatible*, i.e., they must satisfy particular requirements on their source and target: in graphical terms*

$$\begin{array}{ccc} a & \xrightarrow{f} & b \\ {\scriptstyle x}\downarrow & \alpha & \downarrow{\scriptstyle y} \\ c & \xrightarrow[g]{} & d \end{array}$$

In addition, d-cells can be composed both vertically $(\alpha *_v \beta)$ *and horizontally* $(\gamma *_h \delta)$*: given* $\alpha : x \Rightarrow_h y$ *and* $\beta : y \Rightarrow_h z$*, then*

$$\begin{array}{ccccc} a & \xrightarrow{f} & b & \xrightarrow{h} & e \\ {\scriptstyle x}\downarrow & \alpha & \downarrow{\scriptstyle y} & \beta & \downarrow{\scriptstyle z} \\ c & \xrightarrow[g]{} & d & \xrightarrow[i]{} & m \end{array} \quad = \quad \begin{array}{ccc} a & \xrightarrow{f;h} & e \\ {\scriptstyle x}\downarrow & \alpha *_h \beta & \downarrow{\scriptstyle z} \\ c & \xrightarrow[g;i]{} & m \end{array}$$

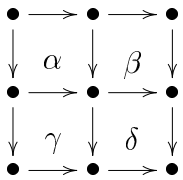
$$\begin{array}{ccc} a & \xrightarrow{f} & b \\ {\scriptstyle x;w}\downarrow & \gamma *_v \delta & \downarrow{\scriptstyle y;z} \\ e & \xrightarrow[h]{} & m \end{array} \quad = \quad \begin{array}{ccc} a & \xrightarrow{f} & b \\ {\scriptstyle x}\downarrow & \gamma & \downarrow{\scriptstyle y} \\ c & \xrightarrow{g} & d \\ {\scriptstyle w}\downarrow & \delta & \downarrow{\scriptstyle z} \\ e & \xrightarrow[h]{} & m \end{array}$$

Under each of these laws d-cells form a category, the horizontal category $\mathbf{C}_h$ *and the* vertical category *respectively, with identities*

$$\begin{array}{ccc} a & \xrightarrow{id_a} & a \\ {\scriptstyle x}\downarrow & 1_x & \downarrow{\scriptstyle x} \\ b & \xrightarrow[id_b]{} & b \end{array} \qquad \begin{array}{ccc} a & \xrightarrow{f} & b \\ {\scriptstyle id^a}\downarrow & 1^f & \downarrow{\scriptstyle id^b} \\ a & \xrightarrow[f]{} & b \end{array}$$

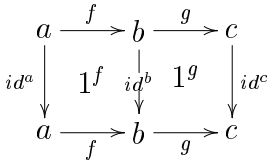
Moreover, the following equations must hold:

1. *whenever the composite*



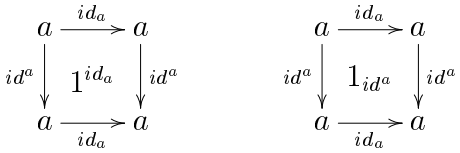
is well-defined, then $(\alpha *_v \gamma) *_h (\beta *_v \delta) = (\alpha *_h \beta) *_v (\gamma *_h \delta)$*;*

2. *the composite*



has to be $1^{f;g}$*, and similarly for vertical composition of horizontal identities;*

3. *finally, the horizontal and vertical identities*



must coincide.

Given $\underline{\mathbf{C}}, \underline{\mathbf{D}}$ *d-categories, a* d-functor $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ *is a 4-tuple of functions mapping objects to objects, horizontal (vertical) arrows to horizontal (vertical) arrows and d-cells to d-cells, preserving identities and compositions of all kinds.* ∎

We denote by **D-Cat** the category of d-categories and d-functors. It includes **2-Cat** as a subcategory, since each 2-category is a double-category such that the vertical 1-category is a discrete one, i.e., all its arrows are identities.

Since a d-category is a cat-object in **Cat**, a d-functor $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ can be equivalently defined as a triple $\langle F, F_h, F_c \rangle$ of functors, such that $F : \mathbf{C} \to \mathbf{D}$ is a functor between the horizontal 1-categories, $F_h : \mathbf{C}_h \to \mathbf{D}_h$ is a functor between the horizontal categories and $F_C : \mathbf{C}_h \times_0 \mathbf{C}_h \to \mathbf{D}_h \times_0 \mathbf{D}_h$ between their pullbacks, preserving vertical compositions and identities of all kind. Let $F, G : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ be d-functors: a *d-transformation* $\eta : F \Rightarrow G$ is a functor $\eta : \mathbf{C} \to \mathbf{D}_h$ from the horizontal 1-category $\mathbf{C}$ to the horizontal category $\mathbf{D}_h$, satisfying $\eta; \delta_0 = F$ and $\eta; \delta_1 = G$, where ; is composition in **Cat**, while δ_0, δ_1 are

functors from $\mathbf{D}_h$ to $\mathbf{D}$. A d-transformation is *natural* if it satisfies the intuitive natural requirement. We say that a d-category is cartesian with chosen (finite) products if it is a cat-object in **CC-Cat** (**FC-Cat**) — the category of small categories with chosen (finite) products and chosen functors — and the associated functors representing, source, target, composition and identity strictly preserve products (we are then considering a suitable subcategory of **Cat(CC-Cat)** (**Cat(FC-Cat)**), analogously to what already done about strict continuous categories).

We give now an equivalent, yet more explicit definition of horizontal product. Let $\underline{\mathbf{C}}$ be a d-category: we say that $\underline{\mathbf{C}}$ is cartesian (with *horizontal products*) if $\mathbf{C}_h$ and $\mathbf{C}$ have (finite) chosen products, and they preserve composition in the vertical (1-)category. The preservation requirement is equivalent to imposing the functoriality of horizontal product with respect to vertical composition: that is, whenever the composite is well defined, then $\langle \alpha *_v \beta, \gamma *_v \delta \rangle = \langle \alpha, \gamma \rangle *_v \langle \beta, \delta \rangle$. For instance, let a, b and c, d be 0-cells such that their products in the horizontal 1-category are the triples $\langle a \times b, p_0, p_1 \rangle$ and $\langle c \times d, p'_0, p'_1 \rangle$: we say that two objects $x : a \to c$ and $y : b \to d$ in $\mathbf{C}_h$ have product $x \times y : \langle a \times b \to c \times d, \pi_0, \pi_1 \rangle$, where

$$\begin{array}{ccc} a \times b & \xrightarrow{p_0} & a \\ {\scriptstyle x\times y}\downarrow & \pi_0 & \downarrow{\scriptstyle x} \\ c \times d & \xrightarrow[p'_0]{} & c \end{array} \qquad \begin{array}{ccc} a \times b & \xrightarrow{p_1} & b \\ {\scriptstyle x\times y}\downarrow & \pi_1 & \downarrow{\scriptstyle y} \\ c \times d & \xrightarrow[p'_1]{} & d \end{array}$$

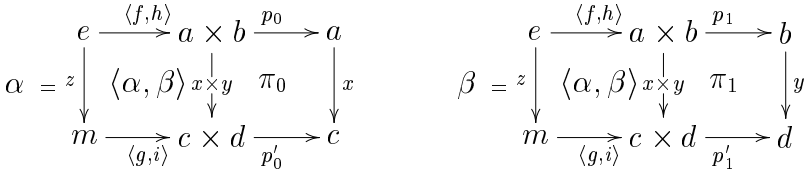
if, given any two double cells

$$\begin{array}{ccc} e & \xrightarrow{f} & a \\ {\scriptstyle z}\downarrow & \alpha & \downarrow{\scriptstyle x} \\ m & \xrightarrow[g]{} & c \end{array} \qquad \begin{array}{ccc} e & \xrightarrow{h} & b \\ {\scriptstyle z}\downarrow & \beta & \downarrow{\scriptstyle y} \\ m & \xrightarrow[i]{} & d \end{array}$$

then there exists a unique double-cell

$$\begin{array}{ccc} e & \xrightarrow{\langle f,h\rangle} & a \times b \\ {\scriptstyle z}\downarrow & \langle \alpha, \beta \rangle & \downarrow{\scriptstyle x\times y} \\ m & \xrightarrow[\langle g,i\rangle]{} & c \times d \end{array}$$

such that $\alpha = \langle \alpha, \beta \rangle *_h \pi_0$, $\beta = \langle \alpha, \beta \rangle *_h \pi_1$ or, equivalently, such that the following identities hold:



In the following, we will denote with **DC-Cat** (**DFC-Cat**) the category of d-categories with chosen (finite) horizontal products, and chosen d-functors.

Definition 3.16 (Double-Computads) *A* d-computad *is a triple* $\langle \mathbf{C}, \mathbf{C}_h, \tau \rangle$ *such that* $\mathbf{C}$, $\mathbf{C}_h$ *are categories, and* $\tau = \langle \tau_0, \tau_1, id_\tau \rangle$ *is a triple of functors* $\tau_i : \mathbf{C}_h \to \mathbf{C}, id_\tau : \mathbf{C} \to \mathbf{C}_h$ *such that* $id_\tau ; \tau_0 = id_\tau ; \tau_1 = id_C$. *A* d-morphism *is a pair* $\langle F, F_h \rangle$ *of functors* $F : \mathbf{C} \to \mathbf{D}$, $F_h : \mathbf{C}_h \to \mathbf{D}_h$ *between the underlying categories, such that* τ, τ_i *are preserved.* ∎

In other words, a *d-computad* is an reflexive graph internal to **Cat**. It can also be described in a slightly more restrictive way (but for our purposes there will be no loss of generality) as a triple $\langle \mathbf{C}, D, S \rangle$, where $\mathbf{C}$ is a category, D is a reflexive graph (and its set of nodes is the same as the set of objects of $\mathbf{C}$) and S is a set of d-cells, each of which has assigned two pairs of compatible arrows in $\mathbf{C}$ and D as *horizontal* and *vertical source* and *target*, respectively. Given the d-computads $\langle \mathbf{C}, D, S \rangle$ and $\langle \mathbf{C}', D', S' \rangle$, a *d-morphism* is a triple $\langle F, G, h \rangle$ such that $F : \mathbf{C} \to \mathbf{C}'$ is a functor, $G : D \to D'$ is a graph morphism, and $h : S \to S'$ is a function (such that if $\alpha : f \to g \in A_C$ and $\beta : x \to y \in A_D$, then $h(\alpha) : F(f) \to F(g) \in A'_C$ and $h(\beta) : G(x) \to G(y) \in A'_D$), preserving identities and compositions of all kind.

Let **D-Comp** be the category of d-computads and d-morphisms: there exists an obvious forgetful functor U_d : **D-Cat** $\to$ **D-Comp**, with a left-adjoint F_d. This adjoint composes the cells of a d-computad in all the possible ways, both horizontally and vertically, imposing further equalities in order to satisfy the axioms of a d-category. We indicate with **DFC-Comp** the category of d-computads $\langle \mathbf{C}, D, S \rangle$ such that $\mathbf{C}$ has finite products and D has pairing: also in this case there exists a forgetful functor U_{df} : **DFC-Cat** $\to$ **DFC-Comp**, with a left-adjoint F_{df}.

3.5 Algebraic Theories

The categories of algebras we introduced can be presented in an equivalent way by using simple categorical techniques. Although such definitions are slightly more involved than the classical, set-theoretical ones, they have the advantage of separating in a better

way the "Σ-structure" from the additional algebraic structure that the carrier can enjoy. An *algebraic theories* [Law63, Law68, KR77] is just a cartesian category having natural numbers as objects. Given a signature Σ, the associated algebraic theory $\mathbf{Th}(\Sigma)$ (usually denoted in the following as the *Lawvere theory* associated to Σ) can be also described by means of a suitable free construction.

Definition 3.17 (Lawvere Theories) *Given a signature* Σ*, the associated* Lawvere theory *is the cartesian category* $\mathbf{Th}(\Sigma)$ *with finite products, freely generated from the graph with pairing* G_Σ *such that*

- *its objects are underlined natural numbers:* $\underline{0}$ *is the identity element and pairing is defined as* $\underline{n} \otimes \underline{m} = \underline{n+m}$*;*
- *for every operator* $f \in \Sigma_n$*, there is a basic arrow* $f_\Sigma : \underline{n} \to \underline{1}$*.* ∎

The relevant property of $\mathbf{Th}(\Sigma)$ is that arrows from m to n are in one-to-one correspondence with n-tuple of terms of the free Σ-algebra with at most m variables. Each arrow t_Σ: $\underline{n} \to \underline{1}$ identifies a Σ-term t with variables among x_1,...,x_n; an arrow $\underline{n} \to \underline{m}$ is a m-tuple of Σ-terms with n variables, and arrow composition is term substitution. The Lawvere theory can be regarded as an alternative presentation of a signature. Indeed, the additional structure it contains (besides the operators of the signature) is generated in a completely free way, so, in a sense, it does not add "information" to the original signature. The advantage of this presentation is that, using categorical techniques, we can easily define a very general notion of *model* for a Lawvere theory, which subsumes both algebras and continuous algebras.

Definition 3.18 (Models of Lawvere Theories) *Let* $\mathbf{C}$ *be a cartesian category with chosen products. A* $\mathbf{C}$-model *of the Lawvere theory associated to a signature* Σ *is a chosen functor* $\mathcal{M} : \mathbf{Th}(\Sigma) \to \mathbf{C}$*, while a* model morphism *is a natural transformation between models. The category* $\mathbf{C\text{-}Mod}_\Sigma$ *of* $\mathbf{C}$*-models is the functor category* $[\mathbf{Th}(\Sigma) \to \mathbf{C}]$*: its objects are* $\mathbf{C}$*-models of* $\mathbf{Th}(\Sigma)$*, while its arrows are model morphisms.* ∎

By replacing the generic cartesian category $\mathbf{C}$ with specific categories, we obtain models which are equivalent (in a strong sense) to the various kinds of Σ-algebras introduced above. In particular, the $\mathbf{Set}$-models of $\mathbf{Th}(\boldsymbol{\Sigma})$ turn out to be essentially the Σ-algebras, while its $\mathbf{Cpo}$-models are nothing else than the complete ordered Σ-algebras.

Proposition 3.3 (Categories of Algebras as Functor Categories) *For any signature Σ, the categorical equivalences Σ-**Alg** $\equiv$ **Set-Mod**$_\Sigma$ and Σ-**COAlg** $\equiv$ **Cpo-Mod**$_\Sigma$ hold. Moreover, let* **S-Cpo-Mod**$_\Sigma$ *be the sub-category of* **Cpo-Mod**$_\Sigma$ *such that the target of the functors are strict continuous categories, and all the components of the natural transformations are strict: then Σ-**SCAlg** $\equiv$ **S-CPO-Mod**$_\Sigma$.* □

The result for **Set**-models is well-known: here we only sketch the underlying ideas. Given a **Set**-model $\mathcal{M}$: **Th**$(\Sigma) \to$ **Set**, consider the pair $A_\mathcal{M} = \langle \mathcal{M}(\underline{1}), \rho = \{\mathcal{M}(f_\Sigma) \mid f \in \Sigma\}\rangle$. It is easy to check that $A_\mathcal{M}$ is a Σ-algebra: indeed, $\mathcal{M}(\underline{1})$ is a set, and for any $f \in \Sigma_n$, since $f_\Sigma : \underline{n} \to \underline{1}$ in **Th**(Σ), we have $f_{A_\mathcal{M}} \stackrel{def}{=} \mathcal{M}(f_\Sigma) : \mathcal{M}(\underline{n}) \to \mathcal{M}(\underline{1})$, and thus $\mathcal{M}(f_\Sigma) : \mathcal{M}(\underline{1})^n \to \mathcal{M}(\underline{1})$ (because $\mathcal{M}$ is product preserving and $\underline{n} = \underline{1}^n$ in **Th**(Σ)), showing that $f_{A_\mathcal{M}}$ has the correct type. Besides, each natural transformation $\alpha : \mathcal{M} \Rightarrow \mathcal{N}$ between **Set**-models characterizes a homomorphism between $A_\mathcal{M}$ and $A_\mathcal{N}$: in fact, the naturality requirement implies that for any operator $f \in \Sigma_n$, we have $\alpha_{\underline{1}} \circ \mathcal{M}(f) = \mathcal{N}(f) \circ \alpha_n$, i.e., $\alpha_{\underline{1}} \circ f_{A_\mathcal{M}} = f_{A_\mathcal{N}} \circ \alpha_{\underline{1}}^n$.

These ideas apply also to (strict) **Cpo**-models: here $\mathcal{M}$ is by definition a (strict) CPO, all operators of the signature are mapped by $\mathcal{M}$ to continuous functions of the right type, while a model is a natural transformation $\eta : \mathcal{M} \Rightarrow \mathcal{N}$, associating to 1 a (strict) continuous homomorphism $\eta_1 = \mathcal{M}(1) \to \mathcal{N}(1)$.

All the previous results can be easily lifted when we deal with an equational theory (Σ, E): for each axiom $s = t$, let $X = var(s) \cup var(t)$ be the set of (distinguished) variables in s and t, with $\#|X| = n$. Due to the correspondence between terms and arrows, we consider $t_\Sigma, s_\Sigma \in$ **Th**$(\Sigma)[n, 1]$, and we quotient **Th**(Σ) with the axiom $t_\Sigma = s_\Sigma$. The resulting algebraic theory is denoted **Th**(Σ, E). Also in this case we can get a suitable functor category [**Th**$((\Sigma, E)) \to$ **Set**]. An equivalent category could be obtained also "internalizing" the axioms, that's to say, restricting our attention to the full sub-category of [**Th**$(\Sigma) \to$ **Set**], whose functors F preserve the axioms, i.e, such that $F(t_\Sigma) = F(s_\Sigma)$.

Chapter 4

Rewriting Logic: Syntax and Semantics

In this chapter we introduce the basic definitions of (unconditional) *Rewriting Logic*. As devised in the introduction, the basic idea is to provide a logic that is able to reason about the *changes* of a computational system in an intrinsically *concurrent* way. A *rewriting theory* is roughly described by an equational theory, and a set of (labeled) rewriting rules over terms of the theory. Each rule can be considered as a general pattern for a basic action, while an actual rewrite is described by a sequent: a tuple $\langle \alpha, t, s \rangle$, stating that t *rewrites to* s *via* α, where α is a suitable encoding of the causes of the rewrite. A sequent is obtained by finitely many applications of a set of *rules of deduction*: a rewriting logic is actually given by a rewriting theory, and a set of deduction rules; obviously, *different* sets may entail different families of sequents. In the first section of the chapter we present the *syntax* of rewriting logic, we formally define rewriting theories, and introduce two sets of deduction rules. (An equivalent version of) the first one was originally proposed in [Mes92], while the second was introduced in [CGM95]. In the second section we provide two algebraic axiomatizations over sequents; this way we provide a somewhat more abstract description of the computations performed by a system, equating families of sequents that are *computationally* equivalent. In the third and fourth sections we aim instead at describing the *model-theoretic* side of the Lambek-Lawvere analogy we devised in the introduction. We introduce two different kinds of categorical semantics for rewriting theories, and we prove their "equivalence" with the axiomatic one.

4.1 Rewriting Theories

We open this section introducing *rewriting logic*. For a summary of the basic notions about universal algebras, we refer to Chapter 2.

Definition 4.1 (Rewriting Theories) *Let X be a set of variables. A* rewriting theory *$\mathcal{R}$ (over X) is a tuple $\langle(\Sigma, E), L, R\rangle$, where (Σ, E) is an equational theory, L is a set of labels, and R is a function $R : L \to T_\Sigma(X) \times T_\Sigma(X)$, such that for all $d \in L$, if $R(d) = \langle l, r\rangle$ then $var(r) \subseteq var(l) \subseteq X$ and l is not a variable.* ∎

Given a rewriting theory $\mathcal{R}$, we write $d : l \to r \in R$ if $d \in L$ and $R(d) = \langle l, r\rangle$; sometimes, to make explicit the variables contained in a rule, we will write $d(x_1, \ldots, x_n) : l(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n) \in R$ where $\{x_1, \ldots, x_n\} = var(l)$; given a substitution $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, we will write $l(t_1, \ldots, t_n)$ for $l\sigma$.

Actually, in this thesis we are going to deal mainly with *term rewriting systems* (TRS's): i.e., rewriting theories such that the associated equational theory has an empty set of axioms. A TRS is simply indicated by $\langle\Sigma, L, R\rangle$, and (usually) is uniquely determinated by its set of rules.

In classical term rewriting, a rule $d : l \to r$ can be applied to a term t if there is a subterm $\langle w, s\rangle$ of t such that l matches s, and the result is the term t where the matched subterm is replaced by a suitable instantiation of r. Moreover, a term can be rewritten into another if there exists an appropriate chain of rule applications. This presentation makes *sequences* of rewrites the basic notion: it does not allow to reason about *how* a rewrite can be executed, i.e., to record the possible, different justifications of a derivation. Instead, in rewriting logic the idea is to take a logical viewpoint, regarding a rewriting theory $\mathcal{R}$ as a logical theory, and any rewriting — making use of rules in $\mathcal{R}$ — as a sequent entailed by the theory. The entailment relation is defined inductively by a set of *deduction rules*. With respect to the set-theoretical viewpoint, this choice allows us to equip each rewriting step with a reasonable encoding of its causes.

Definition 4.2 (Rewriting Sequents) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. Let $\Lambda = \cup_n \Lambda_n$ be the signature containing all the rules $d : l \to r \in R$ with the corresponding arity given by the number of variables in d: more precisely, for each n, $\Lambda_n = \{d \mid d(x_1, \ldots, x_n) : l(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n) \in R\}$. A* proof term *$\alpha$ is a term of the algebra $T_\mathcal{R} = T_{\Sigma \cup \Lambda \cup \{\cdot\}}$, where "$\cdot$" is a binary operator (we assume that there are no clashes of names between the various sets of operators). A* (rewriting) sequent *is a triple $\langle\alpha, t, s\rangle$ (usually written as $\alpha : t \to s$) where α is a proof term and $t, s \in T_\Sigma$.* ∎

So, sequents in rewriting logic have the form $\alpha : t \to s$, where t and s are terms of the algebra $T_{(\Sigma,E)}$ and α is a proof term, encoding a justification of the rewriting of t into s. Note that, for the sake of simplicity, we are restricting ourselves to *ground* proof terms, i.e., to deal only with rewrites over T_Σ: this does not limit the generality, since each variable can be considered as a new constant, as already remarked in Chapter 2. We say that t *rewrites to* s *via* α if the sequent $\alpha : t \to s$ can be obtained by finitely many applications of certain *rules of deduction.*

Definition 4.3 (Rewriting Logic) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. We say that $\mathcal{R}$* entails *the* full sequent $\alpha : s \to t$ *if it can be obtained by a finite number of applications of the following rules of deduction:*

- (Full Instantiation)
$$\frac{d : l \to r \in R, d \in \Lambda_n, \alpha_i : t_i \to s_i \; for \; i = 1, \ldots, n}{d(\alpha_1, \ldots, \alpha_n) : l(t_1, \ldots, t_n) \to r(s_1, \ldots, s_n)};$$
- (Congruence)
$$\frac{f \in \Sigma_n, \alpha_i : t_i \to s_i \; for \; i = 1, \ldots, n}{f(\alpha_1, \ldots, \alpha_n) : f(t_1, \ldots, t_n) \to f(s_1, \ldots, s_n)};$$
- (Transitivity)
$$\frac{\alpha : s \to t, \;\; \beta : t \to u}{\alpha \cdot \beta : s \to u}.$$

□

Let $\mathcal{R}$ be a rewrite theory: the class of full sequents entailed by $\mathcal{R}$ induces a set-theoretical rewrite relation over terms, simply obtained by dropping the proof term of a sequent. We indicate such a relation with $\mathcal{B}_\mathcal{R}$.

The deduction system we introduced is equivalent to the one defined in [Mes92]. Of course, this is only one of the possible, equivalent ways to define the class of sequents that are actually entailed by the system, and *a fortiori*, to obtain the relation $\mathcal{B}_\mathcal{R}$. It has, however, the advantage of being rather intuitive. *Transitivity* states that the rewrite relation entailed by the system is (not-so-surprisingly) transitive: two suitable rewrites can be composed, and the resulting proof term is given by the composition of the two components. *Congruence* states that the rewrite relation is also *compatible* with respect to the algebraic structure, since it is closed under contexts; moreover, there are sequents entailed by the theory describing the parallel execution of *disjoint* rewrites: the associated

proof term provides the context for the respective justifications. This rule also says that the rewrite relation is *reflexive*: each term t can be rewritten to itself, if all its subcomponents are idle, i.e., the sequent $\langle t, t, t\rangle$ can be obtained by an inductive application (on the structure of t) of the congruence rule. Maybe, the most interesting rule is *full instantiation*: first, it implies that the transition relation is *stable* under substitution, that is, it is closed under substitutions. But the associated sequent describes also the *simultaneous* execution of *nested* rewrites: two subterms matching the left-hand sides of two rules can be rewritten in parallel even if their roots are not disjoint, i.e., if one is above the other, provided that they do not overlap.

Full Instantiation is not the only possible choice to get a stable and compatible rewrite relation: the property is shared by *flat sequents*, originally introduced in [CGM95] as the algebraic counterpart of a (categorical) model of term rewriting proposed in [Ste94].

Definition 4.4 (Flat Rewriting Logic) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. We say that $\mathcal{R}$* entails *the* flat sequent *$\alpha : s \to t$ if it can be obtained by a finite number of applications of the following rules of deduction:*

(Flat Instantiation)

$$\frac{d : l \to r \in R, d \in \Lambda_n,\ t_i \in T_\Sigma \text{ for } i = 1, \ldots, n}{d(t_1, \ldots, t_n) : l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)}.$$

(Congruence) *As in Definition 4.3;*

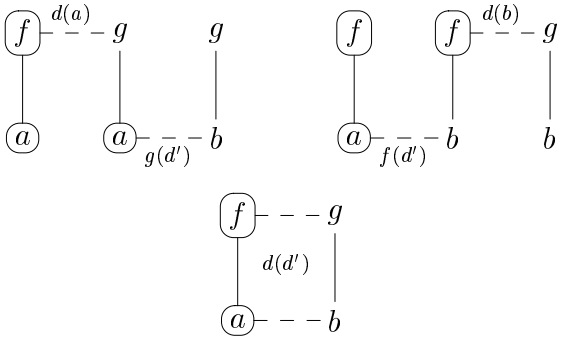
(Transitivity) *As in Definition 4.3.* □

Flat instantiation still induces a compatible rewrite relation, even if it does not offer suitable sequents for describing the nesting of rewrites. A rule can only be instantiated with elements of T_Σ, and thus the rule names in a proof term generated using this rule instead of *full instantiation* will appear at mutually disjoint positions. Nevertheless, the *flat* rewrite relation is equivalent to $\mathcal{B}_\mathcal{R}$, since each nested rewrite can be described as a suitable sequence of flat ones. Then, the basic difference between the two approaches relies on the assumptions we have on a possible *implementation schema*[1].

As for now, let us put to work the two different definitions of instantiation, and consider the TRS $\mathcal{V} = \{d(x) : f(x) \to g(x), d' : a \to b\}$. It entails the flat sequents $d(a) \cdot g(d')$ and

[1]For other suitable instantiations of the logic, as well as a careful mapping into classic term rewriting, the reader has to wait for the next chapters.

$f(d') \cdot d(b)$ both with source $f(a)$ and target $g(b)$: rule d has been instantiated with $\{x/a\}$ and $\{x/b\}$ respectively, while d' has been contextualized; it also entails the full sequent $d(d') : f(a) \to g(b)$, where d' is *nested inside* d. Graphically, we have the rewrites



using the standard (yet suggestive, in our case) representation of terms as trees. We already said that, when considering just the rewrite relation, the two systems are equivalent: it is easy to show (and the proposition will be made more precise in the next section) that a TRS $\mathcal{R}$ entails a full sequent $\alpha : t \to s$, iff there exists a finite chain $\alpha_i : t_i \to s_i, i = 1, \ldots, n$ of flat sequents, such that $\alpha_1 \cdot \ldots \cdot \alpha_n : t \to s$. So, why to distinguish between flat and full rewrites? First of all, we must remind the reader that sequents represent an operational model for rewriting theories: each sequent can be considered as the encoding of a "concrete" computation of the machine. If we consider a term as a totally distributed structure, i.e., such that an actual rewriting machine can act separately on each occurrence of the term, then full sequents are able to describe the simultaneous execution of nested rewrites. Instead, a flat sequent can express only "disjoint" concurrency: two rewrites can be executed simultaneously only if they act on disjoint positions of a term. This is what we were saying stating that choosing a set of deduction rules is implicitly the same as choosing a particular implementation schema, so to say, for a rewriting theory.

4.2 Algebraic Semantics

In our view, the slogan of rewriting logic should be "an algebraic structure and a suitable axiomatization capture the concurrency of a system". In Chapter 6 we will elaborate on this statement about concurrency, trying to explain why (and when) equipping the set of derivations of a rewriting theory with a suitable equivalence relation means to provide a concurrent semantics for the system the theory aims at describing. As for now, we recall the reader that choosing a set of deduction rules means to specify a given implementa-

tion schema for the reduction mechanism. From this point of view, an equivalence over derivations can be considered just as a way to abstract away from implementation details, equating derivations that are *computationally* equivalent. Describing an equivalence over derivations is a particularly easy task in the setting of rewriting logic, where the elements of the space of computations are encoded by terms of the algebra $T_{\mathcal{R}}$: a suitable equivalence can be easily expressed as an appropriate set of axioms on proof terms.

Definition 4.5 (Abstract Flat Sequents) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. An* abstract flat sequent *entailed by $\mathcal{R}$ is an equivalence class of flat sequents entailed by $\mathcal{R}$ modulo the following set E_1 of axioms, which are intended to apply to the corresponding proof terms:*

- (Associativity)
$$\frac{\alpha, \beta, \gamma \in T_{\mathcal{R}}}{\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma};$$
- (Axiomatizing)
$$\frac{t(x_1, \ldots, x_n) = s(x_1, \ldots, x_n) \in E, \alpha_i \in T_{\mathcal{R}} \text{ for } i = 1, \ldots, n}{t(\alpha_1, \ldots, \alpha_n) = s(\alpha_1, \ldots, \alpha_n)};$$
- (Distributivity)
$$\frac{f \in \Sigma_n, \alpha_i, \beta_i \in T_{\mathcal{R}} \text{ for } i = 1, \ldots, n}{f(\alpha_1 \cdot \beta_1, \ldots, \alpha_n \cdot \beta_n) = f(\alpha_1, \ldots, \alpha_n) \cdot f(\beta_1, \ldots, \beta_n)};$$
- (Identity)
$$\frac{\alpha : s \to t}{s \cdot \alpha = \alpha = \alpha \cdot t}.$$

□

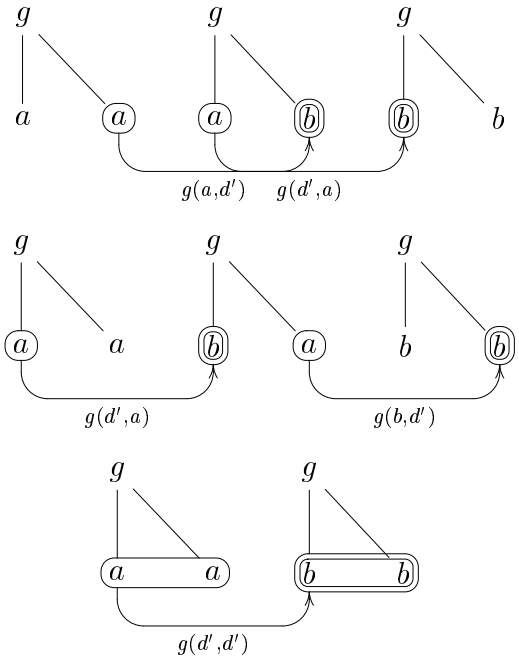
Note that we are rather informal here, since we are not in the classical framework of algebraic varieties: we implicitly assume to apply the axioms only to *well-formed* proof terms, i.e., those that are actually entailed by the deduction rules. The problem can be overcome simply noting that the class of entailed flat sequents forms a *partial algebra*: its total operators are induced by those in Σ (e.g., its constants are the triples $\langle a, a, a\rangle$ for a constant in Σ, etc.); while rules and composition induce partial operators defined only over a subset of tuples of elements, determined by suitable equations (e.g., a rule $d(x_1 \ldots x_n)$ induces an operator defined on the tuples $\langle t, t, t\rangle$ for $t \in T_\Sigma$). All this amounts to say that the class of entailed flat sequents can be defined by means of an *essentially*

algebraic structure, and can be formally described by means of *sketches* [BW90]. A careful treatment of partial algebras in the framework of algebraic semantics can be found in [Rei87], while for some basic results we refer the reader to [Gra79]. The following fact is actually sufficient for our purposes.

Fact 4.1 *Let us consider two flat entailed sequents $\alpha : t \to s$, $\beta : u \to v$, and let $\cong_{E_1}$ be the minimal congruence obtained closing the axioms in E_1 with respect to substitution. The flat proof terms α, β are equated if there exists a sequence $\alpha_i : t_i \to s_i$ for $i = 1 \ldots n$, of flat entailed sequents such that $\alpha_i \cong_{E_1} \alpha_{i+1}$, $\alpha_1 = \alpha$, $\alpha_n = \beta$. Moreover, flat entailed proof terms equated by the axioms in E_1 have the same source and target terms: if $\alpha : s \to t$ and $\beta : u \to v$ are flat sequents entailed by $\mathcal{R}$, and $\alpha \cong_{E_1} \beta$, then $s \cong_E u$ and $t \cong_E v$. Thus, an abstract flat sequent can be safely represented as a triple $\alpha : s \to t$.* ∎

The axioms have an intuitive meaning. *Associativity* and *Identity* need no explanation. Also *Distributivity* has an obvious meaning: to give a context to the composition of two rewrites is the same as to compose the contextualization of the single rewrites. *Axiomatizing* lifts on proof terms the axioms verified by the underlying algebra of terms.

Let us consider for example the TRS $\mathcal{W} = \{d(x) : f(x) \to g(x,x), d' : a \to b, d''(x) : h(x) \to c\}$. The distributivity axiom identifies the proof terms $g(d', d')$, $g(d', a) \cdot g(b, d')$ and $g(a, d') \cdot g(d', b)$. Graphically, the following rewrites are equated.



The equivalence induced by E_1 equates proof terms representing derivations differing only in the order in which disjoint rewrites are performed. This is not the case for derivations differing in the order in which nested rewrites are executed. Let us take the TRS $\mathcal{V}$ described in the previous section: the proof terms $f(d') \cdot d(b)$ and $d(a) \cdot g(d')$ are not equated by E_1. A different axiomatization, taking care of such identifications, is the following, originally proposed in [Mes92].

Definition 4.6 (Abstract Full Sequents) *Let $\mathcal{R} = \langle (\Sigma, E), L, R \rangle$ be a rewriting theory. An* abstract full sequent *entailed by $\mathcal{R}$ is an equivalence class of full sequents entailed by $\mathcal{R}$ modulo the set E_2 of axioms, which are intended to apply to the corresponding proof terms; E_2 is the union of the axioms in E_1 with the following one:*

- (Interchange)

$$\frac{d : l \to r \in R, d \in \Lambda_n, \alpha_i : t_i \to s_i \text{ for } i = 1, \ldots, n}{d(\alpha_1, \ldots, \alpha_n) = l(\alpha_1, \ldots, \alpha_n) \cdot d(s_1, \ldots, s_n) = d(t_1, \ldots, t_n) \cdot r(\alpha_1, \ldots, \alpha_n)}.$$

□

The *Interchange* axiom is applied to full sequents only: it states that, whenever we have the simultaneous execution of nested rewrites, it can be *simulated* as the *sequential* composition of two simpler rewrites. Its intuitive meaning is that rewriting at the top by means of a rule d, and rewriting "below", i.e. in the subterms matched by the left-hand side of the rule, are to a certain extent *independent* processes, and therefore can be executed in any order. Let us consider again the TRS $\mathcal{V}$: the proof term $d(d')$, corresponding to the parallel execution of the rewrites of f and a, is equated to the linearizations $f(d') \cdot d(b)$ and $d(a) \cdot g(d')$.

Fact 4.2 *Let us consider two full entailed sequents $\alpha : t \to s$, $\beta : u \to v$, and let $\cong_{E_2}$ be the minimal congruence obtained closing the axioms in E_2 with respect to substitution. The full proof terms α, β are equated if there exists a sequence $\alpha_i : t_i \to s_i$ for $i = 1 \ldots n$, of full entailed sequents such that $\alpha_i \cong_{E_2} \alpha_{i+1}$, $\alpha_1 = \alpha$, $\alpha_n = \beta$. Moreover, full entailed proof terms equated by the axioms in E_2 have the same source and target terms: if $\alpha : s \to t$ and $\beta : u \to v$ are full sequents entailed by $\mathcal{R}$, and $\alpha \cong_{E_2} \beta$, then $s \cong_E u$ and $t \cong_E v$. Thus, an abstract full sequent can be safely represented as a triple $\alpha : s \to t$.* □

A third relation over sequents can be obtained simply applying the axioms E_1 to full sequents. In this case, the theory entails the sequent $d(d') : f(a) \to g(b)$, describing the

simultaneous execution of two nested rewrites, but the three proof terms $d(d')$, $f(d') \cdot d(b)$ and $d(a) \cdot g(d')$ are not identified anymore: that's to say, we assume that our actual implementation is able to distinguish the concurrent execution of nested rewrites from any of its linearizations. In Chapter 6 we deal extensively with the analysis of the properties of the models from the point of view of a concurrent implementation. Again, for our purposes it is enough to note that these "abstractions" do not change the intuitive equivalence (from an operational point of view) of the rewrite relations induced by full and flat sequents, as shown by the following proposition, already stated in [Mes92].

Proposition 4.1 *Let $\mathcal{R} = \langle (\Sigma, E), L, R \rangle$ be a rewriting theory: it entails an abstract full sequent $\alpha : t \to s$, iff there exists a finite chain $\alpha_i : t_i \to s_i, i = 1, \ldots, n$ of abstract flat sequents such that $\alpha_1 \cdot \ldots \cdot \alpha_n : t \to s$, and $\alpha \cong_{E_2} \alpha_1 \cdot \ldots \cdot \alpha_n$.* □

4.3 $\mathcal{R}$-Systems

The classes of abstract sequents, both full and flat, represent possible models for our logic. Actually, they turn out to be a very special case of a more general notion. According to [Mes92], a reasonable model for the full entailment is defined as follows.

Definition 4.7 ($\mathcal{R}$-Systems) *Let $\mathcal{R} = \langle (\Sigma, E), L, R \rangle$ be a rewriting theory. An $\mathcal{R}$-system $\mathcal{S}$ is a category $\mathbf{S}$ together with*

- *a (Σ, E)-algebraic structure, i.e., for each $f \in \Sigma_n$ a functor $f_{\mathcal{S}} : \mathbf{S}^n \to \mathbf{S}$, preserving the equations in E: for any $t = s \in E$, the identity $t_{\mathcal{S}} = s_{\mathcal{S}}$ holds;*
- *for each rewrite rule $d : s \to t \in R$, a natural transformation $\alpha_{\mathcal{S}} : s_{\mathcal{S}} \Rightarrow t_{\mathcal{S}}$;*

where the functors $s_{\mathcal{S}}$, $t_{\mathcal{S}}$ are defined inductively from the basic functors $f_{\mathcal{S}}$.

An $\mathcal{R}$-homomorphism *$F : \mathcal{S} \to \mathcal{S}'$ is a functor $F : \mathbf{S} \to \mathbf{S}'$ preserving the algebraic structure (i.e., $f_{\mathcal{S}'} \circ F^n = F \circ f_{\mathcal{S}}$ for each $f \in \Sigma_n$) and the rewriting rules (i.e., given $id_F : F \to F$ the identity natural transformation, $id_F \circ \alpha_{\mathcal{S}} = \alpha_{\mathcal{S}'} \circ id_{F^n}$ holds for every rule $\alpha \in R$). $\mathcal{R}$-**Sys** denotes the category of $\mathcal{R}$-systems and $\mathcal{R}$-homomorphisms.* □

Then $\mathcal{R}$-systems are triples $\langle \mathbf{S}, \rho_S, \phi_S \rangle$ where $\rho_S = \{f_S \mid f \in \Sigma\}$ is a family of functors, and $\phi_S = \{d_S \mid d \in R\}$ is a family of natural transformations.

Some remarks are in order. As a start, with $\mathbf{S}^n$ we denote the n-fold product category of $\mathbf{S}$: its objects are n-tuples of objects of $\mathbf{S}$, and its arrows are n-tuples of arrows, with

source and target defined pointwise. In particular, $\mathbf{S}^0 = \mathbf{1}$, the category with one object and one arrow, terminal in **Cat**.

Moreover, with *identity* of two functors we mean that they identify objects "on-the-nose", instead of "up-to-isomorphism". Equivalently, this means that the two functors coincide in the (functor) category of functors and (natural) transformations.

Finally, let us consider the TRS $\mathcal{W}$: in any rewriting system $\mathcal{S}$, the functor associated to $h(x)$ and a are respectively $h_\Sigma : \mathbf{S} \to \mathbf{S}$ and $a_\Sigma : \mathbf{1} \to \mathbf{S}$. To provide a suitable natural transformation, we must realize that, when considering a as an object of $T_\Sigma(x)$, the associated functor is actually $a_{\Sigma,\{x\}} : \mathbf{S} \to \mathbf{S}$, with $a_{\Sigma,\{x\}} = !_S; a_\Sigma$, where $!_S : \mathbf{S} \to \mathbf{1}$ is uniquely determined. This discussion is analogous to the one about axiomatizations in Lawvere theories, and we refer the reader to Section 3.5.

This notion of model is reasonable, since, as put in [Mes92], it "captures the idea that the models of a rewrite theory are *systems*", i.e., a system is a "machine-like entity that can be in a variety of *states*, and that can change its state by performing certain *transitions*". This intuition is further confirmed by the following characterization of the initial model of $\mathcal{R}$-**Sys** (originally proved in [Mes92]).

Proposition 4.2 (Initial Model, I) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. The initial object $\mathcal{I}_\mathcal{R}$ of $\mathcal{R}$-**Sys** is the category having as objects equivalence classes of terms in the algebra $T_{(\Sigma,E)}$, and as arrows the elements of the class of abstract full sequents entailed by $\mathcal{R}$, where $\langle\alpha, s, t\rangle$ has source s and target t.* □

This characterization allows for an intuitive soundness and completeness result (also due to Meseguer), expressing the fact that any $\mathcal{R}$-system faithfully describes the full entailment relation.

Proposition 4.3 (Soundness and Completeness of $\mathcal{R}$-Systems) *Let $\mathcal{R}$ be a rewriting theory: it entails an abstract full sequent $\alpha : t \to s$ iff there exists a natural transformation $\alpha_{\mathcal{I}_\mathcal{R}} : t_{\mathcal{I}_\mathcal{R}} \Rightarrow s_{\mathcal{I}_\mathcal{R}}$ (then, iff there exists a natural transformation $\alpha_\mathcal{S} : t_\mathcal{S} \Rightarrow s_\mathcal{S}$ for each $\mathcal{R}$-system S).*

Proof We simply sketch the relevant cases.

- *(Sequent implies Transformation).* The proof is constructive, in the sense that any sequent $\alpha : t \to s$ inductively defines a natural transformation $\alpha_\mathcal{S} : t_\mathcal{S} \Rightarrow s_\mathcal{S}$ for each $\mathcal{R}$-system $\langle\mathbf{S}, \rho_S, \phi_S\rangle$. We proceed by induction on the structure of proof

terms. Let us assume that we have a sequent $\alpha = d(\alpha_1, \ldots, \alpha_n) : t(t_1, \ldots, t_n) \to s(s_1, \ldots, s_n)$. By hypothesis there exist $(\alpha_i)_S : (t_i)_S \Rightarrow (s_i)_S : \mathbf{1} \to \mathbf{S}$ natural transformations: then, we can build the natural transformation $\langle (\alpha_1)_S, \ldots, (\alpha_n)_S \rangle : \langle (t_1)_S, \ldots, (t_n)_S \rangle \Rightarrow \langle (s_1)_S, \ldots, (s_n)_S \rangle : \mathbf{1} \to \mathbf{S}^n$, where the components are defined pointwise. Then the natural transformation associated to α is $\langle (\alpha_1)_S, \ldots, (\alpha_n)_S \rangle * d_S$. The other cases are similar.

- *(Transformation implies Sequent).* The result is an immediate consequence of the characterization of the initial model $\mathcal{I}_{\mathcal{R}}$. ∎

As an example, let us consider a generic $\mathcal{R}$-system $\mathcal{S}$ associated to the TRS $\mathcal{V}$: the rules have associated natural transformations $d_{\mathcal{S}} : f_{\mathcal{S}} \Rightarrow g_{\mathcal{S}} : \mathbf{S} \to \mathbf{S}$ and $d'_{\mathcal{S}} : a_{\mathcal{S}} \Rightarrow b_{\mathcal{S}} : \mathbf{1} \to \mathbf{S}$. The abstract full sequents $d(d')$ is associated to the natural transformation $d_{\mathcal{S}} \circ d'_{\mathcal{S}} : a_{\mathcal{S}}; f_{\mathcal{S}} \Rightarrow b_{\mathcal{S}}; g_{\mathcal{S}} : \mathbf{1} \to \mathbf{S}$; it is uniquely defined thanks to the naturality requirement, identifying the natural transformations $(id_{a_S} \circ d_{\mathcal{S}}) \cdot (d'_{\mathcal{S}} \circ id_{g_S})$ and $(d'_{\mathcal{S}} \circ id_{f_S}) \cdot (id_{b_S} \circ d_{\mathcal{S}})$.

It is rather intuitive that the previous notion of model can be generalized in order to characterize also abstract flat sequents by dropping the naturality requirement.

Definition 4.8 (Flat $\mathcal{R}$-Systems) *Let $\mathcal{R} = \langle (\Sigma, E), L, R \rangle$ be a rewriting theory. A* flat $\mathcal{R}$-system $\mathcal{S}$ *is a category* $\mathbf{S}$ *together with*

- *a (Σ, E)-algebraic structure, i.e., for each $f \in \Sigma_n$ a functor $f_{\mathcal{S}} : \mathbf{S}^n \to \mathbf{S}$, preserving the equations in E: for any $t = s \in E$, the identity $t_{\mathcal{S}} = s_{\mathcal{S}}$ holds;*
- *for each rewrite rule $d : s \to t \in R$, a transformation $\alpha_{\mathcal{S}} : s_{\mathcal{S}} \Rightarrow t_{\mathcal{S}}$.*

where the functors $s_{\mathcal{S}}$, $t_{\mathcal{S}}$ are defined inductively from the basic functors $f_{\mathcal{S}}$.

A flat $\mathcal{R}$-homomorphism $F : \mathcal{S} \to \mathcal{S}'$ *is a functor $F : \mathbf{S} \to \mathbf{S}'$ preserving the algebraic structure (i.e., $f_{\mathcal{S}'} \circ F^n = F \circ f_{\mathcal{S}}$ for each $f \in \Sigma_n$) and the rewriting rules (i.e., such that the identity of transformations $F *_R \alpha_{\mathcal{S}} = \alpha_{\mathcal{S}'} *_L F^n$ holds for every rule $\alpha \in R$). $\mathcal{FR}$-**Sys** denotes the category of flat $\mathcal{R}$-systems and flat R-homomorphisms.* ∎

$\mathcal{FR}$-systems are triples $\langle \mathbf{S}, \rho_S, \phi_S \rangle$ where $\rho_S = \{f_S \mid f \in \Sigma\}$ is a family of functors, and $\phi_S = \{d_S \mid d \in R\}$ is a family of transformations; hence, $\mathcal{R}$-**Sys** is a sub-category of $\mathcal{FR}$-**Sys**. Actually, $\mathcal{R}$-**Sys** is *reflective* inside $\mathcal{FR}$-**Sys**: the inclusion functor has a left-adjoint, and the co-unit of the adjunction pair is a natural isomorphism. Note that the left-adjoint simply adds all the identifications necessary to make a transformation into a *natural* one: then Proposition 4.2 is an obvious consequence of the following result.

Proposition 4.4 (Initial Model, II) *Let $\mathcal{R} = \langle (\Sigma, E), L, R \rangle$ be a rewriting theory. The initial object $\mathcal{I}_{\mathcal{FR}}$ of $\mathcal{FR}$-**Sys** is the category having as objects equivalence classes of terms in the algebra $T_{(\Sigma,E)}$, and as arrows the elements of the class of abstract flat sequents entailed by $\mathcal{R}$.* □

Proof The forgetful functor $\mathcal{FR}$-**Sys** $\hookrightarrow$ **Cat**, associating to each $\mathcal{FR}$-system its underlying category, has a left-adjoint associating to each category **C** the free $\mathcal{FR}$-system obtained adding the Σ-structure. Since also the forgetful functor **Cat** $\hookrightarrow$ **Set**, associating to each category **C** its set of objects O_C has a left-adjoint associating to each set X the discrete category **X** (with only identity arrows), there is an adjoint pair $\langle F, G \rangle$ from $\mathcal{FR}$-**Sys** to **Set**. Since **0** is initial in **Set**, then $G(\mathbf{0})$ is initial in $\mathcal{FR}$-**Sys**. $G(\mathbf{0})$ has the structure of $\mathcal{I}_{\mathcal{FR}}$ (since there is an obvious one-to-one correspondence between the flat axioms and the coherence requirements for source, target and identity function in a category), then the thesis holds. □

The result proved in Proposition 4.3 can be easily reformulated for $\mathcal{FR}$-systems.

Proposition 4.5 (Soundness and Completeness of $\mathcal{FR}$-Systems) *Let $\mathcal{R}$ be a rewriting theory: it entails an abstract flat sequent $\alpha : t \to s$ iff there exists a transformation $\alpha_{\mathcal{I}_{\mathcal{PR}}} : t_{\mathcal{I}_{\mathcal{PR}}} \Rightarrow s_{\mathcal{I}_{\mathcal{PR}}}$ (then, iff there exists a transformation $\alpha_{\mathcal{S}} : t_{\mathcal{S}} \Rightarrow s_{\mathcal{S}}$ for each $\mathcal{FR}$-system S).* □

Note that, since $\mathcal{R}$-**Sys** is reflective inside $\mathcal{FR}$-**Sys**, then Proposition 4.4 implies Proposition 4.2, and Proposition 4.3 implies Proposition 4.5.

A more abstract notion of model can be defined, as already done for the category of (continuous) algebras in Chapter 3. The main concern of the next section will be for functorial models; we will introduce enriched structures such that their cells are in one-to-one correspondence with the sequents of a rewriting system. We first define a structure such that its set of cells is in one-to-one correspondence with the rewrite rules in $\mathcal{R}$ and, moreover, such that the underlying category is able to describe in a faithful way the structure of the initial algebra associated to a given equational theory.

4.4 Functorial Models of Rewriting Theories

Along the presentation in Section 3.5, an algebra can be considered as a "model" of a signature. The presentation of categories of algebras as functor categories makes this

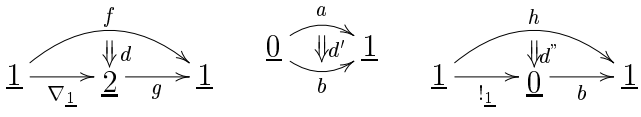
interpretation explicit, showing that categories of models in different universes (like **Set** and **Cpo**) can be taken into account. Essentially the same ideas can be applied to rewriting theories as well: such systems can be considered as syntactical specifications, and models for them are algebraic structures where all the "possible rewrites" have a suitable interpretation. Once again, this approach has the advantage of separating in a clear way what we can call the "$\mathcal{R}$-structure" (i.e., the algebraic structure defined by the equational theory, the axioms and the deduction rules of the rewriting logic) from the additional algebraic structure that can be enjoyed by the model. As for signatures, two kinds of categorical models for a system will be considered, namely the **Set**-based (in the last part of this section) and the **Cpo**-based (in Chapter 7). For a given rewriting theory $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$, the arrows of the Lawvere theory $\mathbf{Th}(\Sigma, E)$ are the "states" of the system (because those arrows represent terms of the algebra $T_{(\Sigma,E)}(X)$), and rewrites are *cells*, i.e., arrows between arrows.

Definition 4.9 (From Theories to Computads) *Let $\mathcal{R} = \langle(\Sigma, E), L, R\rangle$ be a rewriting theory. The associated* c-computad *$Th(\mathcal{R})$ is given by the pair $\langle\mathbf{Th}(\Sigma, E), R_c\rangle$, where $\mathbf{Th}(\Sigma, E)$ is the Lawvere theory associated to the equational theory (Σ, E) and R_c is a set of cells between the arrows of $\mathbf{Th}(\Sigma, E)$, such that $d : s \to t \in \mathcal{R}$ iff $d_c : s_{(\Sigma,E)} \Rightarrow t_{(\Sigma,E)} \in R_c$.* ∎

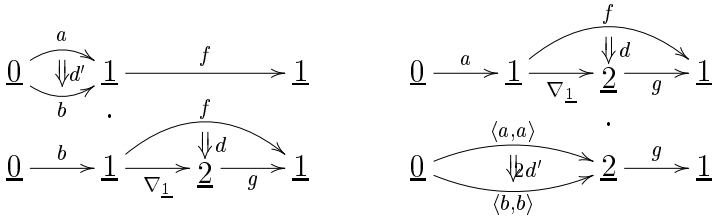
After seminal studies in the late Eighties (see e.g. [RS87, Pow89] and in particular [Mes90], as for the use of Lawvere theories; but also [See87, Str92, Ste94]), the correspondence between rewriting systems and c-computads has been (often implicitly!!) at the basis of many works on the semantics of rewriting. We freely generate an enriched category from a c-computad, such that its cells represent (equivalence classes of) sequences of rewrites.

Definition 4.10 (Spaces of Computations) *Let $\mathcal{R}$ be a rewriting theory and $Th(\mathcal{R})$ its associated c-computad. Then the associated* Lawvere 2-Theory $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}$ *is the cartesian 2-category $F_2(Th(\mathcal{R}))$ with finite products, while its* Lawvere S-Theory $\underline{\mathbf{S\text{-}Th}(\mathcal{R})}$ *is the cartesian sesqui-category $F_s(Th(\mathcal{R}))$ with finite products.* ∎

In a moment we will make more precise the relationship between these enriched categories and the models presented in the previous section. Note only that they describe *different* models, since they impose different equivalences over cells, due to the interchange axiom. Let us consider the rewriting theory $\mathcal{W} = \{d(x) : f(x) \to g(x, x), d' : a \to b, d''(x) : h(x) \to c\}$; the computad $Th(\mathcal{W})$ has the following set of cells



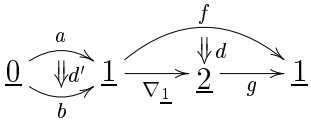
First of all, note the importance of duplicator $\nabla_{\underline{1}}$ and terminal arrow $!_{\underline{1}}$ for the correspondence between deduction rules and cells: it is the same problem we dealt with when defining $\mathcal{R}$-systems. Note also that cells such as $(d' *_R f) \cdot (b *_L d)$ and $(a *_L d) \cdot (d' *_R (\nabla_1; g)) = (a *_L d) \cdot (\langle d', d' \rangle *_R g)$, originating from $a; f : \underline{0} \to \underline{1}$ (the arrow associated to the term $f(a)$), belong to both spaces of computations. The associated cells can be graphically represented as:



where $2d' = \langle d', d' \rangle$. The diagrams should make clear the different meaning of left and right composition: left-composition instantiates a rewrite (for example, $a *_L d$ corresponds to instantiating $d(x)$ with $\{x/a\}$); while right-composition inserts a rewrite in a context ($d' *_R f$ inserts d' in the context $f(-)$).

The two cells are different in the sesqui-category $\underline{\mathbf{S\text{-}Th}(\mathcal{W})}$, while in the 2-category $\underline{\mathbf{2\text{-}Th}(\mathcal{W})}$ they are equated, thanks to the interchange axiom. The same happens to the cells $(d' *_R h) \cdot (b *_L d'')$ and $(a *_L d'') \cdot (d' *_R (!_{\underline{1}}; b)) = (a *_L d'')$ originating from the morphism $a; h$.

The equations above suggest that sesqui-categories are suitable models of flat entailment. As an example, there is no intuitive semantic counterpart for the abstract full sequent $d(d') : f(a) \to g(b, b)$ in $\underline{\mathbf{S\text{-}Th}(\mathcal{W})}$. On the contrary, 2-categories are models for the full entailment relation. Since the interchange axiom holds, the cell associated to that sequent is given by $(d' *_R f) \cdot (b *_L d) = (a *_L d) \cdot (d' *_R (\nabla_1; g)) = d * d'$. That cell is graphically represented as:



Proposition 4.6 (Correspondence with Algebraic Models, I) *Let $\mathcal{R}$ be a rewriting theory. Then there exists a bijective function ϕ between the set of all abstract* full *sequents entailed by $\mathcal{R}$ and the cells in $\underline{\mathbf{2}\text{-}\mathbf{Th}(\mathcal{R})}[\underline{0}, \underline{1}]$, such that $\phi(\alpha) : s_{(\Sigma,E)} \to t_{(\Sigma,E)}$ iff $\alpha : s \to t$.*

Proof The proof is constructive, and proceeds by induction on the structure of proof terms and cells, respectively; it is analogous to the (sequent implies transformation) side of the proof carried out for $\mathcal{R}$-systems in Proposition 4.3. Note however that it is fundamental that $\underline{\mathbf{2}\text{-}\mathbf{Th}(\mathcal{R})}$ has finite products, since it provides a suitable structure on cells that is respectful of the finite products on terms. ∎

Models of the Lawvere theory of a signature are cartesian functors to a suitable cartesian category with chosen finite products. Similarly, we can define the models of the Lawvere 2-theory and S-theory associated to a rewriting theory $\mathcal{R}$ as functors to a suitable universe; however, those functors (as well as the corresponding natural transformations) have to preserve the relevant structure, which is now much richer.

Definition 4.11 (Models of Lawvere 2-Theories) *Let $\mathcal{R}$ be a rewriting theory, and $\underline{\mathbf{C}}$ a cartesian 2-category with chosen products. A* $\underline{\mathbf{C}}$-model *for the Lawvere 2-theory associated to $\mathcal{R}$ is a chosen 2-functor $\mathcal{M} : \underline{\mathbf{2} - \mathbf{Th}(\mathcal{R})} \to \underline{\mathbf{C}}$, while a* model morphism *is a 2-natural transformation between models. The category $\underline{\mathbf{C}}\text{-}\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{C}}$-models is the 2-functor category $[\underline{\mathbf{2} - \mathbf{Th}(\mathcal{R})} \to \underline{\mathbf{C}}]$: its objects are $\underline{\mathbf{C}}$-models of $\mathcal{R}$, while its arrows are model morphisms.* ∎

As **Set** is the paradigmatic example of category, so $\underline{\mathbf{Cat}}$ is the paradigmatic example of 2-category: the objects are small categories, the arrows are functors, and the 2-cells are natural transformations. Since $\underline{\mathbf{Cat}}$ is cartesian with chosen products (where the monoidal operator $\mathbf{C} \otimes \mathbf{D}$ is just given by the product category $\mathbf{C} \times \mathbf{D}$), we are allowed to consider $\underline{\mathbf{Cat}}$-models of a rewriting theory $\mathcal{R}$; they are nothing else than the $\mathcal{R}$-models introduced in the previous section.

Proposition 4.7 (The Functor Category of $\mathcal{R}$-Systems) *Let $\mathcal{R}$ be a rewriting theory. The category of $\mathcal{R}$-systems (see Definition 4.7) is equivalent to the category $\underline{\mathbf{Cat}}\text{-}\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{Cat}}$-models for $\mathcal{R}$.* ∎

Proof Any $\mathcal{R}$-system $\langle \mathbf{S}, \rho_S, \phi_S \rangle$ induces a functor $\mathcal{M}_S : \underline{\mathbf{2-Th}(\mathcal{R})} \to \underline{\mathbf{Cat}}$, such that $\mathcal{M}_S(1) = \mathbf{S}$, while arrows and cells are defined accordingly. Also the converse holds, in the sense that each functor $\mathcal{M}$ induces an $\mathcal{R}$-system whose underlying category is $\mathbf{S}_{\mathcal{M}} = \mathcal{M}(1)$, while functions and natural transformations are defined accordingly. □

This result (in a different form) was already stated in [Mes90], and provides an elegant characterization of $\mathcal{R}$-systems. Note however that the categories are just *equivalent*, since the functor category is much larger than the other: to get an isomorphism we need to restrict the functor category, in order to take into account only one representative for each class of chosen functors that differ only for the choice of the monoidal natural transformations associated. For example, if we consider two functors $\mathcal{M}_1, \mathcal{M}_2$ such that $\mathcal{M}_1(1) = \mathcal{M}_2(1)$, they induce the same $\mathcal{R}$-system, even if they may be different: however, they are isomorphic by a natural transformation.

All the previous results given for the full entailment relation are easily reformulated to take into account the flat entailment. They are stated in the following, without proof.

Proposition 4.8 (Correspondence with Algebraic Models, II) *Let $\mathcal{R}$ be a rewriting theory. Then there exists a bijective function ξ between the set of all abstract* flat *sequents entailed by $\mathcal{R}$ and the cells in $\underline{\mathbf{S\text{-}Th}(\mathcal{R})}[\underline{0}, \underline{1}]$, such that $\xi(\alpha) : s_{(\Sigma,E)} \to t_{(\Sigma,E)}$ iff $\alpha : s \to t$.* □

As explained in Chapter 3, **Cat** can also be equipped with a different enriched structure: we indicate with $\underline{\mathbf{Cat}}_S$ the sesqui-category with small categories as objects, functors as arrows and transformations as cells.

Definition 4.12 (Models of Lawvere S-theories) *Let $\mathcal{R}$ be a rewriting theory, and $\underline{\mathbf{C}}_S$ a cartesian sesqui-category with chosen products. A* $\underline{\mathbf{C}}_S$-model *for the Lawvere s-theory associated to $\mathcal{R}$ is a chosen s-functor $\mathcal{M} : \underline{\mathbf{S\text{-}Th}(\mathcal{R})}_S \to \underline{\mathbf{C}}_S$, while a* model morphism *is a s-natural transformation between models. The category $\underline{\mathbf{C}}_S$-$\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{C}}_S$-models is the s-functor category $[\underline{\mathbf{S\text{-}Th}(\mathcal{R})}_S \to \underline{\mathbf{C}}_S]$: its objects are $\underline{\mathbf{C}}_S$-models of $\mathcal{R}$, while its arrows are model morphisms.* □

The results proved for $\mathcal{R}$-systems are easily extended to flat $\mathcal{R}$-systems.

Proposition 4.9 (The Functor Category of Flat $\mathcal{R}$-Systems) *Let $\mathcal{R}$ be a rewriting theory. The category of flat $\mathcal{R}$-systems (see Definition 4.8) is equivalent to the category $\underline{\mathbf{Cat}}_S$-$\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{Cat}}_S$-models for $\mathcal{R}$.* □

Chapter 5

Consistency with Finitary Rewriting

In the previous chapter we introduced the basic notions of rewriting logic, developing the categorical side of the Lambek-Lawvere analogy we devised in the introduction. The first aim of this chapter instead is to discuss some of the aspects of the set-theoretical side of the analogy, taking into account the classical approach to term rewriting.

The basic notion of the set-theoretical approach to term rewriting is that of *redex* of a term t: a pair (w, d) where w is an occurrence of t, and d is a rule such that its left-hand side matches the subterm t/w; such a redex *rewrites* t to a term s, obtained by substituting the subterm t/w with a suitable instantiation of the right-hand side of the rule. A *derivation* is just a suitable chain of rewrites, while the *derivation space* of a term t is just the set of co-initial derivations, starting from t. Derivation spaces represent a very intuitive operational model for TRS's: in Section 1 we will show that these structures are in one-to-one correspondence with a particular class of sequents entailed by a rewriting theory.

Derivation spaces are easily defined, but, unfortunately, the resulting operational semantics is usually too concrete: in order to obtain a description as much as possible independent from the actual execution of the reduction process performed (implemented) by a given machine, we need to abstract away from irrelevant details. Usually, such a description is recovered imposing a suitable equivalence relation on derivations, equating sequences of rewrites that are the same up to some conditions. Those conditions express the properties of the reduction mechanism over the system under examination: each equivalence class represents an *abstract derivation*, corresponding to a family of computationally equivalent sequences of rewrites.

Depending on the conditions we choose to take into account, we get different equivalences. We open Section 2 recalling the definition of the most famous one, the so-called

permutation equivalence [Lev80, Bou85]: it equates derivations that are the same up to permutation of *compatible* rewrites. Then, we introduce here a new equivalence we call *disjoint equivalence*, equating derivations that are the same up to permutation of *disjoint* rewrites. In the last part of Section 2 we show that these equivalences can be characterized also in terms of suitable axiomatizations over sequents: we will show that for any term t there is a one-to-one correspondence between the families of permutation equivalent derivations originating from t, and the families of abstract full sequents with source t. A careful inspection of the proof allows us to extend the result also to the other equivalence: namely, that for any term t there is a one-to-one correspondence between the families of disjoint equivalent derivations originating from t, and the families of abstract flat sequents with source t.

5.1 Consistency between Operational Semantics

Let us consider the definition of derivation given in Definition 2.17: simultaneous executions of redexes cannot be taken into account, since in any rewrite only one rule is applied. This is confirmed by the particular class of sequents corresponding to sequential derivations. We first need some definitions.

Definition 5.1 (Classes of Sequents) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *A proof term α is* one-step *if it does not contain the operator "$\cdot$", i.e., it is a term of the algebra $T_{\mathcal{O}} = T_{(\Sigma \cup \Lambda)}$; it is* linear *if it is one-step and contains exactly one operator in Λ; it is* many steps *if $\alpha = \alpha_1 \cdot \ldots \cdot \alpha_n$ with $1 \leq n < \omega$ and α_i is one-step for each $i \in \{1, \ldots, n\}$*[1]*; finally, it is* sequential *if it is many steps and all the component one-step proof terms are linear. A sequent $\alpha : t \to s$ is* one-step (linear, many steps, sequential) *if so is α.* □

The rules of deduction we now introduce constitute the fragment of rewriting logic necessary to describe just the application of a redex to a term or a derivation from a term: they allow to derive only sequential sequents.

Definition 5.2 (Sequential Rewriting Logic) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *We say that $\mathcal{R}$* entails *the sequential sequent α: $s \to t$ if it can be obtained by a finite number of applications of the following rules of deduction:*

[1] The application order in this case is not influent. Note however that there are sequents that are neither one-step, nor many-steps.

- (Flat Instantiation)

$$\frac{d : l \to r \in R, d \in \Lambda_n, t_i \in T_\Sigma \text{ for } i = 1, \ldots, n}{d(t_1, \ldots, t_n) : l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)};$$

- (Linear Congruence)

$$\frac{f \in \Sigma_n, \alpha : s \to s', \alpha \in T_{\mathcal{O}}, t_j \in T_\Sigma \text{ for } j \in \{1, \ldots, n\} \backslash i}{f(t_1^{i\perp 1}, \alpha, t_{i+1}^n) : f(t_1^{i\perp 1}, s, t_{i+1}^n) \to f(t_1^{i\perp 1}, s', t_{i+1}^n)};$$

- (Transitivity)

$$\frac{\alpha : s \to t, \ \beta : t \to u}{\alpha \cdot \beta : s \to u}.$$

where t_p^q, $p \leq q$, stands for the tuple $t_p, \ldots, t_q$. □

The following proposition states the precise relationship between the classical presentation of rewriting and the one using sequents (in the sequential case).

Proposition 5.1 (Sequential Sequents and Derivations) *Let $\mathcal{R}$ be a* TRS. (1) *If Δ is a redex of t and $t \to_\Delta s$, then there is a linear proof term α_Δ such that $\mathcal{R}$ entails the sequent $\alpha_\Delta : t \to s$ (using the rules of Definition 5.2). Viceversa,* (2) *if $\mathcal{R}$ entails a linear sequent $\alpha : t \to s$, then there is a redex Δ_α of t such that $t \to_{\Delta_\alpha} s$. Hence, there is a derivation from t to t' iff $\mathcal{R}$ entails a sequential sequent $\alpha : t \to t'$.*

Proof The proof is constructive, in the sense that we inductively define a function over linear proof terms (redexes) that returns the associated redex (proof term, respectively).

1. Let $\Delta = (w, d : l \to r)$: then $t(u) = s(u)$ for each $u \not\geq w$, and there exists $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ such that $l\sigma = t/w$, $r\sigma = s/w$. By the instantiation rule, we have $d(t_1, \ldots, t_n) : t/w \to s/w$. Then, to construct α_Δ, we just need the function ξ_t, parametric over the terms in T_Σ:

$$\xi_t(\Delta) = \begin{cases} d(t_1, \ldots, t_n) & \text{if } w = \lambda; \\ f(s_1^{i\perp 1}, \xi_{s_i}((w', d)), s_{i+1}^n) & \text{if } w = iw' \text{ and } t = f(s_1, \ldots, s_n). \end{cases}$$

We say $\alpha_\Delta = \xi_t(\Delta)$: it is well defined, and it is easy to check that $\alpha_\Delta : t \to s$.

2. We define instead a function χ over proof terms:

$$\chi(\alpha) = \begin{cases} (\lambda, d) & \text{if } \alpha = d(t_1, \ldots, t_n); \\ (iw, d) & \text{if } \alpha = f(s_1^{i\perp 1}, \alpha', s_{i+1}^n) \text{ and } \chi(\alpha') = (w, d). \end{cases}$$

We say $\Delta_\alpha = \chi(\alpha)$. The soundness of χ is easily proved. Let us assume that, in the first case, $\alpha = d(t_1, \ldots, t_n) : l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)$. By definition $l\sigma = t$ and $r\sigma = s$, where $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, so that σ is the substitution associated to (λ, d). In the second case the corresponding substitution is the same inductively associated to $\chi(\alpha')$. □

Note that in the first part of the proof it was fundamental to consider the term to which the redex is applied, in order to get the right context for the proof term of the associated sequent. We remark again that the proof is constructive. In fact, we actually defined a one-to-one correspondence between sequential sequents and sequential derivations, supporting the claim that the two operational descriptions of term rewriting are actually equivalent.

Proposition 5.2 (Equivalence between Operational Models, I) *Let $\mathcal{R}$ be the* TRS *$\langle \Sigma, L, R \rangle$. Then for each $t \in T_\Sigma$ there is a one-to-one correspondence between the families of sequential derivations entailed by $\mathcal{R}$ originating from t, and the families of sequential sequents with source t.*

Proof Given $\Delta = (w, d : l \to r)$ redex of t, we want to show that $\chi(\xi_t(\Delta)) = \Delta$. We proceed by induction on the length of w. The inductive base $w = \lambda$ is obvious. Let us assume $w = iw'$: $t = f(t_1, \ldots, t_n)$, $\Delta' = (w', d)$ is a redex of t_i and by induction hypothesis $\chi(\xi_{t_i}(\Delta')) = \Delta'$. The thesis immediately follows.

Conversely, we want to show that $\xi_t(\chi(\alpha)) = \alpha$. We proceed by induction on the structure of α. If $\alpha = d(t_1, \ldots, t_n)$, then $\chi(\alpha) = (\lambda, d)$ with substitution $\sigma = \{x_1/t_1 \ldots x_n/t_n\}$, and by definition the thesis holds. If $\alpha = f(s_1^{i\perp 1}, \alpha', s_{i+1}^n)$, then by induction hypothesis α' has associated a redex $\chi(\alpha') = (w, d)$ with substitution σ, such that $\xi_{s_i}(\chi(\alpha')) = \alpha'$. Then $\chi(\alpha) = (iw, d)$ with substitution σ, and by definition of ξ_t the thesis holds. □

The previous proof is quite straightforward, but it is paradigmatic of the analogous, but more difficult ones that we will state in the rest of the chapter. Let us now consider again rewriting logic: small changes to the deduction rules of Definition 5.2 are sufficient to generate many-steps sequents.

Definition 5.3 (Parallel Rewriting Logic) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *We say that $\mathcal{R}$ entails the many-steps full sequent $\alpha : s \to t$ if it can be obtained by a finite number of applications of the following rules of deduction:*

- (Elementary Instantiation)
$$\frac{d : l \to r \in R, d \in \Lambda_n, \alpha_i : t_i \to s_i, \alpha_i \in T_{\mathcal{O}},\ \textit{for } i = 1, \ldots, n}{d(\alpha_1, \ldots, \alpha_n) : l(t_1, \ldots, t_n) \to r(s_1, \ldots, s_n)};$$
- (Elementary Congruence)
$$\frac{f \in \Sigma_n, \alpha_i : t_i \to s_i, \alpha_i \in T_{\mathcal{O}},\ \textit{for } i = 1, \ldots, n}{f(\alpha_1, \ldots, \alpha_n) : f(t_1, \ldots, t_n) \to f(s_1, \ldots, s_n)};$$
- (Transitivity)
$$\frac{\alpha : s \to t,\ \beta : t \to u}{\alpha \cdot \beta : s \to u}.$$

□

We say that $\mathcal{R}$ entails a many-steps flat sequent if we substitute the Elementary Congruence rule with the Flat Instantiation rule of Definition 5.2

Definition 5.4 (Disjoint Rewriting Logic) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *We say that $\mathcal{R}$ entails the many-steps flat sequent $\alpha : s \to t$ if it can be obtained by a finite number of applications of the following rules of deduction:*

- (Flat Instantiation) *and* (Transitivity) *as in Definition 5.2;*
- (Elementary Congruence) *as in Definition 5.3.* □

Given a parallel redex Φ and an occurrence w, we define the *restriction of Φ at occurrence w* as the parallel redex $\Phi/w = \{(v, d) \mid (wv, d) \in \Phi\}$. The following proposition generalizes to the parallel case the relationship between the two presentations of rewriting, stated in Proposition 5.1.

Proposition 5.3 (Many-Steps Sequents and Parallel Derivations) *Let $\mathcal{R}$ be a left-linear* TRS. (1) *If Φ is a parallel redex of t and $t \to_\Phi s$, then there is a one-step proof term α_Φ such that $\mathcal{R}$ entails the full sequent $\alpha_\Phi : t \to s$ (using the rules of Definition 5.3). Viceversa,* (2) *if $\mathcal{R}$ entails a one-step full sequent $\alpha : t \to s$, then there is a parallel redex Φ_α of t such that $t \to_{\Phi_\alpha} s$. Hence, there is a parallel derivation from t to t' iff $\mathcal{R}$ entails a many-steps full sequent $\alpha : t \to t'$.*

Proof Also in this case we give a constructive proof, providing suitable functions over full proof term and parallel redexes.

1. Let $\Phi = \{\Delta_1, \ldots, \Delta_m\}$ be the redex under examination, such that the generic i-th redex is given by the pair (w_i, d_i). Moreover, let us assume it is *ordered*, in the sense that $1 \leq i < j \leq n$ implies that either $w_i < w_j$ or $w_i | w_j$. Then $t(u) = s(u)$ for each $u \not\geq w_1$, and there exists $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ such that $l_1\sigma = t/w_1$, $r_1\sigma = s/w_1$. By the instantiation rule we have $d(t_1, \ldots, t_n) : t/w_1 \to s/w_1$, where $t/w_1\mathcal{O}_{x_i}(l_1) = t_i = s/w_1u_{x_i}$ for $u_{x_i} \in \mathcal{O}_{x_i}(r_1)$. Also in this case we use a function κ_t, parametric over the class of terms in T_Σ and defined over sets of compatible redexes; if we assume $t = f(s_1, \ldots, s_n)$,

$$\kappa_t(\Phi) = \begin{cases} t & \text{if } \Phi = \emptyset; \\ f(\kappa_{s_1}(\Phi/1), \ldots, \kappa_{s_n}(\Phi/n)) & \text{if } w_1 \neq \lambda; \\ d_1(\kappa_{t_1}(\Phi/\mathcal{O}_{x_1}(l_1)), \ldots, \kappa_{t_n}(\Phi/\mathcal{O}_{x_n}(l_1))) & \text{if } w_1 = \lambda \end{cases}$$

   We define α_Φ as $\kappa_t(\Phi)$: it is well defined, and it is proved by induction on Φ that $\alpha_\Phi : t \to s$. The idea is that $\kappa_t(\Phi)$ describes the simultaneous application of all the redexes in Φ. In particular, in the case $w_1 = \lambda$, each $\Phi/\mathcal{O}_{x_i}(l_1)$ describes the redexes applicable to t_i; and since by definition $t/\mathcal{O}_{x_i}(l_1) = t_i = s/u_{x_i}$ for $u_{x_i} \in \mathcal{O}_{x_i}(r_1)$, if by induction hypothesis the various $\kappa_{t_i}(\Phi/\mathcal{O}_{x_i}(l_1))$ are well-defined, so is $\kappa_t(\Phi)$.

2. We define a function ψ over proof terms; if we assume $\psi(\alpha_i) = (w_i, d_i)$, then

$$\psi(\alpha) = \begin{cases} \{(\lambda, d)\} \cup \bigcup_{i=1}^n \{(\mathcal{O}_{x_i}(l_1)w_i, d_i)\} & \text{if } \alpha = d(\alpha_1, \ldots, \alpha_n); \\ \bigcup_{i=1}^n (iw_i, d_i) & \text{if } \alpha = f(\alpha_1, \ldots, \alpha_n). \end{cases}$$

   We say that $\Delta_\alpha = \psi(\alpha)$. The soundness is proved by induction on the number of redexes. Let us assume that, in the first case, $\alpha = d(t_1, \ldots, t_n) : l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)$. By definition $l\sigma = t$ and $r\sigma = s$, where $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, so that σ is the substitution associated to (λ, d). In the second case the corresponding substitution is the same inductively associated to $\chi(\alpha')$. □

Also for many-steps the correspondence is one-to-one, since we provided a suitable sequent, equivalent to the parallel execution of all the redexes of the compatible set.

Proposition 5.4 (Equivalence between Operational Models, II) *Let $\mathcal{R}$ be the left-linear* TRS *$\langle \Sigma, L, R \rangle$. Then for each $t \in T_\Sigma$ there is a one-to-one correspondence between the families of parallel derivations entailed by $\mathcal{R}$ originating from t, and the families of many-steps full sequents with source t.*

Proof We proceed by induction on Φ. The base case $\Phi = \{\Delta\}$ is proved by Proposition 5.2. Let us assume that the redex Φ is ordered, and that all the hypotheses of the proof of 5.3 hold. Then we proceed by induction on w_1. If $w_1 \neq \lambda$ and $t = f(s_1, \ldots, s_n)$, by definition $\kappa_t(\Phi) = f(\kappa_{s_1}(\Phi/1), \ldots, \kappa_{s_n}(\Phi/n))$; by induction hypothesis, $\psi(\kappa_{s_i}(\Phi/i)) = \Phi/i$, and by the definition of ψ, the result holds. The case $w_1 = \lambda$ is analogous.

Conversely, we want to show that $\kappa_t(\psi(\alpha)) = \alpha$. We proceed by induction on the structure of α. If $\alpha = f(\alpha_1, \ldots, \alpha_n) : f(t_1, \ldots, t_n) \to f(s_1, \ldots, s_n)$, then by definition there are sequents $\alpha_i : t_i \to s_i$, and by induction hypothesis each α_i has associated a redex $\Psi(\alpha_i) = (w_i, d_i)$ with substitution σ_i, such that $\kappa_{t_i}(\Psi(\alpha_i)) = \alpha_i$. Then by definition of $\kappa_t(f(\alpha_1, \ldots, \alpha_n))$ the thesis holds. The case $\alpha = d(\alpha_1, \ldots, \alpha_n)$ for d rule is analogous. □

Similar properties also hold for disjoint derivations: the following results can be easily recovered just by analyzing the corresponding ones for compatible redexes, and are stated without proof.

Proposition 5.5 (Many-Steps Sequents and Disjoint Derivations) *Let $\mathcal{R}$ be a* TRS. (1) *If Φ is a disjoint redex of t and $t \to_\Phi s$, then there is a one-step proof term α_Φ such that $\mathcal{R}$ entails the flat sequent $\alpha_\Phi : t \to s$ (using the rules of Definition 5.4). Viceversa,* (2) *if $\mathcal{R}$ entails a one-step flat sequent $\alpha : t \to s$, then there is a parallel redex Φ_α of t such that $t \to_{\Phi_\alpha} s$. Hence, there is a disjoint derivation from t to t' iff $\mathcal{R}$ entails a many steps flat sequent $\alpha : t \to t'$.* □

Proposition 5.6 (Equivalence between Operational Models, III) *Let $\mathcal{R}$ be the left-linear* TRS *$\langle \Sigma, L, R \rangle$. Then for each $t \in T_\Sigma$ there is a one-to-one correspondence between the families of disjoint derivations entailed by $\mathcal{R}$ originating from t, and the families of many-steps flat sequents with source t.* □

5.2 Consistency between Abstract Semantics

We open this section recalling the definition of permutation equivalence.

Definition 5.5 (Permutation Equivalence) *Let $\mathcal{R}$ be a left-linear* TRS. *Permutation equivalence (indicated as $\equiv_p$) is the least equivalence relation over parallel derivations satisfying:*

1. *if $\Phi ||_c \Phi'$, then $\rho_\Phi \cdot \rho_{\Phi' \setminus \Phi} \equiv_p \rho_{\Phi'} \cdot \rho_{\Phi \setminus \Phi'}$;*

2. *if* $\rho \equiv_p \rho'$, *then* $\tau \cdot \rho \cdot \tau' \equiv_p \tau \cdot \rho' \cdot \tau'$ *for* τ, τ' *derivations.*

where Φ, Φ' *are compatible sets of redexes, and* $\rho_\Phi, \rho_{\Phi'}$ *any of their complete developments.* □

Thanks to Proposition 2.13, the definition is well-given. It can be obviously reformulated to deal only with disjoint derivations.

Definition 5.6 (Disjoint Equivalence) *Let* $\mathcal{R}$ *be a* TRS. *Disjoint equivalence (indicated as* $\equiv_d$*) is the least equivalence relation over disjoint derivations satisfying:*

1. *if* $\Phi || \Phi'$, *then* $\rho_\Phi \cdot \rho_{(\Phi' \perp \Phi)} \equiv_d \rho_{\Phi'} \cdot \rho_{(\Phi \perp \Phi')}$;

2. *if* $\rho \equiv_d \rho'$, *then* $\tau \cdot \rho \cdot \tau' \equiv_d \tau \cdot \rho' \cdot \tau'$ *for* τ, τ' *derivations.*

where Φ, Φ' *are disjoint sets of redexes, and* $\rho_\Phi, \rho_{\Phi'}$ *any of their complete developments.* □

Disjoint equivalence has a much simpler formulation, since given any two set Φ, Φ' of disjoint redexes, it is easy to prove that the residual $\Phi \backslash \Phi'$ corresponds to the set-difference $\Phi - \Phi'$.

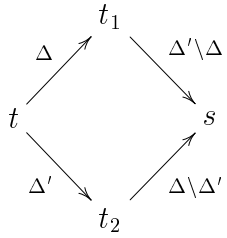
The following result was originally proved in [LM92].

Proposition 5.7 (Correspondence between Abstract Models, I) *Let* $\mathcal{R}$ *be the left-linear* TRS $\langle \Sigma, L, R \rangle$. *Then for each* $t \in T_\Sigma$ *there is a one-to-one correspondence between the families of permutation equivalent derivations in* $\mathcal{R}$ *originating from* t, *and the families of abstract full sequents with source* t *entailed by* $\mathcal{R}$. □

In our setting, this is proved in a simpler way than in the original paper. A first step is to show that the one-to-one correspondence between operational models stated in Proposition 5.4 preserve permutation equivalence.

Proof We proceed by induction on the length of the derivation (number of one-step sequents, respectively).

(*Permutation implies Abstractness*). All we need to show is that, given a set Φ of compatible redexes, then for any two complete developments ρ_Φ and ρ'_Φ of Φ the proof terms $\kappa_t(\rho_\Phi)$ and $\kappa_t(\rho'_\Phi)$ are equated. We proceed by induction on Φ. The base case is $\Phi = \{\Delta, \Delta'\}$, and the situation is indicated by the following diagram:



The case $\Delta||\Delta'$ is obvious, so let us assume $\Delta = (w, d)$ and $\Delta' = (w', d')$, with $w \leq w'$ and $w' = w \cdot \mathcal{O}_{x_i}(l) \cdot v$ for some $x_i \in var(l)$. By definition

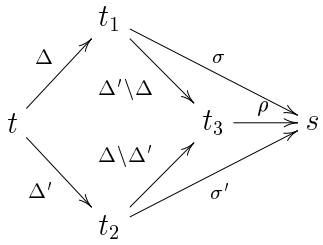
$$\kappa_t(\Phi) = t[w \leftarrow d(t_1^{i\perp 1}, \kappa_{t_i}(\Delta'), t_{i+1}^n)] = t[w \leftarrow d(t_1^{i\perp 1}, t_i[v \leftarrow d'(s_1, \ldots, s_m)], t_{i+1}^n)].$$

Then by construction $\kappa_t(\Delta') = t[w' \leftarrow d'(s_1, \ldots, s_m)]$, with the same substitution associated to $\kappa_{t_i}(\Delta')$, and $t_2 = t[w' \leftarrow r(s_1, \ldots, s_m)]$. Also by definition

$$\kappa_{t_2}(\Delta) = t[w \leftarrow d(t_1^{i\perp 1}, t_i[v \leftarrow r(s_1, \ldots, s_m)], t_{i+1}^n)].$$

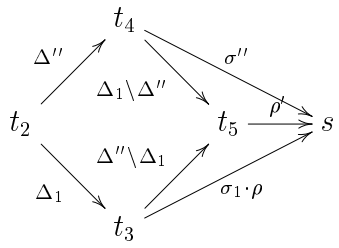
Finally, thanks to the interchange axiom we have $\kappa_t(\Phi) = \kappa_t(\Delta') \cdot \kappa_{t_2}(\Delta)$. Analogously, $\kappa_t(\Phi) = \kappa_t(\Delta) \cdot \kappa_{t_1}(\Delta')$.

Now, for a generic Φ, let us consider two developments $\rho_\Phi = \Delta \cdot \sigma$ and $\rho'_\Phi = \Delta' \cdot \sigma'$. By construction we have $\Phi = \{\Delta\} \cup \{\Delta'\} \cup \Phi'$, and a situation represented by the following diagram:



By construction ρ is a development of $\Phi\backslash\{\Delta, \Delta'\}$, while σ (σ') is a development of $\Phi\backslash\Delta$ ($\Phi\backslash\Delta'$). Now, if both $\Phi\backslash\Delta$ and $\Phi\backslash\Delta'$ have a smaller number of redexes than Φ (e.g., if Δ and Δ' are disjoint with the other redexes of Φ, or they are right-linear), then by induction hypothesis $\kappa_t(\sigma) = \kappa_t((\Delta'\backslash\Delta) \cdot \rho)$ and $\kappa_t(\sigma') = \kappa_t((\Delta\backslash\Delta') \cdot \rho)$. Since by induction we also have $\kappa_t(\Delta) \cdot \kappa_{t_1}(\Delta'\backslash\Delta) = \kappa_t(\Delta') \cdot \kappa_{t_2}(\Delta\backslash\Delta')$, then the thesis holds. If instead for example $\Phi\backslash\Delta'$ has more redexes than Φ, then we apply this procedure to $\sigma' = \Delta'' \cdot \sigma''$; if $\Delta_1 \cdot \sigma_1$ is a complete development of $\Delta\backslash\Delta'$, then

we have a situation of this kind:



Surely $\kappa_{t_2}(\Delta'')\cdot\kappa_{t_4}(\Delta''\backslash\Delta_1) = \kappa_{t_2}(\Delta_1)\cdot\kappa_{t_3}(\Delta_1\backslash\Delta'')$. Now we check the number of redexes of $(\Phi\backslash\Delta')\backslash\Delta''$ and $(\Phi\backslash\Delta')\backslash\Delta_1$, to eventually prove by induction that $\kappa_{t_4}(\sigma'') = \kappa_{t_4}((\Delta_1\backslash\Delta'')\cdot\rho')$ and $\kappa_t(\sigma_1\cdot\rho) = \kappa_{t_5}((\Delta''\backslash\Delta_1)\cdot\rho)$, hence $\kappa_t(\sigma') = \kappa_t((\Delta\backslash\Delta')\cdot\rho)$. Since all the possible developments of Φ are finite, the procedure is convergent, and we are then assured that the thesis holds.

(*Abstractness implies Permutation*). We proceed by induction on the last axiom applied. All the cases are obvious, except for interchange. Let assume for the sake of simplicity that we are applying the axiom on the top, i.e., that

$$\alpha = d(\alpha_1,\ldots,\alpha_n) = l(\alpha_1,\ldots,\alpha_n)\cdot d(s_1,\ldots,s_n) : t(t_1,\ldots,t_n)\to s(s_1,\ldots,s_n).$$

Then by construction the set of redexes $\psi(l(\alpha_1,\ldots,\alpha_n))\cup\psi(d(s_1,\ldots,s_n))$ and $\psi(d(\alpha_1,\ldots,\alpha_n))$ coincide. The case $d(\alpha_1,\ldots,\alpha_n) = d(t_1,\ldots,t_n)\cdot r(\alpha_1,\ldots,\alpha_n)$ is entirely analogous. □

After that, it is enough to note that, although the parallel rewriting logic of Definition 5.3 seems less powerful than full rewriting logic (since the sequents appearing in the premises of the congruence and instantiation rules are not bound to be one-step, but they can be arbitrary, and the operator "$\cdot$" can appear inside other operators in the proof term of a sequent), they entail the same families of abstract sequents.

Proposition 5.8 (Equivalence of Rewriting Logics, I) *Let $\mathcal{R}$ be a* TRS*: it entails a sequent $\alpha : t\to s$ in full rewriting logic iff it entails a many-steps sequent $\alpha' : t\to s$ in parallel rewriting logic, such that $\alpha\cong_{E_2}\alpha'$.*

Proof The proof can be found in Lemma 2.6 (and Lemma 3.6) of [Mes92]. Anyway, it is quite straightforward, proceeding by induction on the last rule applied. Let us just consider the sequent $f(\alpha_1,\ldots,\alpha_n) : f(t_1,\ldots,t_n)\to f(s_1,\ldots,s_n)$, such that Congruence is the last rule applied; by induction, each α_i can be decomposed in a finite chain of one-step sequents $\alpha_i^1\cdot\ldots\cdot\alpha_i^{k_i}$: then $\alpha' = f(\alpha_1^1,t_2\ldots,t_n)\cdot f(\alpha_1^2,t_2,\ldots,t_n)\cdot\ldots\cdot f(s_1,\ldots,s_{n\perp 1},\alpha_n^{k_n})$

is one of the possible many-steps proof terms corresponding to $f(\alpha_1, \ldots, \alpha_n)$, and $\alpha' =_{E_2} f(\alpha_1, \ldots, \alpha_n)$. □

An inspection of the proof of Proposition 5.7 shows that the following result holds.

Proposition 5.9 (Correspondence between Abstract Models, II) *Let $\mathcal{R}$ be the* TRS $\langle \Sigma, L, R \rangle$. *Then for each $t \in T_\Sigma$ there is a one-to-one correspondence between the families of disjoint equivalent derivations originating from t, and the families of abstract flat sequents with source t entailed by $\mathcal{R}$.* □

This is proved showing that the functions defined in Proposition 5.3 preserve the equivalences (as just done in Proposition 5.7), and then using the following result.

Proposition 5.10 (Equivalence of Rewriting Logics, II) *Let $\mathcal{R}$ be a* TRS*: it entails a sequent $\alpha : t \to s$ in flat rewriting logic iff it entails a many-steps sequent $\alpha' : t \to s$ in disjoint rewriting logic, such that $\alpha \cong_{E_1} \alpha'$.* □

Chapter 6

On Concurrent Rewriting

In this chapter we focus our attention on the concurrent aspects of the semantical models for the reduction mechanism proposed in Chapter 4 and Chapter 5. Lévy introduced *permutation equivalence* in his study of the *optimal reduction* strategies for λ-calculus. As we recalled in the introduction of the thesis, his approach was to discard the usual, syntactical representation of terms, in order to describe them as suitable *graph expressions*, where the *sharing* of subexpressions is explicit: subexpressions that should be "syntactically" copied in a reduction step are kept shared in the corresponding graphical contraction. In this setting, a single reduction step (on a graph) can then subsume a long sequence of (single) rewrites, and permutation equivalence equates precisely those sequences of syntactical rewrites that correspond to the same contraction on graphs.

In Section 6.3 we will argue about the adequacy of permutation equivalence in describing graph reduction, showing in particular the kind of problems arising when not orthogonal TRS's are taken into account. However, the main focus of the chapter is a discussion about the adequacy of disjoint and permutation equivalences in describing a suitable *concurrent semantics* of the reduction mechanism. In the concurrent approach we assume to have a distributed system: a network of (loosely) coupled processors, on which to implement the reduction process. Moreover, we also assume a minimal data structure, dealing implicitly with a *one-node/one-processor* architecture, where terms are described as trees, and we have a tree-like implementation schema. Since we aim at describing the simultaneous reduction of *compatible* redexes, it is then necessary to check out that the notion of compatibility is *directly* implementable, in the sense that without any further assumption on the structure of (the coupling of) the network, the simultaneous execution of two compatible redexes is feasible. In the concurrency area (and in Section 6.1 we will try to provide some intuitive motivations about that) such a property is considered

equivalent to show that the abstract derivation space forms a *prime algebraic domain* (PAD).

PAD's are a simple, general and well-accepted model to describe the behaviour of concurrent, non-deterministic systems. Their acceptance in the concurrency field is due to their tight correspondence with *prime event structures* (PES's, [Win89]). First introduced in the early Eighties, PES's are partial orders of "events", equipped with a "conflict" relation. Such structures are very suitable for describing the behaviour of distributed and non-deterministic computational devices generating instantaneous atomic events, which can be causally related or mutually exclusive. The level of abstraction they capture is considered as *directly* reflecting that of a possible, *concurrent* implementation, where each event corresponds to a *basic* action of the underlying machine. To each PES is associated a set of *configurations*: compatible, left-closed subsets of its events. Intuitively, a configuration corresponds to a specific state of the system reached after some computation, and its events are all those generated during that specific computation. A fundamental result due to Winskel shows that the set of all configurations of a PES ordered by set inclusion (its "domain" of configurations) forms a PAD; moreover, for each PAD there is a PES such that its domain of configurations is isomorphic to the given PAD. Thus the use of PAD's or PES's for the description of computational systems is equivalent.

Coming back to TRS's, both the classes of abstract full and flat sequents starting from a given initial term t can be equipped easily with a *prefix* pre-ordering: $\alpha \leq \beta$ iff there exists a proof term γ such that $\alpha \cdot \gamma \cong \beta$. In Section 2 we will show that only the prefix pre-order induced by flat entailment is a PAD, while that induced by full entailment fails to satisfy the "distributive property" of PAD's. Section 1 just provides a brief introduction to PAD's, while Section 3 gives some remarks on the interweaving about the algebraic description of terms and the notion of redexes compatibility: a topic we will deal again with in Chapter 9.

6.1 Permutation vs. Concurrency

In this section we analyze the algebraic properties of the abstract models we just introduced. We open the section introducing *Prime Algebraic Domains* (PAD's for short; see [Win89]). A PAD is a partial order verifying some additional properties. The use of partial orders in semantics relies on the old idea that a computing machine determines an ordered space of computations; the richer structure of PAD's, however, makes them especially suited for modeling distributed systems.

Definition 6.1 (Prime Algebraic Domains) *Let $\mathcal{D} = \langle D, \sqsubseteq \rangle$ be a* PO*:*

1. *The* least upper bound *of a set $X \subseteq D$ is an element $\bigsqcup X$ such that $x \leq \bigsqcup X$ for all $x \in X$, and such that for all $z \in D$, $(\forall x \in X \,.\, x \leq z) \Rightarrow \bigsqcup X \leq z$. We write $x \sqcup y$ for $\bigsqcup\{x, y\}$.*

2. *Symmetrically, the* greatest lower bound *of a set $X \subseteq D$ is an element $\bigsqcap X$ such that $\bigsqcap X \leq x$ for all $x \in X$, and such that for all $z \in D$, $(\forall x \in X \,.\, z \leq x) \Rightarrow z \leq \bigsqcap X$. We write $x \sqcap y$ for $\bigsqcap\{x, y\}$.*

3. *A* directed subset *of D is a subset $S \subseteq D$ such that for any finite subset $X \subseteq S$ there is an element $s \in S$ such that $\forall x \in X \,.\, x \leq s$.*

4. *An element $x \in D$ is* finite *if for all directed sets S, $x \sqsubseteq \bigsqcup S$ implies that there is some $s \in S$ such that $x \sqsubseteq s$.*

5. *$\mathcal{D}$ is* finitary *if for every finite element $x \in D$, the set $\{y \mid y \sqsubseteq x\}$ is finite.*

6. *An element $x \in D$ is* complete prime (prime) *if for each $X \subseteq D$ (each finite $X \subseteq D$), if $\bigsqcup X$ exists and $x \sqsubseteq \bigsqcup X$, then there exists an $y \in X$ such that $x \sqsubseteq y$.*

7. *$\mathcal{D}$ is* prime algebraic *if for all $x \in D$, $x = \bigsqcup\{y \sqsubseteq x \mid y \text{ is complete prime}\}$.*

8. *For $x, y \in D$, we write $x \uparrow y$ (and we say that x and y are* compatible*) if there exists a z such that $x \sqsubseteq z$ and $y \sqsubseteq z$. We say that $X \subseteq D$ is* pairwise compatible *if for all $x, y \in X$ we have $x \uparrow y$.*

9. *A finitary partial order $\langle D, \sqsubseteq \rangle$ is* distributive *if, whenever $x \uparrow y$, then we have $(x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z)$.*

10. *$\mathcal{D}$ is* finitely coherent *if it has* LUB*'s of* finite*, pairwise compatible subsets. $\mathcal{D}$ is a* coherent domain *if it has* LUB *'s of* arbitrary*, pairwise compatible subsets.* ∎

The following characterization is due to Winskel [Win87].

Fact 6.1 *Any coherent, finitary domain $\mathcal{D}$ is prime algebraic iff it is distributive.* ∎

As we remarked in the introduction, whenever the derivation spaces (set of co-initial abstract derivations, for any given term t) of a model form a PAD, this can be seen as an implicit confirmation of the "adequate degree of concurrency" of that model. Given a term t, we define now formally its *derivation space* as the subset of cells originating from it, representing the possible evolutions of the machine with that particular initial state.

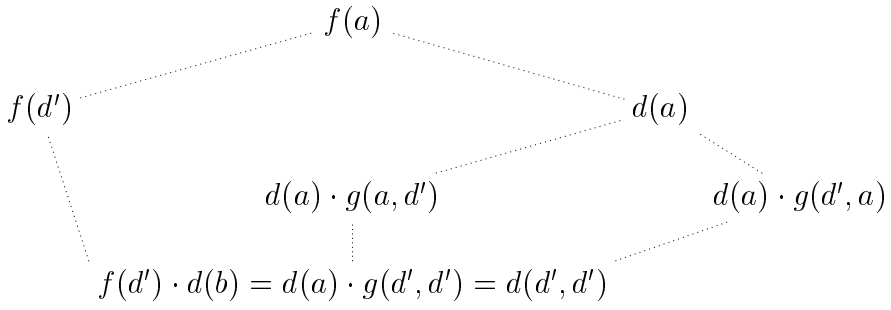
Definition 6.2 (Derivation Spaces) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS*, and $t \in T_\Sigma$. The* full derivation space $\mathcal{L}_2(t)$ *associated to t is the pre-order $\langle \mathcal{D}_2(t), \sqsubseteq_2 \rangle$, where $\mathcal{D}_2(t)$ is the class of abstract full proof terms in $\mathcal{T}_\mathcal{R}$ entailed by $\mathcal{R}$ with source t, and $\sqsubseteq_2$ is the pre-ordering relation defined as $\alpha \sqsubseteq_2 \beta$ iff there exists γ such that $\alpha \cdot \gamma \cong_{E_2} \beta$. Similarly, we define the* flat derivation space $\mathcal{L}_S(t) = \langle \mathcal{D}_S(t), \sqsubseteq_S \rangle$ *associated to t, by simply considering abstract flat proof terms in the above definition, and the set of axioms E_1.* ∎

The ordering relation is obviously well-defined. Actually, both $\mathcal{L}_S(t)$ and $\mathcal{L}_2(t)$ are *partial orders*. The result for the ordering induced by permutation equivalence was shown in [Bou85]; for disjoint equivalence we refer the reader to next section. Let us now restate the previous definitions in categorical terms (thanks to the correspondence between algebraic and categorical models).

Definition 6.3 (Categorical Derivation Spaces) *Let $R = \langle \Sigma, L, R \rangle$ be a* TRS*, and $t \in T_\Sigma$: then $t_\Sigma : \underline{0} \to \underline{1}$, and let us denote $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}[\underline{0}, \underline{1}]$ by $\mathbf{C}$. The full* categorical *derivation space $\mathcal{L}_{2C}(t)$ associated to t is the pre-order $\langle \mathcal{D}_{2C}(t), \sqsubseteq_{2C} \rangle$, where $\mathcal{D}_{2C}(t)$ is the class of objects of the comma-category $(t_\Sigma \downarrow \mathbf{C})$, and $\sqsubseteq_{2C}$ is the pre-ordering relation defined as $d_1 \sqsubseteq_{2C} d_2$ iff there exists γ in the class of arrows of the comma category, such that $\gamma : d_1 \to d_2$. Similarly, we define the flat* categorical *derivation space $\mathcal{L}_{SC}(t) = \langle \mathcal{D}_{SC}(t), \sqsubseteq_{SC} \rangle$ associated to t, by simply replacing $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}$ with $\underline{\mathbf{S\text{-}Th}(\mathcal{R})}$ in the above definition.* ∎

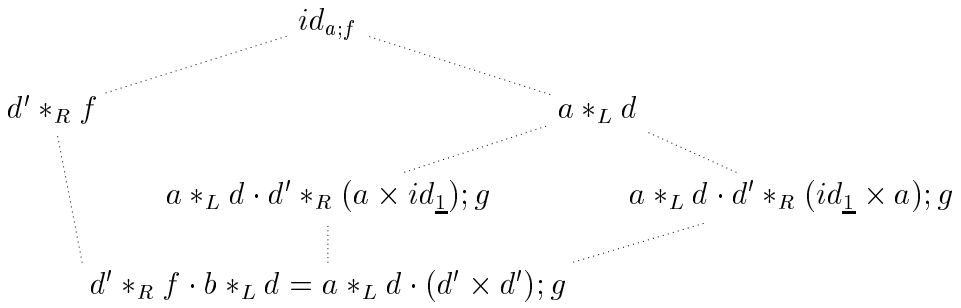
It turns out that, in general, the derivation spaces induced by 2-categorical models fail to satisfy the distributive property of PAD's, since the ordering in which rewrites are executed influences the number of basic steps which are to be performed. This is summarized by a naïve result of category theory: due to their structure, in a cartesian 2-category no notion of length for cells is definable [Mit72, Ste92].

As for a counterexample to distributivity, let us consider the TRS $\mathcal{W}$. The derivation space $\langle \mathcal{D}_2(f(a)), \sqsubseteq_2 \rangle$ has the following structure (where the ordering flows downwards)



Let $x = d(a) \cdot g(a, d')$, $y = d(a) \cdot g(d', a)$, and $z = f(d')$: then $z = (x \sqcup y) \sqcap z \neq (x \sqcap z) \sqcup (y \sqcap z) = f(a)$, hence $\mathcal{D}_2(f(a))$ is not distributive.

In categorical terms, if we consider the space of computations $\underline{\mathbf{2\text{-}Th}(\mathcal{W})}$, the term $f(a)$ and the associated arrow $f(a)_\Sigma = a; f$, the associated derivation space $\mathcal{D}_{2C}(f(a))$ has the following structure (where the ordering flows downwards):



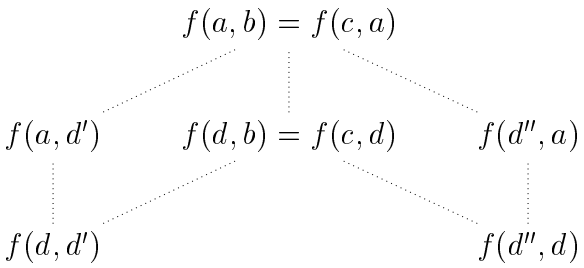
Again, let $x = a *_L d \cdot d' *_R (a \times \underline{id_1}); g$, $y = a *_L d \cdot d' *_R (\underline{id_1} \times a); g$, and $z = d' *_R f$: then $z = (x \sqcup y) \sqcap z \neq (x \sqcap z) \sqcup (y \sqcap z) = id_{a;f}$, hence $\mathcal{D}_{2C}(f(a))$ is not distributive. In categorical terms, this means that to allow together the cartesian structure and the interchange axiom does not permit to recover a PAD semantics: since the execution ordering influences in 2-categorical models the number of basic steps, it creates a causal link between different processors.

Let us remember that we consider terms as trees and that the underlying hypothesis about our implementation schema is that we have a distributed architecture, of the kind one-node/one-processor; each processor "knows" the rewriting rules of the system, the information on the actual node it is associated with, and the nodes it is linked with. Then, performing a reduction can be thought of as a two-steps procedure: first, the information on the nodes to be rewritten is updated, then the information of the connected nodes is eventually duplicated or discarded. If we consider as an example the TRS $\mathcal{W}$, the execution of the reduction $d(a) : f(a) \to g(a, a)$ consists of the updating of the node containing f, and the duplication of the node containing a. Now let us consider the proof term $d(d') = f(d') \cdot d(b) = d(a) \cdot g(d', d')$. Since the two computations are equated, we assume that our "actual" implementation can perform in parallel the rewriting of f and a. The reduction of f, however, implies that we have to duplicate the information we have on the node containing a. The parallel execution of the redexes would then imply a kind of *read-write* conflict, that could be resolved only if the underlying implementation schema does not rely on the tree-like representation of terms.

In the flat model the interchange axiom is dropped: the cartesian structure on cells is preserved, but all the causes of the possible conflicts are removed, since redexes that

have overlapping occurrences are not anymore considered independent.

Let us talk also about the reason why we avoided to take into account a possible axiomatization of the algebra of terms. The methodological reason is that we are assuming to deal with a fixed one-node/one-processor architecture, and that even a simple axiom like $f(x) = g(x)$ would spoil this view. Moreover, an axiom like $f(x) = a$ would destroy the possibility to recover a PAD semantics, even for the flat model, since for any rewrite α we would get $f(\alpha) = a$, equating again computations with different length. In fact, only the restriction to *linear* axioms could work, even if some identities between computations could hold that are difficult to relate to the actual behaviour of a possible implementation. As an example, let us consider the rewriting theory with signature $\{f, a, b, c, e\}$, axiom $f(x, b) = f(c, x)$, and rules $\{d : a \to e, d' : b \to e, d'' : c \to e\}$. The derivation space associated to $t_{(\Sigma,E)} = f(a, b) = f(c, a)$ has the following structure (where the ordering flows downwards)



The derivation space actually forms a PAD, but the ordering suggests a situation with only three different basic rewrites, difficult to relate to a feasible concurrent execution of reductions. In general, this problem arises where there is an overlapping of the left-hand side of a rule with an axiom: a discussion of a case study in the setting of permutation equivalence for λ-calculus can be found in [LM92], while for a study of the problems involved in turning the axioms into rewriting rules, see [Vir95]. However, if we consider flat models just for TRS's, we are able to prove the following result.

Theorem 6.1 (Flat Derivation Spaces and PAD's) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *Then for any $t \in T_\Sigma$, the pre-order $\mathcal{L}_S(t)$ is a coherent, finitary* PAD. □

Next section is devoted to the proof of the theorem. Actually (and it can be proved in an analogous way) also the following theorem holds.

Proposition 6.1 (Full Derivation Spaces and PAD's) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a linear* TRS. *For any $t \in T_\Sigma$, the pre-order $\mathcal{L}_2(t)$ is a coherent, finitary* PAD. □

6.2 Flat Models Form Prime Algebraic Domains

We start by giving the definition of *length* of a derivation, providing for each derivation the number of its basic steps.

Definition 6.4 (Length of a Derivation) *Let α be an abstract flat proof term. Its* length $l(t)$ *is defined inductively as:*

- $l(f(\alpha_1, \ldots, \alpha_n)) = \sum_i l(\alpha_i)$;
- $l(d(t_1, \ldots, t_n)) = 1$;
- $l(\alpha_1 \cdot \alpha_2) = l(\alpha_1) + l(\alpha_2)$. ∎

The definition is well-given, since all the flat axioms preserve length. In the case of a linear TRS, we could define a notion of length also for full proof terms, simply adding:

$$l(d(\alpha_1, \ldots, \alpha_n)) = 1 + \sum_i l(\alpha_i).$$

If the TRS is linear, the interchange axiom preserves length.

We state now some (easy) properties of length. For the sake of readability, in the following we will write $\alpha \equiv \beta$ for $\alpha \cong_{E_1} \beta$.

Proposition 6.2 (Basic Lengths) *Let α be an abstract flat proof term. Then the following properties hold:*

- $l(\alpha) = 0$ *iff* $\exists t \in T_\Sigma$, $\alpha \equiv t$;
- $l(\alpha) = 1$ *iff* $\exists \alpha' \in T_\mathcal{O}$, $\alpha \equiv \alpha'$. ∎

As a first step we prove that the pre-order $\mathcal{L}_S(t)$ is actually a partial order.

Proposition 6.3 (Disjointness implies Partial Order) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS, *and $t \in T_\Sigma$. Then the flat derivation space $\mathcal{L}_S(t) = \langle \mathcal{D}_2(t), \sqsubseteq_2 \rangle$ associated to t is a partial order.*

Proof It is sufficient to prove that the disjoint equivalence is *cancellative*; i.e., if $\alpha\cdot\beta \equiv \gamma\cdot\delta$ and $\alpha \equiv \gamma$, then $\beta \equiv \delta$. From this property immediately follows that $\sqsubseteq_S$ is a partial ordering relation.

Let $\alpha = \alpha_1 \cdot \beta_1 \equiv \alpha_2 \cdot \beta_2$, such that $\alpha_1 \equiv \alpha_2$. We proceed by induction on the length of α, and on the last axiom applied (i.e., on the length of the proof of the equivalence). We prove only the induction base for $l(\alpha) = 2$, since for a generic $l(\alpha)$ is analogous. About the length of the derivations, the only relevant case is $l(\alpha_1) = l(\alpha_2) = 1$. For the induction step on the last rule applied, the only relevant case is the Distributivity axiom, hence $\alpha_2 = \delta_1 \cdot t[w_1 \leftarrow \gamma_1]$, $\beta_2 = t[w_2 \leftarrow \gamma_2] \cdot \delta_2$ with w_1, w_2 disjoint occurrences and $l(\delta_1) = l(\delta_2) = 0$. In the proof of $\alpha_1 \cdot \beta_1 \equiv \delta_1 \cdot t[w_2 \leftarrow \gamma_2] \cdot t[w_1 \leftarrow \gamma_1] \cdot \delta_2$ the Disjoint rule on the same subterms must be used again: on the contrary we would have $\alpha_1 \equiv t[w_2 \leftarrow \gamma_2] \equiv \beta_2$ and $\alpha_2 \equiv t[w_1 \leftarrow \gamma_1] \equiv \beta_1$. And since the axioms preserve length, the result follows.

Now let $\alpha, \beta \in \mathcal{D}_2(t)$ such that $\alpha \sqsubseteq \beta$ and $\beta \sqsubseteq \alpha$. By definition there exist α', β' such that $\alpha \equiv \alpha \cdot \alpha' \cdot \beta'$. By the cancellative property $t \equiv \alpha' \cdot \beta'$, and the thesis follows. □

Now we start giving the basilar definitions to characterize the prime algebraic structure over $\mathcal{L}_S(t)$.

Fact 6.2 *Let $\alpha, \beta \in \mathcal{L}_S(t)$ be two derivations. They are* compatible, *and we write $\alpha \uparrow \beta$, if there exists a derivation $\gamma \in \mathcal{L}_S(t)$ such that $\alpha \sqsubseteq_S \gamma$ and $\beta \sqsubseteq_S \gamma$.* □

The following result characterizes compatible derivations of length 1.

Proposition 6.4 (Compatible Derivations of Length 1) *Let $\alpha, \beta \in \mathcal{L}_S(t)$ be two compatible derivations such that $l(\alpha) = l(\beta) = 1$. If $\alpha \not\equiv \beta$, then there exists a unique pair w_1, w_2 of disjoint occurrences of t and $d_1, d_2 \in ?$ such that $\alpha \equiv t[w_1 \leftarrow d_1(t_1, \ldots, t_n)]$, $\beta \equiv t[w_2 \leftarrow d_2(s_1, \ldots, s_m)]$, and $\alpha \sqcup \beta = t[w_1 \leftarrow d_1(t_1, \ldots, t_n), w_2 \leftarrow d_2(s_1, \ldots, s_m)]$.*

Proof Since α, β are compatible, then there exist α_i, β_i such that $\alpha = \alpha_1 \cdot t[w_1 \leftarrow f(s_1^{i\perp 1}, \alpha', s_{i+1}^n)]$, $\beta = \beta_1 \cdot t[w_2 \leftarrow g(s_1^{j\perp 1}, \beta', s_{j+1}^m)]$ (with $l(\alpha_1) = l(\beta_1) = 0$) and $\alpha \cdot \alpha_2 \equiv \beta \cdot \beta_2$. Now the characterization of α, β follows by induction on the last axiom applied, similarly to the proof of Proposition 6.3. The characterization of $\alpha \sqcup \beta$ is an immediate consequence. □

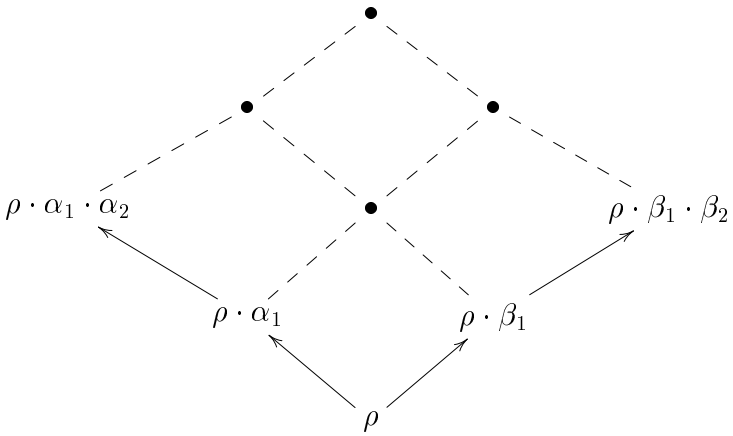
The following is an immediate consequence of the characterization of compatible derivations of length 1.

Proposition 6.5 (Compatible Set of Derivations of Length 1) *Let $\alpha, \beta, \gamma \in \mathcal{L}_S(t)$ be pairwise compatible derivations of length 1. Then the* LUB $\bigsqcup\{\alpha, \beta, \gamma\}$ *exists.* □

We can generalize the existence of the LUB to derivations of arbitrary length.

Proposition 6.6 (Compatible Derivations) *Let $\alpha, \beta \in \mathcal{L}_S(t)$ be two compatible derivations. Then they have* LUB γ*, with derivations σ_1, σ_2 such that $\alpha \cdot \sigma_1 \equiv \beta \cdot \sigma_2 \equiv \gamma$ and $l(\sigma_1) \leq l(\beta), l(\sigma_2) \leq l(\alpha)$.*

Proof Let ρ be the greatest common prefix of α, β, i.e., such that $\alpha \equiv \rho \cdot \alpha'$ and $\beta \equiv \rho \cdot \beta'$. We proceed by induction on $l = l(\rho) + l(\alpha') + l(\beta')$. The case $l = 2$ is obvious, due to Proposition 6.4. Now, let us assume $\alpha \equiv \rho \cdot \alpha_1 \cdot \alpha_2$, $\beta \equiv \rho \cdot \beta_1 \cdot \beta_2$ with $l(\alpha_2) = (\beta_2) = 1$; then, we have a situation denoted by the following diagram



where we applied the inductive hypothesis to the different computations, and from which we infer that α, β have a LUB. □

From the previous result the following proposition can be easily inferred.

Proposition 6.7 (Compatible Set of Derivations) *Let $X \subseteq \mathcal{L}_S(t)$ be a set of pairwise compatible derivations. Then the* LUB $\bigsqcup X$ *exists.* □

Proposition 6.8 (GLB of Derivations) *Let $X \subseteq \mathcal{L}_S(t)$ be a set of derivations: then the* GLB $\bigsqcap X$ *exists.* □

Fact 6.3 *Let $\alpha \in \mathcal{L}_S(t)$ be a derivation: it is* complete prime *if, whenever there are derivations $\beta, \gamma \in \mathcal{L}_S(t)$ such that $\alpha \sqsubseteq \beta \sqcup \gamma$, then either $\alpha \sqsubseteq \beta$ or $\alpha \sqsubseteq \gamma$.* □

We define now a *unique predecessor* property for derivations, that will be used as an alternative characterization of complete primeness.

Definition 6.5 (Unique Predecessor) *Let $\alpha \in \mathcal{L}_S(t)$ be a derivation. We say that it has a* unique predecessor *if, whenever there are derivations $\beta, \gamma \in \mathcal{L}_S(t)$ such that $l(\beta) = l(\gamma) = l(\alpha) - 1$, then $\beta \equiv \gamma$.* □

Proposition 6.9 (Uniqueness Implies Primeness) *Let $\alpha \in \mathcal{L}_S(t)$ be a derivation, such that it has a unique predecessor. Then it is a complete prime.* □

Since the converse trivially holds, then a derivation $\alpha \in \mathcal{L}_S(t)$ is a complete prime iff it has a unique predecessor. Thanks to this characterization, we can finally give the proof of our theorem.

Proof (of Theorem 6.1). Let us indicate the set $\{\beta \sqsubseteq \alpha \mid \beta \text{ complete prime}\}$ with P_α; we have to show that for all $\alpha \in \mathcal{L}_S(t)$ the identity $\alpha \equiv \bigsqcup P_\alpha$ holds. If α is complete prime, the thesis is obviously verified. On the contrary, we proceed by induction on the ordering relation. The case $\alpha = t$ is immediate. Now let us assume that $\overline{\alpha} = \bigsqcup P_\alpha$, with $\alpha \neq \overline{\alpha}$: by construction $\overline{\alpha} \sqsubset \alpha$, and by induction hypothesis $\overline{\alpha} \equiv \bigsqcup P_{\overline{\alpha}}$. Now we show that $l(\overline{\alpha}) = l(\alpha) - 1$. In fact, if for example $l(\overline{\alpha}) < l(\alpha) - 1$, then there would be $\underline{\alpha}$ such that $\overline{\alpha} \sqsubset \underline{\alpha} \sqsubset \alpha$, and by induction hypothesis $\underline{\alpha} \equiv \bigsqcup P_{\underline{\alpha}}$: hence, it would follow that

$$\overline{\alpha} \sqsubset \underline{\alpha} \equiv \bigsqcup P_{\underline{\alpha}} \sqsubseteq \bigsqcup P_\alpha \equiv \overline{\alpha}$$

that would imply $\overline{\alpha} \equiv \underline{\alpha}$, absurd. Since α is not a complete prime it has not a unique predecessor; then there exists $\tilde{\alpha}$ such that $\tilde{\alpha} \sqsubset \alpha$, $l(\tilde{\alpha}) = l(\alpha) - 1$ and $\tilde{\alpha} \not\equiv \overline{\alpha}$; moreover, by induction hypothesis $\tilde{\alpha} \equiv \bigsqcup P_{\tilde{\alpha}}$. From this we have that

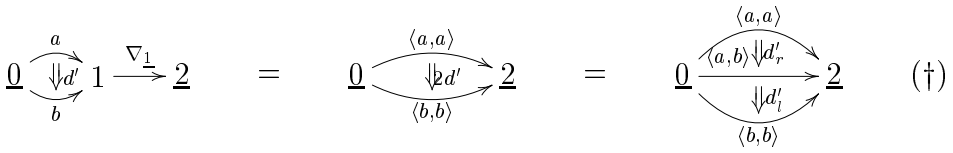
$$\alpha \equiv \overline{\alpha} \sqcup \tilde{\alpha} \equiv \bigsqcup P_{\overline{\alpha}} \sqcup \bigsqcup P_{\tilde{\alpha}} \sqsubseteq \bigsqcup P_{\overline{\alpha} \sqcup \tilde{\alpha}} \equiv \bigsqcup P_\alpha \equiv \overline{\alpha}$$

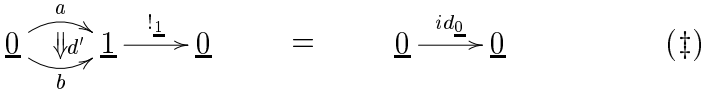
that would imply $\alpha \equiv \overline{\alpha}$, absurd. □

6.3 Cartesianity vs. Interchange

In this chapter we have analyzed the abstract models presented in Chapter 4 from a concurrent point of view, proving that the derivation spaces associated to the flat model form prime algebraic domains, that is, a well accepted model for expressing the concurrency of an abstract formalism. On the contrary, this is not true in general for the derivation spaces of the full model (based on cartesian structure plus interchange axiom).

However, we view the flat model as too narrow: dropping interchange means limiting the degree of concurrency expressible by the model in an unacceptable way. For instance, let us consider the TRS $\mathcal{V} = \{d(x) : f(x) \to g(x), d' : a \to b\}$. The computations corresponding to $d(a); d'$ and $d'; d(b) : f(a) \to g(b)$ are not equated in the sesqui-category $\underline{\mathbf{S\text{-}Th}(\mathcal{V})}$. On the other hand, the 2-categorical model fails to generate a PDA when non-linear rewriting rules are involved, since for such rules the execution order influences the number of basic steps to be performed. As an example, in the 2-category $\underline{\mathbf{2\text{-}Th}(\mathcal{W})}$ the following derivations with different length are equated

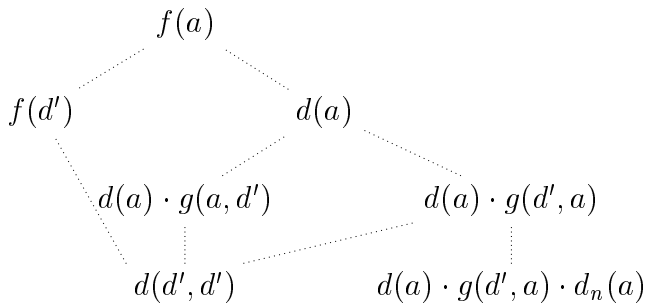




where $2d' = d' *_L \nabla_{\underline{1}} = \langle d', d' \rangle$, $d'_l = \langle d', id_b \rangle$ and $d'_r = \langle id_a, d' \rangle$.

The cartesian structure on cells allows for "implicit" *garbage collection* and *duplication* (as discussed in [Cor96]): these "housekeeping" operations are performed silently, in the sense that the abstract machine corresponding to the model cannot distinguish for instance between states where garbage has been already collected, and states where it has not. The situation is well-shown by (†) and (‡): it seems evident that models with interchange axiom *and* implicit behaviour on these operations cannot be the basis for a PAD semantics. The flat solution still keeps these operations silent, while dropping interchange forces the model to distinguish between derivations differing for the order of execution of nested rewrites.

This discussions is strictly related also to the statement we made in the introduction about the fact that"the [permutation] equivalence does not describe faithfully the behaviour of the reduction mechanism over graphs, whenever not-orthogonal rules are taken into account". Let us consider the TRS W, extended with the rule $\{d_n : g(b,x) \to c\}$. The derivation space associated to $f(a)$ would have the following structure (where the ordering flows downwards)



with a conflict between $d_n(a)$ and $g(b,d')$. From a graph reduction point of view, however, the conflict should not happen, since the occurrences of a in $g(a,a)$ should be unified, after performing $d(a)$. Again, this is due to the ambiguous nature of *coupling* in the cartesian setting, and its *reversibility*, characterized in the present case by the identity $a;\Delta_1 = \langle a,a\rangle$. In Chapter 9 we will investigate the case of preserving interchange and dealing with some kind of weak cartesianity. We will take into account *s-monoidal theories* (partly inspired by [Jac93], on the semantics of linear logic, and [Laf95], on equational reasoning), making housekeeping explicit in order to algebraically characterize *term graph rewriting*.

Although a detailed discussion about the actual implementability of term rewriting in a way reflecting the degree of concurrency of either models goes beyond the scope of the thesis, some further comments are in order. Even if the computations of the 2-categorical model do not form a PAD, it does not mean that the model cannot be *implemented* on a parallel computer, as by the way has been shown by Meseguer's work on the *Rewriting Rule Machine* (see [LMR94]). However, if a *concurrent* machine (one with loosely coupled processors) is chosen as target, the results of this paper show that the 2-category model cannot be implemented *directly*, i.e., by representing operations of processors with events of a PES having the same domain of computations. Of course the 2-categorical model can be mapped to a concrete, concurrent machine, but the compilation process should be designed carefully in order to minimize hidden, expensive synchronizations and sequentializations of processors, which are unavoidable according to our results.

Chapter 7

Dealing with Infinitary Rewriting

One of (maybe *the*) assumption of this thesis is that, despite their simplicity, rewriting theories can be considered a basic paradigm for computational devices: terms are states of an abstract machine, while rewriting rules are state-transforming functions; in this framework, computations simply are sequences of rewrites. Usually, however, TRS's deal with finite terms and finite computations; although the seminal work on continuous algebras by Goguen et al. dates back to the mid-Seventies [ADJ77], the extension of term rewriting to infinite terms is a subject raised to a certain interest only in recent years, mainly due to the use of graphs to model rewriting. In fact, in *term graph rewriting*, a finite, cyclic graph may represent an infinite term, and a single rewriting step can be equivalent to an infinite sequence of rewrites (see e.g. [Cor93, DKP91, KK+95]). The main difference among the various proposals presented in the literature concerns the description of such a sequence: i) as the limit of an infinite, sequential application of the rules ([DKP91, KK+94, KK+95], but also [AN80, Bou85]); ii) as the result of a simultaneous, parallel application of the rules to parts of the term [Cor93, CD96]. These two views really are alternative, since they behave differently with *collapsing rules* (such as $f(x) \to x$). Starting with the work of Boudol [Bou85] in the mid-Eighties, the sequential approach has received further more attention than the other: both these approaches, however, just extend the classical, set-theoretical approach to term rewriting.

In this chapter we introduce a new formalism for dealing with infinitary term rewriting, relying on rewriting logic. In the previous chapters we worked in the *finitary* setting, where sequents represent *finite* sequences of rewrites over *finite* terms. Here we consider proof terms as elements of a continuous algebra. Since the elements of a continuous algebra form a strict CPO, fully exploiting this structure over proof terms we are able to introduce *infinitary rules*: whenever there exists a suitable chain of sequents, then we add the

derivation corresponding to its supremum. Therefore, continuous algebras allow us to define a natural extension of rewriting logic we call *infinite rewriting logic*. The formalism has a nice and clean presentation and, even if straightforward, it is quite powerful: indeed, it consistently includes the *infinite parallel term rewriting* (IPTR) previously proposed in [Cor93], in the sense that for each derivation admissible in IPTR there exists a sequent entailed in IRL with the same source and target.

In the first section of the chapter we introduce IRL (originally presented in our paper [CG95]), proving in Section 2 a result analogous to Proposition 5.4 about its operational semantics, showing a one-to-one correspondence between derivations admissible in IPTR and (a class of) sequents entailed in IRL. In Section 3 we extend instead the functorial semantics of rewriting logic to the infinitary case. In Chapter 4 we showed how suitable models for a TRS $\mathcal{R}$ are given by the functor category $[\underline{\mathbf{2\text{-}Th}(\mathcal{R})} \to \underline{\mathbf{C}}]$ of chosen functors from the Lawvere 2-theory $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}$ to a given 2-category with chosen 2-products $\underline{\mathbf{C}}$. This notion of model has been proved to be adequate for finitary rewriting logic when considering the case $\underline{\mathbf{C}} = \underline{\mathbf{Cat}}$, where $\underline{\mathbf{Cat}}$ is the 2-category of categories, functors and natural transformations. In this chapter we show that this method lifts smoothly to IRL, when considering the sub-2-category $\underline{\mathbf{Cat}(\mathbf{Cpo})}_S$ of categories internal to **Cpo**, proving a soundness and completeness theorem for the infinitary case.

7.1 Infinite Parallel Rewriting Logic

In this section we present our extension of the rewriting logic approach to deal with infinite terms and infinitary rewrites. For the basic notion of ordered algebras, and the definition of IPTR, we refer to Chapter 2. Note that we are not changing the definition of TRS: we still assume that our rules are finite, i.e., that for a given $\langle \Sigma, L, R \rangle$, the elements of R are pairs of terms in T_Σ. We need however to extend the definition of proof terms and sequents, in order to take into account infinitary rewrites.

Definition 7.1 (Continuous Sequents) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS*, and $\Lambda = \cup_n \Lambda_n$ the signature satisfying $\Lambda_n = \{d \mid d(x_1, \ldots, x_n) : l(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n) \in R\}$ for each $n \in \mathbb{N}$. A* continuous *proof term α is a term of $CT_{(\Sigma \cup \Lambda \cup \{\cdot\})}$, where "$\cdot$" is a binary operator (we assume that no name-clashing between the various sets of operators occur). A continuous proof term α is* one-step *if it does not contain the operator "$\cdot$" (i.e., it is a term of the algebra $CT_{\mathcal{O}} = CT_{(\Sigma \cup \Lambda)}$); it is* many steps *if $\alpha = \alpha_1 \cdot \ldots \cdot \alpha_n$ with $1 \leq n < \omega$ and α_i is one-step for each $i \in \{1, \ldots, n\}$. A* continuous *sequent is a triple $\langle \alpha, t, s \rangle$ (usually*

written as $\alpha : t \to s$*) where* α *is a continuous proof term and* $t, s \in CT_\Sigma$*: it is* one-step (many steps) *if so is* α*.* ∎

Now we introduce a natural extension of rewriting logic which allows for the generation of sequents corresponding to infinite parallel derivations. Since each sequent is composed of three terms which belong (by definition) to continuous algebras, the "infinitary" sequents we are looking for are obtained by taking the LUB's of suitable chains.

Definition 7.2 (Infinite Parallel Rewriting Logic) *Let* $\mathcal{R} = \langle \Sigma, L, R \rangle$ *be a* TRS*. We say that* $\mathcal{R}$ entails *the infinite, parallel sequent* $\alpha : s \to t$ *if it can be obtained by a finite number of applications of the following rules of deduction:*

- (Reflexivity)

$$\frac{}{- : - \to -};$$

- (Elementary Instantiation), (Elementary Congruence) *and* (Transitivity) *as in Definition 5.3;*

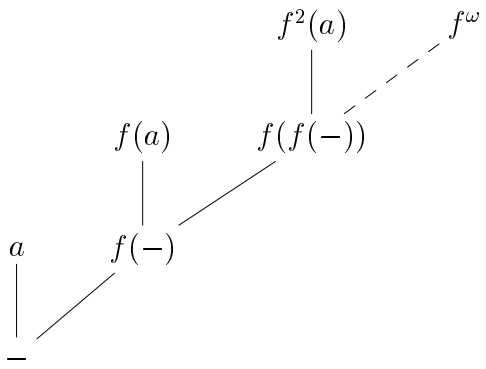
- (Infinite Parallel Rewriting)

$$\frac{\alpha_i : t_i \to s_i, \alpha_i \in CT_{\mathcal{O}}, \alpha_i \leq \alpha_{i+1} \text{ for } i \in \mathbb{N}}{\bigsqcup_i \alpha_i : \bigcup_i t_i \to \bigcup_i s_i}.$$

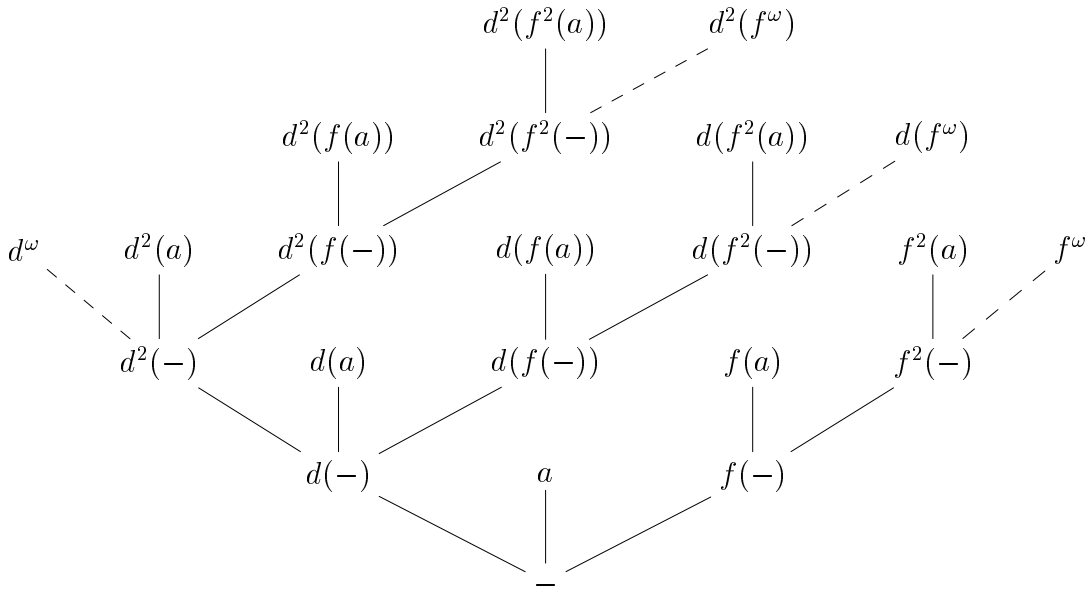
∎

According to Definition 7.1, the sequent $\bigsqcup_i \alpha_i : \bigcup_i t_i \to \bigcup_i s_i$ is also one-step. It is easy to check that the infinitary rule is well-defined: whenever the proof terms of a sequence of one-step sequents (entailed by a rewriting system) form a chain, then the source and target terms form a chain as well, and thus the LUB's exist. In fact, sequents themselves can be equipped with a continuous (partial) algebra structure: the bottom element is $\langle -, -, - \rangle$, and so on. We refer the reader to the analogous discussion in Chapter 4.

For example, let us consider the TRS $\langle \Sigma, L, R \rangle$ such that $\Sigma = \{a, f\}$ and R has the only rule $d(x) : f(x) \to x$. The continuous algebra CT_Σ has the following structure (where the partial ordering flows upwards):



The class of infinite, parallel one-step sequents entailed by the TRS has instead the following structure (again, the partial ordering flows upwards):



The sequent $d^\omega : f^\omega \to -$ describes the simultaneous reduction of the whole set of redexes contained in f^ω. Instead, each $d^n(f^\omega) : f^\omega \to f^\omega$ describes the reduction of the first n redexes of the term, thus the infinite term (as a whole) is unaffected by the rewriting. Note that the family $\{d^n(f^\omega) \mid n < \omega\}$ does not form a chain: in general, infinite sequences of reductions are not defined.

7.2 Consistency between Operational Semantics

In order to extend the correspondence between the set-theoretical and the algebraic presentation of parallel rewriting to the infinite case, we need a pair of technical lemmas. In the following, we will say that a sequent is *finitely entailed* if in its derivation the Infinite

Parallel Rewriting rule is not used[1]; or, equivalently, if it is a full sequent obtained extending the signature of the TRS with a constant $-$. The first lemma is the analogous of the compression lemma for infinitary term rewriting *a là* Huet-Lévy: it shows that each infinite parallel sequent can be obtained with a single application of the Infinite Parallel rule to an ω-chain of finite, one-step sequents.

Proposition 7.1 (Infinite Proof Terms are Finitely Generated) *Let $\mathcal{R}$ be a* TRS*, entailing the (possibly infinite) one-step sequent $\alpha : t \to s$. Then there exists an ω-chain $\{\alpha_i : s_i \to t_i\}_{i \in \mathbb{N}}$ of finitely entailed sequents, such that $\alpha = \bigsqcup_i \alpha_i$, $t = \bigcup_i t_i$ and $s = \bigcup_i s_i$.*

Proof The class of (infinite) one-step proof terms coincides with the class of terms of the algebra $CT_{\mathcal{O}}$. Then, it is enough to show that, for each element α of $FT_{\mathcal{O} \cup \{\bot\}}$, there exists a full sequent of the form $\alpha : t \to s$; this is easily showed by induction on the structure of proof terms. We already remarked that, whenever two finite sequents $\alpha_i : t_i \to s_i$ are entailed, then $\alpha_1 \leq \alpha_2$ implies $t_1 \leq t_2$ and $s_1 \leq s_2$ (and $\alpha_1 = \alpha_2$ implies $t_1 = t_2$ and $s_1 = s_2$). Combining these two facts, the result holds by a continuity argument, since each term in $CT_{\mathcal{O}}$ is equivalent to the LUB of a suitable ω-chain of elements of $FT_{\mathcal{O}}$. □

The following results shows that *i*) the function κ defined in Proposition 5.4 for left-linear TRS's (parametric over finite terms, and ranging over finite sets of redexes) actually preserves the ordering relation; and *ii*) an analogous property holds for ψ.

Proposition 7.2 (Redexes and the Ordering over Proof Terms) *Let t, $s \in FT_{\Sigma}$ be finite terms, and Φ, Φ' finite parallel redexes of t and s, respectively, such that:*

- *$t \leq s$ and $\Phi \subseteq \Phi'$;*
- *if $(w, d) \in \Phi' - \Phi$, then $t/w = -$.*

Then $\kappa_t(\Phi) \leq \kappa_s(\Phi')$.

Proof We proceed by induction on the structure of t and the number of redexes in Φ'. The base case $\Phi' = \emptyset$ is immediate. Note that, unless $t = -$, there can be no rule d such that $(\lambda, d) \in \Phi' - \Phi$.

- $t = -$. Immediate;

[1]Note that for each finitely entailed sequent, its components are finite, while the converse is not true, since ω-chains are not strictly increasing

- $t = f(t_1, \ldots, t_n)$. Since $t \leq s$, then $s = f(s_1, \ldots, s_n)$ with $t_i \leq s_i$ for $i = 1 \ldots n$. We distinguish between two cases:

  - If there is no rule d such that $(\lambda, d) \in \Phi$, then by definition
    $$\kappa_t(\Phi) = f(\kappa_{t_1}(\Phi/1), \ldots, \kappa_{t_1}(\Phi/n)),$$
    and analogously for $\kappa_s(\Phi')$. By construction $\Phi/i \subseteq \Phi'/i$ and, whenever $(w, d) \in (\Phi'/i)/(\Phi/i)$, $t_i/w = -$. Hence by induction $\kappa_{t_i}(\Phi/i) \leq \kappa_{s_i}(\Phi'/i)$, and the thesis follows.
  - If there is a rule d such that $(\lambda, d : l \to r) \in \Phi$: then by definition there exist substitutions $\sigma = \{x_1/u_1, \ldots, x_n/u_n\}$ and $\sigma' = \{x_1/v_1, \ldots, x_n/v_n\}$ such that $l/\sigma = t$, $l/\sigma' = s$. Then by the hypothesis $u_i \leq v_i$,
    $$\kappa_t(\Phi) = d(\kappa_{u_1}(\Phi/\mathcal{O}_{x_1}(l)), \ldots, \kappa_{u_n}(\Phi/\mathcal{O}_{x_n}(l))),$$
    and analogously for $\kappa_s(\Phi')$. By construction $\Phi/\mathcal{O}_{x_i}(l) \subseteq \Phi'/\mathcal{O}_{x_i}(l)$ and, whenever $(w, d) \in (\Phi'/\mathcal{O}_{x_i}(l))/(\Phi/\mathcal{O}_{x_i}(l))$, $u_i/w = -$. Hence by induction
    $$\kappa_{u_i}(\Phi/\mathcal{O}_{x_i}(l)) \leq \kappa_{v_i}(\Phi'/\mathcal{O}_{x_i}(l)),$$
    and the thesis follows. □

Proposition 7.3 (Proof Terms and the Inclusion of Redexes) *Let t, $s \in FT_\Sigma$ be finite terms, and α, α' finite one-step sequents of t and s, respectively, such that $\alpha \leq \alpha'$ (hence, $t \leq s$). Then $\psi(\alpha) \subseteq \psi(\alpha')$ and, if $(w, d) \in \psi(\alpha') - \psi(\alpha)$, then $t/w = -$.* □

Now we are ready to show the relationship between infinite sequents and infinite, parallel redexes. We restrict again to non-overlapping, left-linear rules only.

Proposition 7.4 (Many-Steps Sequents and Infinite Parallel Derivations) *Let $\mathcal{R}$ be an orthogonal* TRS *and Φ be a (possibly infinite) parallel redex.* (1) *If $t \to_\Phi s$, then there is a one-step proof term α_Φ such that $\mathcal{R}$ entails the sequent $\alpha_\Phi : t \to s$ (using the rules of Definition 7.2). Viceversa,* (2) *if $\mathcal{R}$ entails a one-step sequent $\alpha : t \to s$, then there is a (possibly infinite) parallel redex Φ_α such that $t \to_{\Phi_\alpha} s$. As a consequence, there is a parallel derivation from t to t' iff $\mathcal{R}$ entails a many-steps sequent $\alpha : t \to t'$.*

Proof We generalize the proof of Proposition 5.3.

1. Let Φ be an infinite, parallel redex and a chain $\{t_i\}_{i \in \mathbb{N}}$, defined according to Definition 2.24. Let $\kappa_{t_i}(\Phi_i) = \alpha_{\Phi_i} : t_i \to s_i$ be the one-step sequents associated to Φ_i: according to Proposition 7.2, $\kappa_{t_i}(\Phi_i) \leq \kappa_{t_j}(\Phi_j)$ for $i \leq j$. Then the sequent associated to Φ simply is $\alpha_\Phi = \bigsqcup_i \alpha_{\Phi_i} : \bigcup_i t_i \to \bigcup_i s_i$.

2. Since each one-step sequent is finitely generate, i.e., is the LUB of an appropriate ω-chain of finitely entailed one-step sequents (see Proposition 7.1), we can simply extend the function defined in the proof of Proposition 5.3 with the following case, without loss of generality

$$\psi(\bigsqcup_i \alpha_i) = \bigsqcup_i \psi(\alpha_i) \text{ for } \alpha_i \in FT_{\mathcal{O}\cup\{\bot\}}.$$

Thanks to Proposition 7.3, $\bigsqcup_{i<\omega} \psi(\alpha_i)$ is a parallel redex. Now, let us consider the ω-chain $\{t_i\}_{i\in\mathbb{N}}$: it satisfies the requirement of Definition 2.24, and the parallel redex (contained in $\psi(\bigsqcup_i \alpha_i)$) associated to each t_i is exactly $\psi(\alpha_i)$. Then $\psi(\bigsqcup_i \alpha_i)$ is well-defined. □

The following proposition states that we can lift the results about the one-to-one correspondence between operational models described in the previous chapter, to the infinite case.

Proposition 7.5 (Equivalence between Operational Models, IV) *Let $\mathcal{R}$ be the orthogonal* TRS $\langle \Sigma, L, R\rangle$. *Then for each $t \in T_\Sigma$ there is a one-to-one correspondence between the families of infinite parallel derivations entailed by $\mathcal{R}$ originating from t, and the families of infinite parallel sequents with source t.*

Proof It is a routine extension of the proof of Proposition 5.4. We need to take into account also infinite redexes, checking the infinitary cases, but restricting our attention, thanks to Proposition 7.1, to infinite proof terms obtained with only one application of the infinite parallel rule. □

Now we introduce the full version of infinite parallel rewriting logic, to which we provide a suitable categorical model in the next section.

Definition 7.3 (Full Infinite Rewriting Logic) *Let $\mathcal{R} = \langle \Sigma, L, R\rangle$ be a* TRS. *We say that $\mathcal{R}$* entails *the* infinite *sequent $\alpha : s \to t$ if it can be obtained by a finite number of applications of the finitary rules of deduction of infinite parallel rewriting logic (see Definition 7.2), where the Infinite Parallel Rewriting rule is substituted by the following one:*

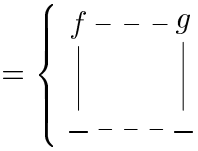
- (Infinite Rewriting)
$$\frac{\alpha_i : t_i \to s_i, \alpha_i \leq \alpha_{i+1} \text{ for } i \in \mathbb{N}}{\bigsqcup_i \alpha_i : \bigcup_i t_i \to \bigcup_i s_i}.$$

□

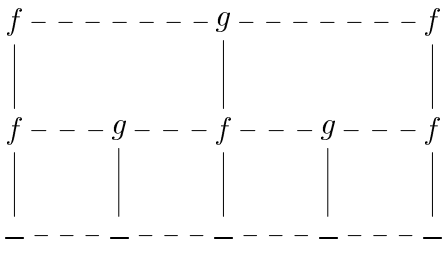
Note that the full version of infinitary rewriting logic seems to be stronger than its parallel counterpart. We suspect that there are TRS's $\mathcal{R}$ such that infinite sequents $\alpha : t \to s$ have no counterpart in a finite sequence $\alpha_1 \cdot \ldots \cdot \alpha_n : t \to s$ of infinite parallel sequents. It is still true that each sequent $\alpha : s \to t$ entailed by full infinite rewriting logic can be described by an ω-chain of finitely entailed, *many-steps* proof terms; but there are chains such that the number of occurrences of the "$\cdot$" operator inside each finite proof term is increasing, hence a result analogous to Proposition 5.8 is unlikely to hold.

A simple example can be given by considering a TRS with cycles, like $\mathcal{U} = \{\alpha(x) : f(x) \to g(x), \beta(x) : g(x) \to f(x)\}$. $\mathcal{U}$ entails the family $\gamma = \bigsqcup_i \gamma_i$, where the elements are inductively defined as $\gamma_1 = \alpha(-)\cdot\beta(-)$, $\gamma_2 = \alpha(\alpha(-)\cdot\beta(-))\cdot\beta(\alpha(-)\cdot\beta(-)) = \alpha(\gamma_1)\cdot\beta(\gamma_1)$ and $\gamma_i = \alpha(\gamma_{i\perp 1}) \cdot \beta(\gamma_{i\perp 1})$. It is easy to show that γ actually forms a ω-chain, and that each sequent describes a sequence of rewrites of increasing length. Graphically, we have the following:

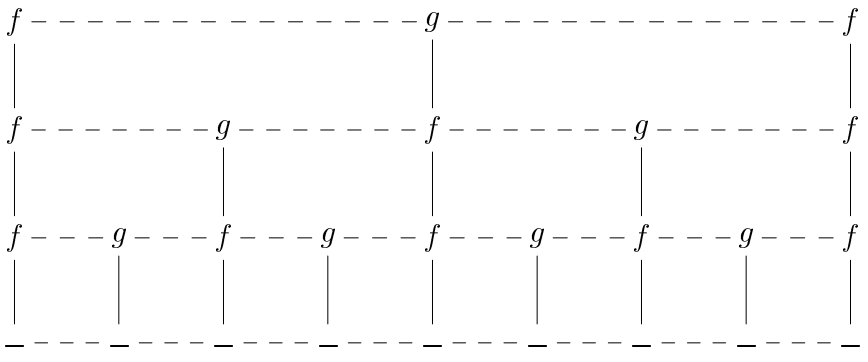
$\gamma_1 =$



$\gamma_2 =$



$\gamma_3 =$



Even if in this particular case the system entails also the infinite, parallel sequent $\delta = \alpha^\omega \cdot \beta^\omega = \bigsqcup_i\{\alpha^i(-)\} \cdot \bigsqcup_i\{\beta^i(-)\} = \bigsqcup_i\{\alpha^i(-) \cdot \beta^i(-)\}$, this is "intrinsically" different from γ, because δ acts exactly once (actually, twice) on each subterm to be reduced, while with γ the length of the reduction increases with the height of the elements in the chain.

7.3 Continuous Models of Term Rewriting

It is worth stressing here in which sense we regard a **Cat**-model for $\mathcal{R}$ as a **Set**-based model, because this will hint the correct structure for the universe 2-category of **Cpo**-based models. A 2-functor $\mathcal{M}$: $\underline{\mathbf{2\text{-}Th}(\mathcal{R})} \to \underline{\mathbf{Cat}}$ maps each object of $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}$ (say $\underline{n}$) to a category $\mathcal{M}(\underline{n})$; in this category, the objects provide an interpretation to the n-tuples of terms, while the arrows give an interpretation to the rewrites. Since $\mathcal{M}(\underline{n})$ is a (small) category, both its objects and its arrows form a set, by definition. Thus $\mathcal{M}$ is considered a **Set**-based model because terms and rewrites are interpreted in a set. By analogy, a **Cpo**-model for $\mathcal{R}$ would interpret (tuples of) terms in a small, strict CPO, and rewrites as continuous functions: in such CPO's we could find an interpretation also for infinite terms and for the sequents generated by the infinitary deduction rules. A model should therefore map each object of $\underline{\mathbf{2\text{-}Th}(\mathcal{R})}$ to a category having more structure, namely a strict CPO of objects and a strict CPO of arrows: i.e., a strict continuous category.

Definition 7.4 (Continuous $\mathcal{R}$-Models) *Let $\mathcal{R} = \langle \Sigma, L, R \rangle$ be a* TRS. *A* continuous $\mathcal{R}$-model $\mathcal{S}$ *is a strict continuous category* $\mathbf{S}$ *together with*

- *a Σ-algebraic structure: for each $f \in \Sigma_n$, a continuous functor $f_{\mathcal{S}}$: $\mathbf{S}^n \to \mathbf{S}$;*
- *for each $d : s \to t \in R$, a continuous natural transformation $\alpha_{\mathcal{S}} : s_{\mathcal{S}} \Rightarrow t_{\mathcal{S}}$.*

A continuous $\mathcal{R}$-homomorphism F: $\mathcal{S} \to \mathcal{S}'$ *is a strict continuous functor F: $\mathbf{S} \to \mathbf{S}'$ preserving the algebraic structure and the rewriting rules.* **C$\mathcal{R}$-Mod** *is the category of continuous $\mathcal{R}$-models and continuous $\mathcal{R}$-homomorphisms.* ∎

A $\mathcal{CR}$-system is then a triple $\langle \mathbf{S}, \rho_S, \phi_S \rangle$ where $\rho_S = \{f_S \mid f \in \Sigma\}$ is a family of continuous functors, and $\phi_S = \{d_S \mid d \in R\}$ is a family of continuous natural transformations. It is easy to show that **C$\mathcal{R}$-Mod** is reflective inside **$\mathcal{R}$-Mod**. Then, since left-adjoints preserve initiality, there exists a continuous $\mathcal{R}$-system $\mathcal{I}_{\mathcal{CR}}$ that is initial in **C$\mathcal{R}$-Mod**, and the following soundness and completeness result holds.

Proposition 7.6 (Soundness and Completeness of Continuous $\mathcal{R}$-Systems) *Let $\mathcal{R}$ be a rewriting theory: it entails a continuous sequent $\alpha : t \to s$ iff there exists a continuous natural transformation $\alpha_{\mathcal{I}_{\mathcal{CR}}} : t_{\mathcal{I}_{\mathcal{CR}}} \Rightarrow s_{\mathcal{I}_{\mathcal{CR}}}$ (and then, iff there exists a continuous natural transformation $\alpha_{\mathcal{S}} : t_{\mathcal{S}} \Rightarrow s_{\mathcal{S}}$ for each continuous $\mathcal{R}$-model S).*

Proof First we recall that, whenever we have $a_i \leq b_i$ objects in $\mathbf{S}$, then $\alpha(a_1, \ldots, a_n) \leq \alpha(b_1, \ldots, b_n)$ and $F(a_1, \ldots, a_n) \leq F(b_1, \ldots, b_n)$ for any continuous natural transformation α and for any continuous functor F; hence, also the classes of continuous natural transformations and continuous functors comes equipped with a strict CPO structure, that is defined pointwise. Moreover, each continuous natural transformation $\alpha : \bigsqcup_i F_i \Rightarrow \bigsqcup_i G_i$ can be obtained as a suitable ω-chain $\bigsqcup_i \alpha_i$, with $\alpha_i : F_i \Rightarrow G_i$ and where F_i, G_i are obtained by completion of suitable functors over $\mathcal{I}_{\mathcal{R}'}$ (and $\mathcal{R}'$ is obtained extending the signature of the equational theory associated to $\mathcal{R}$ with the constant $-$).

Now we proceed by induction, recalling that also each infinite sequent can actually be described as the LUB of an ω-chain $\bigsqcup_i \alpha_i$ of finitely entailed ones, relying on the existing correspondence between finite sequents and natural transformations between functors defined over $\mathcal{I}_{\mathcal{R}'}$ (see Proposition 4.3). In the case of the sequent $\alpha = \bigsqcup_i \alpha_i$, for example, each finite sequents α_i induces a natural transformation $(\alpha_i)_{\mathcal{R}'}$ in $\mathcal{I}_{\mathcal{R}'}$, that can be extended to a continuous one in $\mathcal{I}_{\mathcal{CR}}$, such that $\alpha_i(a_1, \ldots, a_n) \leq \alpha_{i+1}(a_1, \ldots, a_n)$ for all i. Then the ω-chain on proof terms induces a suitable continuous natural transformation $\bigsqcup_i (\alpha_i)_{\mathcal{I}_{\mathcal{CR}}}$, and the result follows. □

Now, let $\underline{\mathbf{Cat(Cpo)}}_S$ be the 2-category such that objects are strict continuous categories, arrows are continuous functors, and 2-cells are continuous natural transformations, and equipped with the intuitive, chosen products structure. Thus we can define the **Cpo**-based models of a term rewriting system as 2-functors from the Lawvere 2-theory of $\mathcal{R}$ to $\underline{\mathbf{Cat(Cpo)}}_S$.

Definition 7.5 (Continuous Models of Term Rewriting Systems) *Let $\mathcal{R}$ be a* TRS. *A* $\underline{\mathbf{Cat(Cpo)}}$-model *for $\mathcal{R}$ is a chosen cartesian 2-functor $\mathcal{M} : \underline{\mathbf{2\text{-}Th}(\mathcal{R})} \to \underline{\mathbf{Cat(Cpo)}}_S$, and a* model morphism *is a strict continuous 2-natural transformation between models. The category* $\underline{\mathbf{Cat(Cpo)}}$-$\mathbf{Mod}_{\mathcal{R}}$ *of* $\underline{\mathbf{Cat(Cpo)}}$*-models is given by a sub-category of the 2-functor category $[\underline{\mathbf{2-Th}(\mathcal{R})} \to \underline{\mathbf{Cat(Cpo)}}]$: its objects are* $\underline{\mathbf{Cat(Cpo)}}$*-models of $\mathcal{R}$, and its arrows are model morphisms.* □

Finally, we are able to state the main result of this section, precisely relating continuous $\mathcal{R}$-models and functorial models.

Proposition 7.7 (The Functor Category of Continuous $\mathcal{R}$-models) *Let $\mathcal{R}$ be a* TRS. *The category of continuous $\mathcal{R}$-models (see Definition 7.4) is equivalent to the category* $\underline{\mathbf{Cat(Cpo)}}$-$\mathbf{Mod}_{\mathcal{R}}$ *of* $\underline{\mathbf{Cat(Cpo)}}$*-models for $\mathcal{R}$.* □

Chapter 8

Typed Rewriting Logic

In the previous chapters we have tried to show the relevance and flexibility of the rewriting logic paradigm. We hope we have made clear why the use of a suitable algebra of proof terms (this way equipping the computations of a system with an algebraic structure) allows to recover information about the concurrent and spatial structure of a system. However, we always dealt with an *unconditional* version of the logic, where proof terms are just elements of a term algebra, and sequents are freely generated from a given signature. With the exception of the composition operator, there is no way to express any suitable restriction over the class of sequents entailed from a given set of deduction rules. This is a main limitation, if we want to take into account in our framework formalisms relying on the use of *side-effects* in determining the actual behaviour of a given system. In the classical setting, this aim has been pursued by enriching the structure of the algebra of terms (taking as an example into account order-sorted theories) or using *conditional rules*, where the application of a rule is subject to the satisfaction of a suitable equation. This is also the starting point of *conditional rewriting logic* [Mes92]; the logic is extended allowing rules with the format $d : l \to r \ \textit{if}\ u \to v$, where the conditions do not require equalities to hold, but the existence of suitable rewrites between terms. The resulting formalism is quite powerful, but to allow such a general format in the conditions makes the reduction mechanism difficult to be effectively implemented. In this chapter we introduce a new formalism, *typed rewriting logic*, that is able to express some restrictions about the class of sequents to which a given rule can be applied. Differently from the more traditional approach, however, we assume that the additional information is carried out by *typing* conditions *over* proof terms.

In Section 1 we introduce typed rewriting logic, while in Section 2 we propose a categorical semantics, similar to the one of classical rewriting logic described in Chapter

4, but relying on the use of double-categories. Finally, in Section 3 we will recast some well-known formalisms in our typed logic, in order to sustain the claim of generality and flexibility of our paradigm; while in Section 4 we present some preliminary work on the implementation of *strategies for rewriting* in typed rewriting logic.

8.1 Typed Rewriting Logic: Syntax

The assumption of typed rewriting logic is that sequents carry information about the type of the associated reduction. Then a typed sequent is a 4-tuple $\langle \alpha, t, a, s \rangle$, where $\alpha : t \to s$ is an untyped sequent, and a is the *type* of the sequent itself. Thus, rewrite rules are tuples $\langle d, a_1, \ldots a_n, t, a, s \rangle$, where t and s are respectively source and target, d is the label, $a_1, \ldots, a_n$ are the (input) types needed by the sequents to which we want to apply the rule d, and a is the (output) type of the resulting sequent. This enrichment is in line with the intuition that proof terms are the central objects of the paradigm and, from this point of view, it is "analogous" to the extensions of the logic adopted for infinitary and term graph rewriting.

Definition 8.1 (Typed Rewriting Theories) *Let X be a set of variables. A* typed rewriting theory (TRT) *$\mathcal{R}$ (over X) is a tuple $\langle (\Sigma, E), A, L, R \rangle$, where (Σ, E) is an equational signature, L is a set of labels, A is a set of* types *(with a distinguished element, ι), and R is a function $R : L \to T_\Sigma(X) \times A^* \times A \times T_\Sigma(X)$ such that for all $d \in L$, if $R(d) = \langle l, a_1 \ldots a_n, a, r \rangle$, then $var(r) \subseteq var(l) \subseteq X$, $\#var(l) = n$ and l is not a variable: we usually write $d : l \xrightarrow[a_1 \ldots a_n]{a} r$.* ∎

The type ι means "no information", and it implies that we are not imposing to the underlying sequents to satisfy any particular requirement.

Definition 8.2 (Proof Terms and Typed Sequents) *Let $\mathcal{R} = \langle (\Sigma, E), A, L, R \rangle$ be a* TRT. *Let $\Lambda = \cup_n \Lambda_n$ be the signature containing all the rules $d : l \to r \in R$ with the corresponding arity given by the number of variables in d; more precisely, for each n, $\Lambda_n = \{d \mid d(x_1, \ldots, x_n) : l(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n) \in R\}$. A* proof term *$\alpha$ is a term of the algebra $T_\mathcal{R} = T_{\Sigma \cup \Lambda \cup \{\cdot\}}$, where "$\cdot$" is a binary operator (we assume that there are no clashes of names between the various sets of operators). A* (typed) sequent *is a 4-tuple $\langle \alpha, t, a, s \rangle$ (usually written as $\alpha : t \xrightarrow{a} s$) where α is a proof term, $t, s \in T_\Sigma$ and $a \in A^*$.*
∎

Typed sequents then record not only the justification of a given rewrite, but provide also information on the "type" of the reduction step. As for the untyped case, we say that *t rewrites to s via* α (with an *effect* a) if the sequent $\alpha : t \xrightarrow{a} s$ can be obtained by finitely many applications of the following *rules of deduction.*

Definition 8.3 (Typed Rewriting Logic) *Let* $\mathcal{R} = \langle (\Sigma, E), A, L, R \rangle$ *be a* TRT. *We say that* $\mathcal{R}$ entails *the* typed sequent $\alpha : s \xrightarrow{a} t$ *if it can be obtained by a finite number of applications of the following rules of deduction:*

- (Instantiation)

$$\frac{d : l \xrightarrow[a_1 \ldots a_n]{a} r \in R, d \in \Lambda_n, \alpha_i : t_i \xrightarrow{a_i} s_i \text{ for } i = 1, \ldots, n}{d(\alpha_1, \ldots, \alpha_n) : l(t_1, \ldots, t_n) \xrightarrow{a} r(s_1, \ldots, s_n)};$$

- (Congruence)

$$\frac{f \in \Sigma_n, \alpha_i : t_i \xrightarrow{\iota} s_i \text{ for } i = 1, \ldots, n}{f(\alpha_1, \ldots, \alpha_n) : f(t_1, \ldots, t_n) \xrightarrow{\iota} f(s_1, \ldots, s_n)};$$

- (Transitivity)

$$\frac{\alpha : s \xrightarrow{\overline{a}} t, \ \beta : t \xrightarrow{\overline{b}} u}{\alpha \cdot \beta : s \xrightarrow{\overline{ab}} u}.$$

where $\overline{a}, \overline{b} \in A^+$ *are strings, and* $\overline{ab}$ *means string composition.* ∎

Congruence is defined only over sequents with type ι; this means that we can correctly contextualize only sequents whose type carries no additional information. Type ι (and the Congruence rule) provide a notion of *idleness* for transitions; as for untyped logic, a term can be rewritten to itself (with a type specifying such a behaviour), only if all the subterms are themselves idle. If there is no rule d such that its output type is ι, then only identity sequents can be provided with a context, and the Congruence rule can be substituted by

- *(Reflexivity)*

$$\frac{t \in T_\Sigma}{t : t \xrightarrow{\iota} t}.$$

Whenever we want to allow a more general kind of congruence, we need to introduce specific rules for each operator $f \in \Sigma_n$ and each family $\{\alpha_i : t_i \xrightarrow{a_i} s_i\}$ for which we ask $f(\alpha_1, \ldots, \alpha_n)$ to be defined.

Transitivity is the only deduction rule that builds *sequences* of types. Note that, since the input types for the rewrite rules are elements in A, then both congruence and instantiation can be applied only to one-step sequents. This is reflected in the algebraic semantics, that is analogous to the one defined in Chapter 4 for flat rewriting logic.

Definition 8.4 (Abstract Typed Sequents) *Let $\mathcal{R} = \langle (\Sigma, E), A, L, R \rangle$ be a typed rewriting theory. An* abstract typed sequent *entailed by $\mathcal{R}$ is an equivalence class of typed sequents entailed by $\mathcal{R}$ modulo the following set E_3 of axioms, which are intended to apply to the corresponding proof terms:*

- (Associativity)
$$\frac{\alpha, \beta, \gamma \in T_{\mathcal{R}}}{\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma};$$
- (Axiomatizing)
$$\frac{t(x_1, \ldots, x_n) = s(x_1, \ldots, x_n) \in E, \alpha_i : t_i \xrightarrow{\iota} s_i \text{ for } i = 1, \ldots, n}{t(\alpha_1, \ldots, \alpha_n) = s(\alpha_1, \ldots, \alpha_n)};$$
- (Distributivity)
$$\frac{f \in \Sigma_n, \alpha_i : t_i \xrightarrow{\iota} s_i, \beta_i : u_i \xrightarrow{\iota} v_i \text{ for } i = 1, \ldots, n}{f(\alpha_1 \cdot \beta_1, \ldots, \alpha_n \cdot \beta_n) = f(\alpha_1, \ldots, \alpha_n) \cdot f(\beta_1, \ldots, \beta_n)};$$
- (Identity)
$$\frac{\alpha : s \xrightarrow{\overline{a}} t}{s \cdot \alpha = \alpha = \alpha \cdot t}.$$

where we implicitly assumed that ι is the identity (empty sequence) over A^*: i.e., $\overline{a}\iota = \iota\overline{a} = \overline{a}$ for each $\overline{a} \in A^*$. □

8.2 Typed Rewriting Logic: Functorial Semantics

It is not possible to define a notion of $\mathcal{R}$-system for typed rewriting; its definition in the untyped case was due to the particular structure of 2-categories as enriched categories over **Cat**. Instead, it is easy to generalize the functorial approach; it is enough to describe a

suitable *double-category*, equivalent to the Lawvere 2-theory defined for untyped rewriting. First, note that for any TRT $\mathcal{R} = \langle(\Sigma, E), A, L, R\rangle$, we can define a *graph with monoidal structure* A_G (informally, a reflexive graph G with a graph morphism $\otimes : G \times G \to G$), freely generated from the set of actions A; nodes are the (underlined) natural numbers, $\iota : \underline{0} \to \underline{0}$ is the identity and $a : 1 \to 1 \in A_G$ for any $a \in A$.

Definition 8.5 (From Typed Rewriting Theories to D-Computads) *Let $\mathcal{R}$ be the* TRT $\langle(\Sigma, E), A, L, R\rangle$. *The associated d-computad* $\mathbf{Th}(\mathcal{R})$ *is the tuple* $\langle \mathbf{Th}(\mathcal{C}), A_G, S_R\rangle$, *where S_R is the set of d-cells such that*

$$\begin{array}{ccc} n & \xrightarrow{t_{(\Sigma,E)}} & 1 \\ {\scriptstyle \overline{a}}\downarrow & d_R & \downarrow{\scriptstyle a} \\ n & \xrightarrow[s_{(\Sigma,E)}]{} & 1 \end{array} \in S_R \qquad \textit{iff} \qquad d : t \xrightarrow[a_1 \ldots a_n]{a} s \in R$$

where $\overline{a} = a_1 \otimes \ldots \otimes a_n$. ∎

Again, the existence of a left-adjoint between the category of d-computads with finite products and **DFC-Cat** is pivotal in defining a suitable model for typed rewriting theories; a double-category is freely generated from a d-computad, such that its cells represent (abstract) sequents.

Definition 8.6 (Spaces of Typed Computations) *Let $\mathcal{R}$ be a* TRT, *and let* $\mathbf{Th}(\mathcal{R})$ *be its associated d-computad. Then the associated* Lawvere D-Theory $\underline{\mathbf{D\text{-}Th}(\mathcal{R})}$ *is the cartesian d-category $F_{df}(\mathbf{Th}(\mathcal{V}_d))$ with finite horizontal products.* ∎

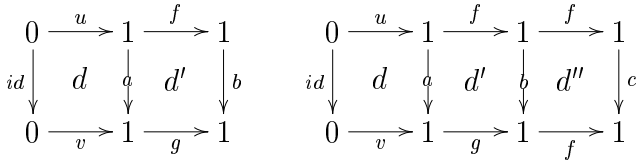
Let us consider the typed rewriting theory

$$\mathcal{V}_d = \{d : u \xrightarrow{a} v, d'(x) : f(x) \xrightarrow[a]{b} g(x), d''(x) : f(x) \xrightarrow[b]{c} f(x)\}.$$

The d-computad $\mathbf{Th}(\mathcal{V}_d)$ has the following set of cells (where, for the sake of readability, we did not put the subscript on generators):

$$\begin{array}{ccc} 0 & \xrightarrow{u} & 1 \\ {\scriptstyle id}\downarrow & d & \downarrow{\scriptstyle a} \\ 0 & \xrightarrow[v]{} & 1 \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{f} & 1 \\ {\scriptstyle a}\downarrow & d' & \downarrow{\scriptstyle b} \\ 1 & \xrightarrow[g]{} & 1 \end{array} \qquad \begin{array}{ccc} B1 & \xrightarrow{f} & 1 \\ {\scriptstyle b}\downarrow & d'' & \downarrow{\scriptstyle c} \\ 1 & \xrightarrow[f]{} & 1 \end{array}$$

The only (ground) typed sequents different from identities that are entailed by the theory are $d : u \xrightarrow{a} v$, $d'(d) : f(u) \xrightarrow{b} g(v)$ and $d''(d'(d)) : f(f(u)) \xrightarrow{f} (g(v))$; graphically,



The following proposition states the precise relationship, for a typed rewriting theory $\mathcal{R}$, between cells of the associated Lawvere d-theory, and families of abstract derivations.

Proposition 8.1 (Correspondence between Abstract Models, III) *Let $\mathcal{R}$ be a* TRT. *Then there exists a bijective function ϕ between the set of all abstract typed sequents entailed by $\mathcal{R}$ and the cells in $\underline{\mathbf{D\text{-}Th}(\mathcal{R})}[\underline{0}, \underline{1}]$, such that $\phi(\alpha) : s_{(\Sigma,E)} \to t_{(\Sigma,E)}$ iff $\alpha : s \to t$.* □

The proof is along the lines of the equivalent results for the untyped logic. In fact, a cell is obtained simply closing with respect to the monoidal operation and the vertical and horizontal composition. The axioms of double-categories with chosen horizontal products induce a one-to-one correspondence with the abstract sequents of the logic. Note however that type ι is fundamental in establishing the correspondence, since it offers a suitable counterpart to those cells that are identities of the vertical category.

In Chapter 4 we defined the class of models for the Lawvere 2-theory and s-theory associated to an untyped rewriting theory $\mathcal{R}$ as functors to a suitable universe. For typed rewriting logic, it is enough to consider double-categories; the constraints on types have their semantical counterpart in the additional structure that double-categories have with respect to 2-categories.

Definition 8.7 (Models of Lawvere D-Theories) *Let $\underline{\mathbf{C}}$ be a cartesian d-category with finite products. A* $\underline{\mathbf{C}}$-model *for the Lawvere d-theory associated to a* TRT *$\mathcal{R}$ is a chosen d-functor $\mathcal{M} : \underline{\mathbf{D-Th}(\mathcal{R})} \to \underline{\mathbf{C}}$, and a* model morphism *is a d-natural transformation between models. The category $\underline{\mathbf{C}}\text{-}\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{C}}$-models is the d-functor category $[\underline{\mathbf{D-Th}(\mathcal{R})} \to \underline{\mathbf{C}}]$; its objects are $\underline{\mathbf{C}}$-models of $\mathcal{R}$, and its arrows are model morphisms.* □

8.3 An Application in Concurrency Theory

It is quite common in *concurrency theory* to deal with formalisms relying on the use of *side-effects* in determining the actual behaviour of a given system. In this case, usually

a transition relation is not sufficient anymore to describe their evolution, and this makes quite difficult to recast them in the framework of (classical) term rewriting. In this section we provide the reader with two suitable examples, where typed theories are used to describe *process algebras*, a well-known example of specification languages, and a recently introduced generalization of this paradigm, *context systems*.

8.3.1 Process Description Algebras

Process (Description) Algebras [BK84, Hoa85, Mil89] offer a constructive way to describe *concurrent systems*, considered as structured entities (the *agents*) interacting by means of some synchronization mechanism. They define each system as a term of an algebra over a set of process constructors, building new systems from existing ones, on the assumption that algebraic operators represent basic features of a concurrent system. In the following we will present one of the better known example of process algebra, the *Calculus of Communicating Systems* (CCS), introduced by Milner in the early Eighties [Mil89], restricting ourselves, for the sake of exposition, to the case of *finite* CCS.

Definition 8.8 (The Calculus of Communicating Systems) *Let* Act *be a set of atomic* actions*, ranged over by μ, with a distinguished symbol τ and equipped with an involutive function $\overline{\mu}$ such that $\overline{\tau} = \tau$. Moreover, let $\alpha, \overline{\alpha}, \ldots$ range over* $\mathrm{Act}\backslash\{\tau\}$*. A* CCS process *is a term freely generated by the following syntax*

$$P ::= nil, \mu.P, P\backslash_\alpha, P[\Phi], P_1 + P_2, P_1||P_2$$

where $\Phi : Act \to Act$ is a relabeling *function, preserving involution. Usually, we let $P, Q, R...$ range over the set* Proc *of processes.* ∎

In the following, we will indicate as Σ_{CCS} the signature associated with CCS processes (for example, *nil* is a constant, μ a unary operator for each element in *Act*, and so on...). Given a process P, its dynamic behaviour can be described in a suitable transition system, along the lines of the so-called SOS approach [Plo81], where the transition relation is freely generated from a set of labeled inference rules.

Definition 8.9 (Operational Semantics of CCS) *The* CCS transition system *is the relation $T_{CCS} \subseteq Proc \times Act \times Proc$ inductively generated from the following set of axioms and deduction rules*

$$\frac{}{\mu.P \xrightarrow{\mu} P} \textit{ for } \mu \in \text{Act} \qquad \frac{P \xrightarrow{\mu} Q}{P[\Phi] \xrightarrow{\Phi(\mu)} Q[\Phi]} \textit{ for } \Phi \textit{ relabeling}$$

$$\frac{P \xrightarrow{\mu} Q}{P\backslash_\alpha \xrightarrow{\mu} Q\backslash_\alpha} \textit{ for } \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\frac{P \xrightarrow{\mu} Q}{P + R \xrightarrow{\mu} Q} \qquad \frac{P \xrightarrow{\mu} Q}{R + P \xrightarrow{\mu} Q}$$

$$\frac{P \xrightarrow{\mu} Q}{P||R \xrightarrow{\mu} Q||R} \qquad \frac{P \xrightarrow{\alpha} Q, P' \xrightarrow{\overline{\alpha}} Q'}{P||P' \xrightarrow{\tau} Q||Q'} \qquad \frac{P \xrightarrow{\mu} Q}{R||P \xrightarrow{\mu} R||Q}$$

where $P \xrightarrow{\mu} Q$ *means that* $\langle P, \mu, Q\rangle \in T_{CCS}$ ∎

A process P can execute an action μ and become Q if we can *inductively* construct a sequence of rule applications. As an example, to infer that from $P = (\alpha.nil + \beta.nil)||\overline{\alpha}.nil$ we can deduct $P \xrightarrow{\beta} Q = nil||\overline{\alpha}.nil$, three different rules must be taken into account. Moreover, a process P can be rewritten into another if there exists an appropriate chain of one-step reductions $P \xrightarrow{\mu_1} P_1 \ldots P_{n\perp 1} \xrightarrow{\mu_n} P_n$.

From an operational point of view, then, a process algebra can be faithfully described by a triple $\langle \Sigma, A, R\rangle$, where Σ is the signature of the algebra of agents, A is the set of actions and R is the set of deduction rules. There are two differences, with respect to term rewriting. First, deduction rules are *conditional*: you need information on the *label* of the underlying transitions, before applying a rule. Moreover, the rewriting steps are always performed *on top*; the order in which the rewrites are actually executed is important since, as an example, the correct operational behaviour of the agent $P = \alpha.\beta.nil$ is expressed saying that it executes first α and then β.

Both these features are easily described in the framework of typed rewriting logic. Actually, we will show how to describe a whole class of process algebras sharing a given format of the rules, the well-studied case of the *De Simone* format [DeS85].

Definition 8.10 (Deduction Rules in De Simone format) *A process algebra* $\mathcal{P}$ *is in the* De Simone format *if all its deduction rules have the form*

$$\frac{P_i \xrightarrow{\alpha_i} Q_i \textit{ for } i = 1 \ldots n}{C[P_1 \ldots P_n] \xrightarrow{\gamma} D[Q_1 \ldots Q_n]}$$

where C, D *are process contexts and each process variable appears* exactly once *in the premise and* at most once *in the conclusion of the rule.* ∎

We start noting that process algebras can be considered as a generalization of TRS's, where the constraints on labels have a syntactical counterpart in the typing over proof terms, and a semantical counterpart in the additional structure that double-categories have with respect to 2-categories.

Definition 8.11 (From Process Algebras to Typed Rewriting Theories) *Let $\mathcal{P}$ be the process algebra $\langle \Sigma, A, R \rangle$. Then the associated* TRT *is the tuple $\langle \Sigma, A, L, D_R \rangle$, where L is an arbitrary set of names, and D_R is the set of rules such that*

$$d(x_1 \dots x_n) : C \xrightarrow[a_1 \dots a_n]{a} D \in D_R \qquad \textit{iff} \qquad \frac{P_i \xrightarrow{a_i} Q_i \textit{ for } i = 1 \dots n}{C[P_1 \dots P_n] \xrightarrow{a} D[Q_1 \dots Q_n]} \in R.$$

□

The translation is even more clear if we consider the associated d-computad, where the distinction between process contexts and constraints is stressed by the different dimension they act in.

Definition 8.12 (From Process Algebras to D-Computads) *Let $\mathcal{P}$ be the process algebra $\langle \Sigma, A, R \rangle$. Then the associated d-computad $Th(\mathcal{P})$ is given by $\langle \mathbf{Th}(\Sigma), A_G, S_R \rangle$, where S_R is the set of d-cells such that*

$$\begin{array}{ccc} n & \xrightarrow{C_\Sigma} & m \\ {\scriptstyle \overline{a}_A}\downarrow & d & \downarrow{\scriptstyle a_A} \\ n & \xrightarrow[D_\Sigma]{} & m \end{array} \in S_R \qquad \text{iff} \qquad \frac{P_i \perp\!\!\!\rightarrow^{a_i} Q_i \text{ for } i=1\dots n}{C[P_1 \dots P_n] \perp\!\!\!\rightarrow^{a} D[Q_1 \dots Q_n]} \in R$$

where A_G is the graph with monoidal structure freely associated to the set of actions A, and $\overline{a}_A = (a_1)_A \otimes \dots \otimes (a_n)_A$. □

As an example, the following d-computad is just an instance of the previous, more general construction.

Definition 8.13 (The CCS D-Computad) *The d-computad $Th(\mathcal{P}_{CCS})$ associated to* CCS *is the tuple $\langle \mathbf{Th}(\Sigma_{CCS}), Act_G, S_{R_{CCS}} \rangle$, with the following structure:*

1. *Act_G is the graph freely associated to Act;*
2. *$\mathbf{Th}(\Sigma_{CCS})$ is the Lawvere theory associated to the signature Σ_{CCS};*

3. $S_{R_{CCS}}$ is the following set of cells

$$\begin{array}{ccc} 1 & \xrightarrow{\mu} & 1 \\ {\scriptstyle id}\downarrow & \mathrm{act}_\mu & \downarrow{\scriptstyle \mu} \\ 1 & \xrightarrow[id]{} & 1 \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{\Phi} & 1 \\ {\scriptstyle \mu}\downarrow & \mathrm{rel}_\Phi & \downarrow{\scriptstyle \Phi(\mu)} \\ 1 & \xrightarrow[\Phi]{} & 1 \end{array}$$

$$\begin{array}{ccc} 1 & \xrightarrow{\backslash\alpha} & 1 \\ {\scriptstyle \mu}\downarrow & \mathrm{res}_\alpha & \downarrow{\scriptstyle \mu} \\ 1 & \xrightarrow[\backslash\alpha]{} & 1 \end{array} \qquad for \quad \mu \notin \{\alpha, \overline{\alpha}\}$$

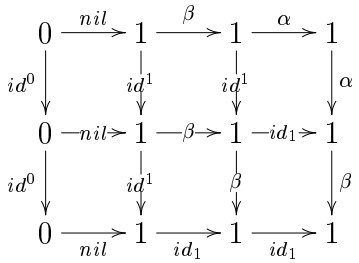
$$\begin{array}{ccc} 2 & \xrightarrow{+} & 1 \\ {\scriptstyle \mu\otimes id}\downarrow & \langle + & \downarrow{\scriptstyle \mu} \\ 2 & \xrightarrow[\pi_0]{} & 1 \end{array} \qquad \begin{array}{ccc} 2 & \xrightarrow{+} & 1 \\ {\scriptstyle id\otimes\mu}\downarrow & + \rangle & \downarrow{\scriptstyle \mu} \\ 2 & \xrightarrow[\pi_1]{} & 1 \end{array}$$

$$\begin{array}{ccc} 2 & \xrightarrow{\|} & 1 \\ {\scriptstyle \mu\otimes id}\downarrow & \xi & \downarrow{\scriptstyle \mu} \\ 2 & \xrightarrow[\|]{} & 1 \end{array} \qquad \begin{array}{ccc} 2 & \xrightarrow{\|} & 1 \\ {\scriptstyle id\otimes\mu}\downarrow & \rho & \downarrow{\scriptstyle \mu} \\ 2 & \xrightarrow[\|]{} & 1 \end{array} \qquad \begin{array}{ccc} 2 & \xrightarrow{\|} & 1 \\ {\scriptstyle \alpha\otimes\overline{\alpha}}\downarrow & \sigma & \downarrow{\scriptstyle \tau} \\ 2 & \xrightarrow[\|]{} & 1 \end{array}$$

(where we omitted the subscripts for the sake of readability). □

Note again that there is exactly one cell for each rule; some of them (such as act_μ and rel_Φ) are parametric with respect to the set of actions or to the set of relabeling functions, since the corresponding rules are so. The vertical arrow μ indicates that the 'underlying' process is actually running, outputting the action μ. It is the cell act_μ that prefixes an idle process with the action μ, and then starts the execution, consuming that same action. There are three cells dealing with the parallel operator: σ synchronizes two running processes, while ξ and ρ perform an asynchronous move, and to this end they take a running and an idle process.

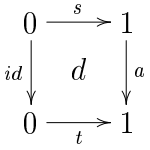
As an example of d-cells construction, let us consider the process $P = \alpha.\beta.nil$, executing sequentially first the action α, then the action β. It is not easy to model even such a simple agent in term rewriting, since the execution ordering, that is fundamental for expressing its behaviour correctly, is difficult to model in that setting. Instead, the computation is described by the double-cell



showing the importance of the vertical dimension in expressing the ordering constraints: the process can execute α only if the underlying process is actually idle.

Note that the axioms of double-categories impose an equivalence relation over d-cells (i.e., over computations), and then offer a description that, even if more concrete than the one given by the set–theoretical relation entailed by a given transition system, is still somewhat "abstract": there are many derivations that are identified, corresponding to "essentially" equivalent CCS derivations. There is an obvious adequacy result, holding for any generic PDA, stated by the following theorem.

Proposition 8.2 (Correspondence between Models) *Let $\mathcal{P} = \langle \Sigma, A, R \rangle$ be a* PDA. *Then the derivation $s \xrightarrow{a} t$ is entailed by $\mathcal{P}$ iff there exists a cell d in $F_{dc}(Th(\mathcal{P}))$ with*



(where we omitted the subscripts for the sake of readability). ∎

8.3.2 Context Systems

Context systems (briefly, CS's) have been introduced by Larsen and Xinxin [LX90] as a framework for developing a verification methodology for concurrent systems, providing a *modular* solution to the problem of characterizing what properties the subcomponents of a process must satisfy, in order to infer that the process itself satisfies a given specification. In this setting, "modular" means that the required properties should be decomposable into constraints to be verified by each subcomponent. Their proposal is based on an *operational semantics of contexts*. A context is a triple (C, n, m) (also, C_n^m) where C denotes the piece of the system already designed, taking an n-tuple of subcomponents and making available an m-tuple of parts, so that it can be considered as a *partial implementation*. A *transduction* between contexts is a 4-tuple $\langle C_n^m, (a_1, \ldots, a_n), (b_1, \ldots, b_m), D_n^m \rangle$; its meaning is that by consuming the actions $a_1, \ldots, a_n$, the context C_n^m produces the actions $b_1, \ldots, b_m$

for an external observer and it changes into D_n^m. Transduction subsumes the classical SOS approach, since a SOS rule can be considered in most cases as a transduction of the form $\langle C_n^1, (a_1, \ldots, a_n), b, D_n^1 \rangle$. Transductions can be composed both sequentially and in parallel, and it can be proved that, whenever each rule of a SOS specification can be described by a transduction, for each derivation possible in the associated transition system there is a corresponding one in the induced context system.

Definition 8.14 (Context Systems) *A* context system $\mathcal{CS}$ *is a triple* $\langle \mathcal{C}, A, R \rangle$ *where* $\mathcal{C}$ *is a set of contexts (i.e., of triples* (C, n, m) *for* $n, m \in \mathbb{N}$*: briefly,* C_n^m*); A is a set of actions with a distinguished one* ι*; and R is a set of (labeled) rules (i.e., a set of 4-tuples* $\langle C_n^m, (a_1, \ldots, a_n), (b_1, \ldots, b_m), D_n^m \rangle$ *for* $a_i, b_j \in A$*) satisfying* $\langle C, \overline{a}, \overline{\iota}, D \rangle \in R$ *iff* $C = D$ *and* $\overline{a} = \overline{\iota}$*.* ∎

We already said that, given a generic $\langle C, (a_1, \ldots, a_n), (b_1, \ldots, b_m), D \rangle$, its implicit meaning is that by consuming the actions $a_1, \ldots, a_n$, the context C makes available the actions $b_1, \ldots, b_m$ and evolves into D. When a component is ι, it means that the (internal) process is not involved in the transduction, i.e., it is *idle*. Two basic operators are defined on contexts: *composition* (given the contexts C_n^m and D_m^r, then also the combined context $(C \cdot D)_n^r$ is defined) and *product* (given the contexts C_n^m and D_s^r, then also the combined context $(C \times D)_{n+s}^{m+r}$ is defined).

Definition 8.15 (Operational Semantics of Context Systems) *Let* $\mathcal{CS}$ *be the context system* $\langle \mathcal{C}, A, R \rangle$*. Then the associated* transduction relation *is the relation on* $C_n^m \times A^n \times A^m \times D_n^m$ *for generic* n, m *and* $C_n^m, D_n^m \in \mathcal{C}$ *containing the tuples in R and closed with respect to the following rules:*

- $$\frac{t_1 : C_n^m \xrightarrow[\overline{a}]{\overline{b}} C_n'^m, \quad t_2 : D_m^r \xrightarrow[\overline{b}]{\overline{c}} D_m'^r}{t_1 \cdot t_2 : C \cdot D \xrightarrow[\overline{a}]{\overline{c}} C' \cdot D'};$$

- $$\frac{t_1 : C_n^m \xrightarrow[\overline{a}]{\overline{b}} C_n'^m, \quad t_2 : D_r^s \xrightarrow[\overline{c}]{\overline{d}} D_r'^s}{t_1 \times t_2 : C \times D \xrightarrow[\overline{ac}]{\overline{bd}} C' \times D'};$$

where $\overline{ab}$ *means string composition. A* transduction *is just an element* $C \xrightarrow[\overline{a}]{\overline{b}} D$ *of the entailed relation.* ∎

Context systems offer a framework where the classical, operational semantics of CCS can be easily recovered: each operator induces a context, and each rule of the SOS semantics has a corresponding rule in the associated context system.

Definition 8.16 (CCS Context System) *The* context system $\mathcal{CS}_{CCS}$ *associated to* CCS *is the tuple* $\langle \mathcal{C}_{CCS}, Act, R_{CCS} \rangle$*, with the following structure:*

1. *its set of contexts is*

$$(nil, 0, 1) \qquad (\mu, 1, 1) \qquad (\Phi, 1, 1)$$

$$(\backslash_\alpha, 1, 1) \qquad (+, 2, 1) \qquad (||, 2, 1)$$

$$(I, 1, 1) \qquad (\pi_0, 2, 1) \qquad (\pi_1, 2, 1)$$

   associating a basic context to each CCS *operator (the first six) and introducing some auxiliary ones (the latter three);*

2. *its set of actions is that of the basic actions of* CCS*;*

3. *its set of rules is*

$$act_\mu : \mu \xrightarrow[\iota]{\mu} I \qquad rel_\Phi : \Phi \xrightarrow[\mu]{\Phi(\mu)} \Phi$$

$$res_\alpha : \backslash_\alpha \xrightarrow[\mu]{\mu} \backslash_\alpha \qquad for\ \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\langle + : + \xrightarrow[(\mu,\iota)]{\mu} \pi_0 \qquad +\rangle : + \xrightarrow[(\iota,\mu)]{\mu} \pi_1$$

$$\sigma : || \xrightarrow[(\alpha,\overline{\alpha})]{\tau} || \qquad \xi : || \xrightarrow[(\mu,\iota)]{\mu} || \qquad \rho : || \xrightarrow[(\iota,\mu)]{\mu} ||$$

$$proj_0 : \pi_0 \xrightarrow[(\mu,\iota)]{\mu} \pi_0 \qquad proj_1 : \pi_1 \xrightarrow[(\iota,\mu)]{\mu} \pi_1 \qquad I : I \xrightarrow[\mu]{\mu} I$$

□

Auxiliary contexts are needed due to the syntactical nature of transduction entailment. The context systems approach to the dynamic behaviour forces the introduction of the rules $proj_0$, $proj_1$ and I, that make actions "floating" outside. Let us consider for instance the process $Q = \alpha.\beta.nil$, and the sequence $\beta.\alpha.nil \xrightarrow{\beta} \alpha.nil \xrightarrow{\alpha} nil$. From the associated

context $C_Q = nil \cdot \alpha \cdot \beta$, both transductions $nil \cdot \alpha \cdot act_\beta : C \xrightarrow[\lambda]{\beta} D = nil \cdot \alpha \cdot I$ and $nil \cdot act_\alpha \cdot I : D \xrightarrow[\lambda]{\alpha} nil \cdot I \cdot I$ are entailed (where λ is the empty string). Note that instead there is no need of auxiliary cells in the d-computad (hence, in the typed rewriting logic) associated to CCS, due to the cartesian structure of the horizontal category; action floating is "automatic", so to say, thanks to the structural axioms. For instance, many different contexts (such as $nil \cdot \alpha \cdot I \cdot \beta \cdot I$, $(nil \times nil) \cdot proj_1 \cdot \alpha \cdot \beta$ and $nil \cdot \alpha \cdot \beta$) are associated to the CCS process $P = \alpha.\beta.nil$, all of them essentially equivalent. In the Lawvere theory $\mathbf{Th}(\Sigma_{CCS})$, instead, all this contexts are again identified, and the auxiliary rules simply correspond to d-cells induced by the cartesian and categorical structure of the space of computations.

Now let us consider again the CCS process $P = (\alpha.nil + \beta.nil)||\overline{\alpha}.nil$. One of its associated contexts is $C_P = ((((nil \cdot \alpha) \times (nil \cdot \beta)) \cdot +) \times (nil \cdot \overline{\alpha})) \cdot ||$, while an entailed transduction is $t : C_P \xrightarrow[\lambda]{\beta} D$, where $t = ((((nil \cdot \alpha) \times (nil \cdot act_\beta)) \cdot +\rangle) \times (nil \cdot \overline{\alpha})) \cdot \xi$ and $D = ((((nil \cdot \alpha) \times (nil \cdot I)) \cdot \pi_1) \times (nil \cdot \overline{\alpha})) \cdot ||$. In general, for each process P there exist many associated contexts, each of them with the same operational behaviour, but there is obviously a *minimal* one we will indicate with $(C_P, 0, 1)$. The following result assures that transduction preserves the derivation relation of the transition system.

Proposition 8.3 (Transduction as Derivation) *Let P be a CCS process: then $P \xrightarrow{\alpha} Q$ in the CCS transition system iff a transduction $t : C_P \xrightarrow[\lambda]{\alpha} D$ is entailed by the CCS context system, where D is a context associated to process Q.* ∎

The proposition can be generalized to any process algebra that can be expressed in the so-called *basic De Simone format*: i.e., such that the rules have the form

$$\frac{P_i \xrightarrow{\alpha_i} Q_i \text{ for } i = 1 \ldots n}{C[P_1 \ldots P_n] \xrightarrow{\gamma} D[Q_1 \ldots Q_n]}$$

where C, D are process contexts and each process variable appears exactly once *both* in the premise and in the conclusion of the rule.

Definition 8.17 (From Context Systems to D-Computads) *Given a context system $\mathcal{CS} = \langle \mathcal{C}, A, R \rangle$, the associated d-computad $\mathbf{Th}(\mathcal{CS})$ is the tuple $\langle \mathbf{Th}(\mathcal{C}), A_G, S_R \rangle$, where S_R is the set of d-cells such that*

$$\begin{array}{ccc} n & \xrightarrow{C_{\mathcal{C}}} & m \\ {\scriptstyle \overline{a}_A}\downarrow & r_R & \downarrow{\scriptstyle \overline{b}_A} \\ n & \xrightarrow[D_{\mathcal{C}}]{} & m \end{array} \in S_R \qquad \textit{iff} \qquad r : C_n^m \xrightarrow[\overline{a}]{\overline{b}} D_n^m \in \mathcal{C}$$

□

Proposition 8.4 (Correspondence between Models) *Let $\mathcal{CS}$ be a context system. Then the transduction $C_n^m \xrightarrow[\overline{a}]{\overline{b}} D_n^m$ is entailed by $\mathcal{CS}$ iff there exists a cell d in $UF(\mathcal{CS})$, such that*

$$\begin{array}{ccc} n & \xrightarrow{C} & m \\ {\scriptstyle \overline{a}}\downarrow & d & \downarrow{\scriptstyle \overline{b}} \\ n & \xrightarrow[D]{} & m \end{array}$$

(where we omitted the subscripts for the sake of readability). □

8.4 Strategies for Rewriting

Given a TRS $\mathcal{R} = \langle \Sigma, L, R \rangle$, a *strategy* for $\mathcal{R}$ is a function $T_\Sigma \to T_\mathcal{R}$, associating to each term t a subset of the (full) concrete derivation space of t. The idea is to *prune* those parts of a derivation space that represent forbidden choices, i.e., to avoid all those derivations that a programmer does not want to be performed. In this sense, strategies define "allowed" proof-terms in the same way as types define allowed terms.

The easiest example is *head-rewriting*: our processor may perform a reduction only on the top-operator of a given term, that is, the possible redex has the format (λ, d) for a given rule d. Given a TRS $\mathcal{R} = \langle \Sigma, L, R \rangle$, to implement such a strategy means to provide a typed rewriting theory performing just the allowed sequents; in this case, it is simply defined as $\mathcal{R}_h = \langle \Sigma, A, L, R_h \rangle$ such that $A = \{\iota, \bullet\}$ and $d_h : t \xrightarrow[\overline{\iota}]{\bullet} s \in R_h$ iff $d : t \to s \in R$. If we consider the TRS $\mathcal{W}$ and the term $f(a)$, then only the typed sequent $d_R(a) : f(a) \xrightarrow{\bullet} g(a, a)$ is allowed, corresponding to the untyped sequent $d(a) : f(a) \to g(a, a)$, while the untyped sequent $f(d)$ has no corresponding typed one.

Now, more general strategies can be defined by appropriate congruence rules. If we consider a *strictly* left-most strategy, it means that *only* the possible reductions on the left-most operators (i.e., those occurring at a position $1 \ldots 1$) are executed. Given a TRS $\mathcal{R} = \langle \Sigma, L, R \rangle$, to implement such a strategy we define a typed rewriting theory $\mathcal{R}_l = \langle \Sigma, A, L, R_t \rangle$ such that $A = \{\iota, \bullet\}$, while for all $d : t \to s \in R$ and $f \in \sigma$ then $d_l :$

$t \xrightarrow[\bar{\imath}]{\bullet} s, f_l : f \xrightarrow[\bullet\bar{\imath}]{\bullet} f \in R_l$. If we consider again the TRS $\mathcal{W}$ and the term $f(a)$, then both derivations $f(a) \to g(a,a) \to g(b,a)$ and $f(a) \to f(b) \to g(b,b)$ can be executed, with corresponding typed sequents $d_l(a) \cdot g_l(d', a) : f(a) \xrightarrow{\bullet\bullet} g(b,a)$ and $f_l(d'_l) \cdot d_l(b) : f(a) \xrightarrow{\bullet\bullet} g(b,b)$. Note however that the untyped sequent $d(d') : f(a) \to g(b)$, corresponding to the parallel execution of two strictly left-most reductions, can not be executed. Such a strategy could be called strictly *parallel* left-most, and for each TRS $\langle \Sigma, L, R \rangle$ a suitable typed rewriting theory is obtained simply adding to the one for strictly left-most the rule $d_p : t \xrightarrow[\bullet\bar{\imath}]{\bullet} s \in R_p$ for all $d : t \to s \in R$.

Typed rewriting seems to be the most expressive way of describing strategies, able to recover all the usual notions, either position-based, like leftmost or innermost, or operator based, like in OBJ [GK+87]. It may also provide a simple and clear semantics to strategies, which have always been considered as an implementation device and never studied for themselves.

The problem is that, in general, a strategy needs to make *negative assumptions* on the derivation spaces of a term, in order to correctly define the "pruning" function. For example, the usual left-most strategy executes the reduction occurring at the far left *among all those* that can actually be executed, and this means that it does not necessarily reduce the left-most operator. In the TRS $\mathcal{W}$, both head-rewriting and strictly left-most prune the whole derivation space of $g(b,a)$, while the left-most selects the only reduction $g(b,d') : g(b,a) \to g(b,b)$, since no reduction can take place in the left-most subterms.

A possible solution to this problem lies in computing the complement of the sets of "patterns" that can match a given set of rules, thus expressing the non applicability of a set of rules only with positive conditions. This is known to be possible for left-linear rules, but the general case should be investigated. From a general point of view, however, difficulties arise due to the simple kind of constraints we used in the typed version of the logic. We aimed at providing a formalism *as similar as possible* to the unconditional version of rewriting logic, in order to get a "nice" and "clean" algebraic (categorical) formulation. Our hope is that we will be able to suitable extend the notion of "type", in order to make typed rewriting logic an useful tool for implementing the reduction process when more general constraints are taken into account (see e.g. [KKR90]).

8.4.1 Recovering ELAN

ELAN [Vit94, KKV95] is a high-level language based on rewriting in a many-sorted algebra, possibly modulo axioms of associativity and/or commutativity. It has some unique

features like possibilities to describe maps of logics, and what interests us more directly, an original notion of strategies, defined by the following language :

S	:=	R	apply *at the head* a rule whose label is in R
	|	$S_1; S_2$	S_1 followed by S_2
	|	`dont-care`$(S_1, ..., S_n)$	don't care choice with priority
	|	`dont-know`$(S_1, ..., S_n)$	don't know not determistic choice (backtrack)
	|	`iterate`(S)	apply S any number of times
	|	`repeat`(S)	apply S as much as possible

Strategies are applied only at the top of a term. In order to handle subterms, congruence must be made explicit in the side conditions of rules using the `where` construct, as for instance

`[r]` $f(x) \longrightarrow g(y)$ `where` y `:=` $(S)x$

whose intuitive meaning is that whenever the rule `r` is applied, apply recursively the strategy S to the argument x.

A strategy can either *fail* or be *successful*. If at some point no rule can be applied, evaluation is restarted from the previous `dont-care` or `dont-know` choice point. In case of success, a result is output, and evaluation restarts from the previous `dont-know` choice point.

Since congruence must always be explicited, a powerful preprocessor allows to write schemes of rules like congruence for any operator :

Strategy $S =$ `dont-care(cong)`
For each symbol f with arity n,

`[cong]` $f(x_1, ..., x_n) \longrightarrow f(y_1, ..., y_n)$ `where` y_1 `:=` $(S)x_1$
...
y_n `:=` $(S)x_n$

Then, for instance, a *parallel outer-most* strategy for a set R of rules would be described just using a single expression:

S `:= dont-care(`R`, cong);`

There is an obvious problem of non termination, because congruence can always be applied. The ad-hoc solution, though not very satisfying aesthetically, is to add a rule rewriting the top of a term with a side condition :

$$x \longrightarrow y \quad \texttt{where} \quad y := (S)x \quad \texttt{if } y \neq x$$

Coming back to typed proof terms, only part of the full power of ELAN can be recovered using typed rewriting logic. The problem is, as usual, linked with the possibility of negative premises, that are presented in the implementation of the `dont-care` and `repeat` strategy operators. If we assume not to deal with such a strategy, however, it is easy to see that we can get typed rules where labels on the arrows correspond to properties to be verified, like

$$f(x) \quad \stackrel{S_1}{\underset{S_2}{\longrightarrow}} \quad g(y)$$

for

$$S_1 := r$$

$$[r] \quad f(x) \longrightarrow g(y) \quad \texttt{where} \quad y := (S_2)\ \mathrm{x}$$

We are currently trying to recover a precise correspondence between typed rewriting theories and ELAN, looking for suitable extensions of typed rewriting logic that could still be equipped both with an algebraic and a categorical semantics.

Chapter 9

Further ork: On Term *Graph* Rewriting

We can say that our thesis has pursued a two-fold aim. First, we reviewed the classical approach to term rewriting and, via rewriting logic, we studied the algebraic properties of the set-theoretical semantics. Along this line, we investigated some categorical models, in order to get some suggestions about the fine structure of a possible concurrent implementation. At the same time, we remarked the relevance of term rewriting as a basic computational paradigm and, to this purpose, we introduced (usually via suitable extensions of rewriting logic) formalisms to deal with infinite computations and side-effects. In our current work we try to further stretch these intuitions, providing an algebraic characterization also for *term graph rewriting*. The two equivalences on derivations we studied in the thesis rely on different assumptions about the actual implementation of the reduction process. In the case of permutation equivalence, we assume to have a complex data structure describing graphs, and a sequential machine; with disjoint equivalence we assume instead to have a distributed system, and a one-node/one-processor architecture, where terms are described as trees.

The aim of this chapter is to suggest a way to unify the two different views, providing a suitable algebra able to describe *term graphs*. Starting from there, we are working on a recasting of the original notion of term graphs rewriting. Term graphs are defined in [BE+87] as *rooted* DAG*'s*, i.e., DAG's with a distinguished node, which are not really related with our ranked DAG's, as defined in Section 9.2. We intend to rephrase such definitions in a way that is essentially equivalent, but more consistent with our formal framework. If our claim about the algebras of term graphs holds, we will be able to easily recast in a 2-categorical setting also term graph rewriting. We will then provide an operational description of rewriting that is free of the problems of duplication and garbaging

we mentioned in Section 6.3, and whose derivation spaces should form prime algebraic domains; i.e., providing a concurrent semantics where also "garbaged reductions" are taken into account (in the vein of [Cor93], and differently from [KK+93]).

9.1 S-Monoidal Theories

Given a signature Σ, the relevant property of the algebraic theory $\mathbf{Th}(\Sigma)$ is that arrows from m to n are in one-to-one correspondence with n-tuple of terms of the free Σ-algebra with at most m variables. Each arrow t_Σ: $\underline{n} \to \underline{1}$ identifies a Σ-term t with variables among x_1,...,x_n; an arrow $\underline{n} \to \underline{m}$ is a m-tuple of Σ-terms with n variables, and arrow composition is term substitution. The theory can be regarded as an alternative presentation of a signature. Indeed, the additional structure it contains (besides the operators of the signature) is generated in a completely free way, so, in a sense, it does not add "information" to the original signature. On this section we introduce *s-monoidal theories*: they represent an original generalization of Lawvere theories (potentially more general than the one proposed in [Pfe74], from which we borrowed the name and some intuitions), and they have benefited from a careful look at [Laf95, Jac93].

Definition 9.1 (S-Monoidal Theories) *Given a signature Σ, the associated* s-monoidal theory *is the s-monoidal category* $\mathbf{S\text{-}Th}(\boldsymbol{\Sigma})$*, freely generated from the graph with pairing G_Σ such that*

- *its objects are underlined natural numbers: $\underline{0}$ is the identity element and pairing is defined as $\underline{n} \otimes \underline{m} = \underline{n+m}$;*
- *for every operator $f \in \Sigma_n$, there is a basic arrow $f_\Sigma : \underline{n} \to \underline{1}$.* ∎

Note that, since we have a very particular structure on objects, we can also define the arrows as those generated by the following inference rules:

$$(\textit{generators})\ \frac{f \in \Sigma_n}{f_\Sigma : \underline{n} \to \underline{1}}$$

$$(\textit{composition})\ \frac{s : \underline{n} \to \underline{m}, t : \underline{m} \to \underline{k}}{s;t : \underline{n} \to \underline{k}} \qquad (\textit{sum})\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n+n'} \to \underline{m+m'}}$$

$$(\textit{identities})\ \frac{}{id_{\underline{1}} : \underline{1} \to \underline{1}} \qquad (\textit{permutation})\ \frac{}{\rho_{\underline{1},\underline{1}} : \underline{2} \to \underline{2}}$$

$$(\textit{duplicator})\ \frac{}{\nabla_{\underline{1}} : \underline{1} \to \underline{2}} \qquad (\textit{discharger})\ \frac{}{!_{\underline{1}} : \underline{1} \to \underline{0}}$$

and satisfying the intuitive axioms. Even if the additional operators have been defined only for the basic object 1, they can be inductively derived for all the objects starting from these basic arrows, interpreting in a constructive way the coherence axioms. For example, the duplicator $\nabla_{\underline{n}}$ for each object $\underline{n}$ can be defined by means of $\nabla_{\underline{n\perp 1}}$ and $\rho_{\underline{n\perp 1},\underline{1}}$ as $(\nabla_{n\perp 1} \otimes \nabla_1); (id_{n\perp 1} \otimes \rho_{\underline{n\perp 1},\underline{1}} \otimes id_1)$.

Whenever we require the transformation $\nabla, !$ to be natural, we get the algebraic theory associated to the signature Σ. The fact that the arrows from $\underline{n}$ to $\underline{1}$ are in one-to-one correspondence with Σ-terms whose variables are among x_1,...,x_n *requires* that both ∇ and ! are natural. As an example, let us consider a constant c: as a generator, the corresponding arrow is $c_\Sigma : \underline{0} \to \underline{1}$, while, when considering c as an element of $T_\Sigma(x_1, x_2)$, then the associated arrow is $!_{\underline{2}}; c_\Sigma : \underline{2} \to \underline{1}$; moreover, these arrows are unique, since for each $s : \underline{n} \to \underline{m}$, we have $s; !_{\underline{m}} = !_{\underline{n}}$.

The main result we aim at is the proof that an analogous property holds for s-monoidal theories with respect to term graph algebras: i.e., that the arrows of the hom-set $[\underline{n}, \underline{m}]$ are in a one-to-one correspondence with term graphs with a specified m-tuple of accessible nodes and a specified n-tuple of variables. In the next section we report our preliminary results on this topic.

9.2 Some Results on Term Graphs

We open the section recalling the basic definitions regarding term graphs.

Definition 9.2 (Directed Acyclic Graphs) *Let Σ be a signature, i.e., a ranked set of operator symbols, and let* arity *be the function returning the arity of an operator symbol, i.e.,* $\text{arity}(f) = n$ *iff* $f \in \Sigma_n$. *A* labeled graph d (over Σ) *is a triple* $d = \langle N, l, s \rangle$, *where N is a set of* nodes, $l : N \to \Sigma$ *is a partial function called the* labeling function, $s : N \to N^*$ *is a partial function called the* successor function, *and such that the following conditions are satisfied:*

- $dom(l) = dom(s)$, *i.e., labeling and successor functions are defined on the same subset of N; a node $n \in N$ is called* empty *if $n \notin dom(l)$.*
- *for each node $n \in dom(l)$, $arity(l(n)) = length(s(n))$, i.e., each non-empty node has as many successor nodes as the arity of its label.*

*If $s(n) = \langle n_1, \ldots, n_k \rangle$, we say that n_i is the i-*th successor *of n and denote it by $s(n)_i$. A labeled graph is* discrete *if all its nodes are empty. A* path *in d is a sequence*

$$\langle n_0, i_0, n_1, i_1, \ldots, n_{m\perp 1}, i_{m\perp 1}, n_m \rangle$$

where $m \geq 0$, $n_0, \ldots, n_m \in N$, $i_0, \ldots, i_{m\perp 1} \in \mathbb{N}$ *(the natural numbers), and* n_k *is the* $i_{k\perp 1}$*-th successor of* $i_{k\perp 1}$ *for* $k \in \{1, \ldots, m\}$. *The* length *of this path is* m*; if* $m = 0$*, the path is* empty. *A* cycle *is a path like above where* $n_0 = n_m$.

A directed acyclic graph (over Σ)*, shortly* DAG*, is a labeled graph having no non-empty cycles. If* $n \in N$ *is a node of a* DAG $d = \langle N, l, s \rangle$*, then by* $d|n$ *we denote the sub-*DAG *of* d *rooted at* n*, defined in the obvious way. For a* DAG d *we shall often denote its components by* $N(d)$*,* l_d *and* s_d*, respectively.* □

In order to provide a suitable abstraction of the too concrete notion of DAG, we define the notion of morphism between DAG's.

Definition 9.3 (DAG Morphisms and the Category $\mathbf{Dag_\Sigma}$) *Let* d*,* d' *be* DAG*'s. A* (DAG) morphism $f : d \to d'$ *is a function* $f : N(d) \to N(d')$ *that preserves labeling and successors, i.e., such that for each non-empty node* $n \in N(d)$*,* $l_{d'}(f(n)) = l_d(n)$*, and* $s_{d'}(f(n))_i = f(s_d(n)_i)$ *for each* $i \in \{1, \ldots, arity(l_d(n))\}$.

Directed acyclic graphs over Σ *and* DAG *morphisms clearly form a category that will be denoted* $\mathbf{Dag_\Sigma}$. □

Now we introduce an (original) generalization of the notion of DAG, that is suited to describe graphs with multiple roots. Then, we will define a *term graph* as an equivalence class over such a generalization. In the following, for each $i \in \mathbb{N}$ we shall denote by $\underline{i}$ the set $\underline{i} = \{1, \ldots, i\}$ (thus $\underline{0} = \emptyset$).

Definition 9.4 (Ranked DAG's and Term Graphs) *An* (i, j)-ranked DAG *(or also, a* DAG of rank (i, j)*) is a triple* $g = \langle r, d, v \rangle$*, where* d *is a* DAG *with exactly* j *empty nodes,* $r : \underline{i} \to N(d)$ *is a function called the* root mapping*, and* $v : \underline{j} \to N(d)$ *is a bijection between* $\underline{j}$ *and the empty nodes of* d*, called the* variable mapping*. Node* $r(k)$ *is called the* k-th root *of* d*, and* $v(k)$ *is called the* k-th variable *of* d*, for each admissible* i.

Two (i, j)*-ranked* DAG*'s* $g = \langle r, d, v \rangle$ *and* $g' = \langle r', d', v' \rangle$ *are* isomorphic *if there exists a* ranked DAG isomorphism $\phi : g \to g'$*, i.e., a dag isomorphism* $\phi : d \to d'$ *such that* $\phi \circ r = r'$ *and* $\phi \circ v = v'$. *A* (i, j)-ranked term graph G *(or* with rank (i, j)*) is an isomorphism class of* (i, j)*-ranked* DAG*'s. We shall often write* G^i_j *to recall that* G *has rank* (i, j). □

We introduce now two operations on ranked term graphs. The *composition* of two term graphs is obtained by gluing the variables of the first one with the roots of the second one, and it is defined only if their number is equal. The *union* instead is always defined, and it is a sort of disjoint union where roots and variables are suitably renumbered.

Definition 9.5 (Composition of Ranked Term Graphs) *Let $G^i_j = [\langle r, d, v\rangle]$, $G'^j_k = [\langle r', d', v'\rangle]$ be two ranked term graphs. Their* composition *is the ranked term graph $H^i_k = G'^j_k \circ G^i_j$ defined as $H^i_k = [\langle in_d \circ r, d'', in_{d'} \circ v'\rangle]$, where $d'', in_d : d \to d''$ and $in_{d'} : d' \to d''$ are obtained as follows. Assuming that $d = \langle N(d), l_d, s_d\rangle$ and $d' = \langle N(d'), l_{d'}, s_{d'}\rangle$, we have $d'' = \langle (N(d) \uplus N(d'))/_\approx, l'', s''\rangle$, where $\uplus$ denotes disjoint union, $\approx$ is the least equivalence relation such that $v(i) \approx r'(i)$ for $i \in \underline{j}$, and l'' and s'' are determined by l_d and s_d, respectively, for all $\approx$-equivalence classes containing only nodes of d, and by $l_{d'}$ and $s_{d'}$, respectively, for all other classes. Furthermore, the injections $in_d : d \to d''$ and $in_{d'} : d' \to d''$ map each node to its $\approx$-equivalence class.* ∎

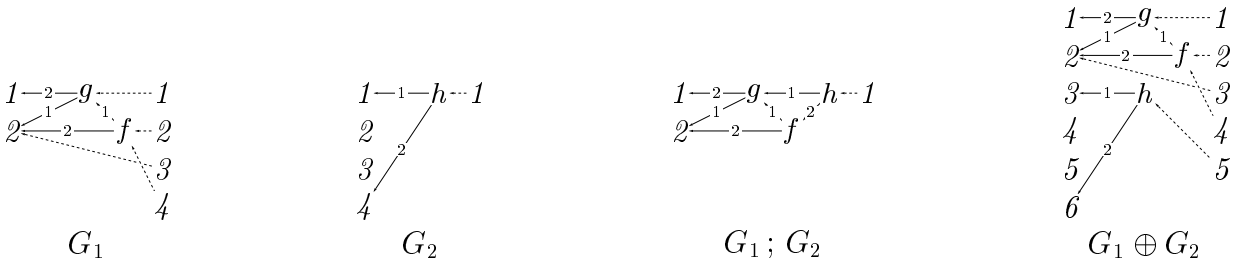
It is worth stressing that such composition can be characterized elegantly as a pushout, in the sense that $\langle d'', in_d, in_{d'}\rangle$ is a pushout of $\langle v : \underline{j} \to d, r' : \underline{j} \to d'\rangle$ in $\mathbf{Dag}_\Sigma$ (set $\underline{j}$ is regarded as a discrete dag). Actually, the pushout of two arrows in category $\mathbf{Dag}_\Sigma$ does not always exist. Necessary and sufficient conditions are given in [CR93] for the category of *jungles*, that can be regarded as a hypergraph variant of DAG's; indeed the categories of jungles and DAG's over the same signature are shown to be equivalent in [CM+91], and therefore the conditions presented in [CR93] also apply, *mutatis mutandis*, to $\mathbf{Dag}_\Sigma$. In particular, an easy consequence of the results in [CR93] is that the pushout does exist in the case we are interested in, since morphism $v : \underline{j} \to d$ is injective and has only empty nodes in the codomain.

Definition 9.6 (Union of Ranked Term Graphs) *Let us consider $G^i_j = [\langle r, d, v\rangle]$ and $G'^k_l = [\langle r', d', v'\rangle]$ ranked term graphs. Their* union *or* parallel composition *is the term graph of rank $(i+k, j+l)$ $G^i_j \oplus G'^k_l = [\langle r'', d \uplus d', v''\rangle]$, where $r'' : \underline{i+k} \to d \uplus d'$ and $v'' : \underline{j+l} \to d \uplus d'$ are defined as*

- $r''(x) = \begin{cases} r(x) & \text{if } x \in \underline{i} \\ r'(x-i) & \text{if } x \in \{i+1, \ldots, i+k\}. \end{cases}$
- $v''(x) = \begin{cases} v(x) & \text{if } x \in \underline{j} \\ v'(x-i) & \text{if } x \in \{j+1, \ldots, j+l\}. \end{cases}$ ∎

It is easy to check that composition and union of ranked term graphs are associative.

Example 9.1 *Let us consider the following four term graphs.*

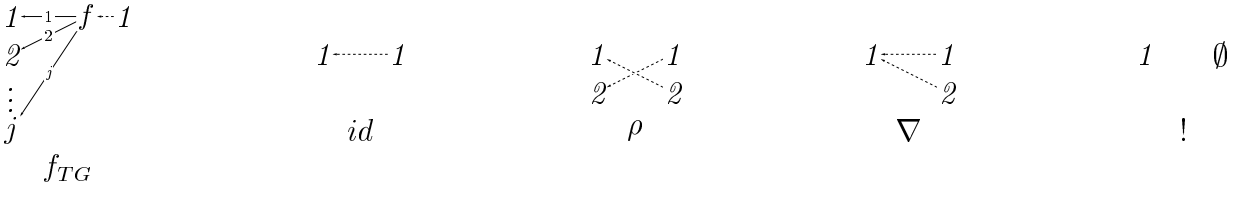


Empty nodes are represented by the natural numbers corresponding to their position in the list of variables, and are depicted as a vertical sequence on the left; non-empty nodes are represented by their label, from where the edges pointing to the successors leave; the list of numbers on the right represent pointers to the roots, and a dashed arrow from j to a node indicates that it is the j-th root. For example, the first term graph G_1 has rank $(4, 2)$, four nodes (two empty, 1 and 2, and two non-empty, f and g), the successors of g are the variables 2 and 1 (in this order), the successors of f are g and 2, and the four roots are g, f, 2, and f.

These graphical conventions make easy the operation of composition, that can be performed by matching the roots of the first graph with the variables of the second one, and then by eliminating them. For example, term graph $G_1 ; G_2$ is the composition of G_1 and of G_2 of rank $(1, 4)$. The last term graph is $G_1 \oplus G_2$, the union of G_1 and G_2, of rank $(5, 6)$.

In the rest of this section we will show that every term graph can be constructed, using composition and union, from a small set of atomic term graphs.

Definition 9.7 (Atomic Term Graphs) *An atomic term graph is a term graph in $\{f_{TG} \mid f \in \Sigma\} \cup \{id, \rho, \nabla, !\}$, which are depicted in the following*

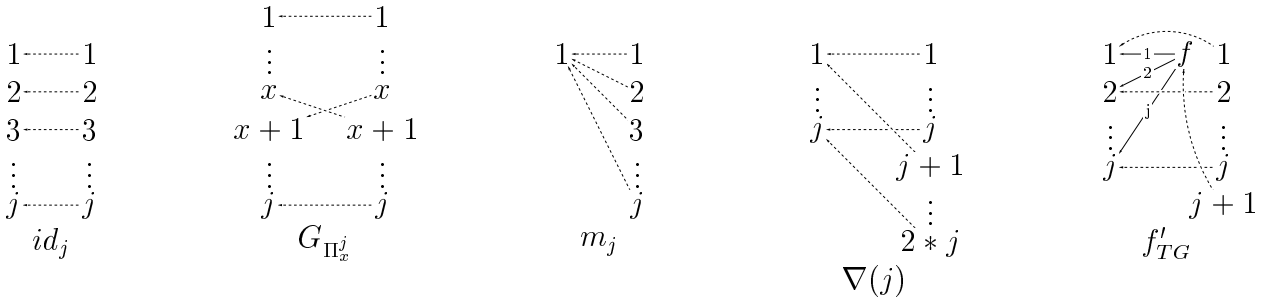


Since every node of an atomic term graph is a root or a variable, such term graphs can be formally defined as follows (using, a bit improperly, r, v, s, and l for the root, variable, successor, and labeling functions, respectively):

- *For each $f \in \Sigma$ with $arity(f) = j$, f_{TG} has rank $(1, j)$, with $l(r(1)) = f$, and $s(r(1))_x = v(x)$ for each $x \in \underline{j}$.*
- *The term graph id has rank $(1, 1)$, with $r(1) = v(1)$.*
- *The term graph ρ has rank $(2, 2)$, with $r(1) = v(2)$ and $r(2) = v(1)$.*
- *The term graph ∇ has rank $(2, 1)$, with $r(1) = r(2) = v(1)$.*
- *The term graph ! has rank $(0, 1)$, one empty node, and no roots.* ∎

Theorem 9.1 (Decomposition of Term Graphs) *Every term graph can be obtained as the value of an expression containing only atomic term graphs as constants and composition and union as operators.*

Proof We first need to define some auxiliary term graphs using atomic ones, composition and union. They are shown in the following



These arrows are defined formally as follows:

[***Identities***] Term graph id_j is defined as $id_j = id \oplus \ldots \oplus id$ (j times).

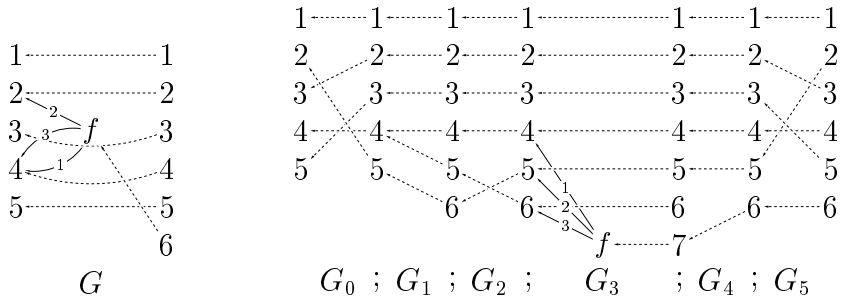
[***Permutations***] For each permutation Π over $\underline{j}$ (i.e., a bijective mapping $\Pi : \underline{j} \to \underline{j}$), G_Π is the discrete term graph of rank (j, j) such that for all $x \in \underline{j}$, $v(x) = r(\Pi(x))$. Since every permutation can be obtained by a finite sequence of exchanges of pairs of adjacent elements, every such term graph can be obtained as the composition of more elementary graphs like $G_{\Pi_x^j}$. In turn, we have $G_{\Pi_x^j} = id_{x\perp 1} \oplus \rho \oplus id_{j\perp x\perp 1}$.

[***Multiplicators***] For each $j \in \mathbb{N}$, term graph m_j has rank $(1, j)$. It is defined recursively as $m_0 = !, m_1 = id, m_2 = \nabla$, and $m_j = \nabla\,;\,(id \oplus m_{j\perp 1})$ if $j > 2$.

[***Duplicators***] Term graph $\nabla(j)$ for $j \in \mathbb{N}^+$ has rank $(2{*}j, j)$ and it is defined recursively as $\nabla(1) = \nabla$, and $\nabla(j) = (\nabla \oplus \nabla(j-1))\,;\,(id \oplus G_{\Pi'} \oplus id_{j\perp 1})$ if $j > 1$, where Π' is the permutation on $\underline{j}$ which maps 1 to j and all other numbers to their predecessors.

[***Multi-rooted atoms***] Term graph f'_{TG} (for $f \in \Sigma$ and $arity(f) = j$) is similar to the atomic graph f_{TG}, but all its nodes are roots. It has rank $(j+1, j)$, and is defined as $f'_{TG} = \nabla(j)\,;\,(id_j \oplus f_{TG})$.

[***Elementary term graphs***] A term graph is *elementary* is it has rank $(j{+}1, j)$ for some $j \in \mathbb{N}$, has only one non-empty node which is the $j+1$-th root, and for each $x \in \underline{j}$, $r(x) = v(x)$. Every elementary term graph can be obtained as the composition of six term graphs which are easily expressed in terms of the above defined auxiliary graphs¿ Let us consider the following case



showing the elementary graph G and its decomposition. Informally, G_3 has the form $id_k \oplus f'_{TG}$, G_2 implements a permutation that puts close to each other the successor nodes of f which are shared in G, G_1 is made of ∇'s and identities and glues together such nodes, G_4 is made of !'s and identities and "forgets" all nodes that all glued together by G_1, and G_0 and G_5 are suitable permutation graphs.

We prove now the statement by induction on the number of non-empty nodes of the term graph. Suppose first that G is a discrete term graph of rank (i, j), and let $g^i_j = \langle r, d, v\rangle$ be a ranked DAG in G. Clearly d has exactly j nodes. For each $x \in \underline{j}$, let $a_x = \#\{y \in \underline{i} \mid r(y) = v(x)\}$ be the number of times the x-th variable of g appears in the

list of roots. Then we have that $G = (m_{a_1} \oplus \ldots \oplus m_{a_j})\,;\, G_\Pi$ for a suitable permutation Π on $\underline{i}$.

Now, let G be a term graph of rank (i, j) with at least one non-empty node, and let $\underline{n}$ be a non empty node such that all its successors are empty: Such a node must exist by acyclicity. Let G' be the term graph of rank $(i, j+1)$ obtained from G by making the node $\underline{n}$ empty, and adding it as the $j+1$-th variable. Furthermore, let G'' be the elementary term graph of rank $(j+1, j)$ which is the subgraph of G containing all its variables and node $\underline{n}$. It is easy to see that $G = G''\,;\, G'$. Then since elementary term graphs can be constructed from atomic ones, the statement follows by induction hypothesis over G', since it has one non-empty node less than G. ∎

The previous results implies the existence of a full functor $\mathbf{S\text{-}Th}(\mathbf{\Sigma}) \to \mathbf{Dag}_\Sigma$. In our current work we are trying to prove the existence also a suitable faithful functor between the two categories, in order to sustain our claim about the algebraicization of term graphs.

Bibliography

[ADJ77] J.A. Goguen, J.W. Tatcher, E.G. Wagner, J.R. Wright, *Initial Algebra Semantics and Continuous Algebras*, Journal of the ACM **24** (1), 1977, pp. 68-95.

[AN80] A. Arnold, M. Nivat, *The Metric Space of Infinite Trees. Algebraic and Topological Properties*, Fundamenta Informaticae **4**, 1980, pp. 445-476.

[BE74] A. Bastiani, C. Ehresmann *Multiple Functors I: Limits Relative to Double Categories*, Cahiers de Topologie ed Géométrie Différentielle **15** (3), 1974, pp. 545-621.

[BE+87] H.P. Barendregt, M.C.J.D. van Eekelen, J.R.W. Glauert, J.R. Kennaway, M.J. Plasmeijer, M.R. Sleep, *Term Graph Reduction*, in Proc. PARLE, LNCS 259, 1987, pp. 141-158.

[Ben75] D.B. Benson, *The Basic Algebraic Structures in Categories of Derivations*, Information and Control **28**, 1975, pp. 1-29.

[BK84] J.A. Bergstra, J.W. Klop, *Process Algebra for Synchronous Communication*, Information and Computation **60**, 1984, pp. 109-137.

[Blo76] S. Bloom, *Varieties of Ordered Algebras*, Journal of Comp. and System Science **13**, 1976, pp. 200–212.

[Bou85] G. Boudol, *Computational Semantics of Term Rewriting Systems*, in Algebraic Methods in Semantics, eds. M.Nivat and J. Reynolds, Cambridge University Press, 1985.

[Bur91] A. Burroni, *Higher Dimensional Word Problem*, in CTCS'91, LNCS 530, pp. 94–105.

[BW90] M. Barr, C. Wells, *Category Theory for Computing Science*, Prentice Hall Series in Computer Science, 1990.

[CD96] A. Corradini, F. Drewes, *(Cyclic) Term Graph Rewriting is Adequate for Rational Parallel Term Rewriting*, draft.

[CG95] A. Corradini, F. Gadducci, *CPO Models for Infinite Term Rewriting*, in Proc. AMAST'95, LNCS 936, 1995, pp. 368–384.

[CGM95] A. Corradini, F. Gadducci, U. Montanari, *Relating Two Categorical Models of Concurrency*, in Proc. RTA'95, LNCS 914, 1995, pp. 225–240.

[CM92] A. Corradini, U. Montanari, *An Algebraic Semantics for Structured Transition Systems and its Application to Logic Programs*, Theoretical Computer Science **103**, 1992, pp. 51-106.

[CM+91] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, M. Löwe, *Logic Programming and Graph Grammars*, in Proc. of the 4^{th} International Workshop on Graph-Grammars and Their Application to Computer Science, LNCS 532, 1991, pp. 221-237.

[Cor93] A. Corradini, *Term Rewriting in CT_Σ*, in Proc. TAPSOFT'93 (CAAP), LNCS 668, 1993, pp. 468–484.

[Cor96] A. Corradini, *Term Rewriting, in Parallel*, draft.

[CR93] A. Corradini, F. Rossi, *Hyperedge Replacement Jungle Rewriting for Term Rewriting Systems and Logic Programming*, in Theoretical Computer Science 109 (1) pp 7-48.

[DeS85] R. De Simone *Higher Level Synchronizing Devices in MEIJE-SCCS*, Theoretical Computer Science **37** (3), pp. 245–267, 1985.

[DJ90] N. Dershowitz, J.-P. Jouannaud, *Rewrite Systems*, in Handbook of Theoretical Computer Science, Vol. B, ed. J. van Leeuwen, North Holland, 1990, pp. 243-320.

[DKP91] N. Dershowitz, S. Kaplan, D.A. Plaisted, *Rewrite, Rewrite, Rewrite, Rewrite, Rewrite,...**, in Selected Papers of 16^{th} ICALP (1989), Theoretical Computer Science **83**, 1991, pp. 71-96.

[DP93] R. Dawson, R. Paré, *General Associativity and General Composition for Double Categories*, in Cahiers de Topologie et Géométrie Différentielle Catégoriques **25** (1), 1993, pp. 57-79.

[EPS93] M.C.J.D. van Eekelen, M.J. Plasmeijer, M.R. Sleep, *Term Graph Rewriting, Theory and Practice*, John Wiley & Sons, 1993.

[Gog90] J.A. Goguen, *Semantic Specifications for the Rewrite Rule Machine*, in Concurrency: Theory, Language and Architecture, eds. A. Yonezawa, W. McColl and T. Ito, LNCS 491, 1990, pp. 216-234.

[GK+87] J.A. Goguen, C. Kirchner, H. Kirchner, J. Meseguer, T. Winkler, *An Introduction to OBJ-3*, in Proc. CTRS'87, LNCS 308, pp. 258-263.

[GKM87] J.A. Goguen, C. Kirchner, J. Meseguer, *Concurrent Term Rewriting as a Model of Computation*, in Proc. Graph Reduction Workshop, LNCS 279, pp. 53-93.

[GM92] J.A. Goguen, J. Meseguer, *Order-Sorted Algebra I*, Theoretical Computer Science **105**, 1992, pp. 217-273.

[GM95a] F. Gadducci, U. Montanari, *Enriched Categories as Models of Computations*, in Italian Conference on Theoretical Computer Science 1995, ed. A. De Santis, World Scientific, to appear.

[GM95b] F. Gadducci, U. Montanari, *SOS Contexts as Cells in Double-Categories*, draft.

[Gra79] G. Grätzer, *Universal Algebra*, Springer-Verlag, New York, 1979.

[Gue81] I. Guesserian, *Algebraic Semantics*, LNCS 99, 1981.

[HL91] G. Huet, J.J. Lévy, *Computations in Orthogonal Rewriting Systems*, in Computational Logic, Essays in Honour of A. Robinson, eds. J.L. Lassez and G. Plotkin, MIT Press, 1991, chapter 11.

[Hoa85] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[Hue80] G. Huet, *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*, Journal of the ACM **27** (4), 1980, pp. 797-821.

[Jac93] B. Jacobs, *Semantics of Weakening and Contraction*, Annals of Pure and Applied Logic **69**, 1994, pp 73-106.

[KB70] D.E. Knuth, P.B. Bendix, *Simple Word Preoblem in Universal Algebra* in Computational Problems in Abstract Algebra, Pergamon Press, Oxford, 1970, pp. 263-297.

[Kel82] G.M. Kelly, *Basic Concepts of Enriched Category Theory*, London Mathematical Society, LN Series 64, 1982.

[KKR90] C. Kirchner, H. Kirchner, M. Rusinovitch, *Deduction with Symbolic Constraints*, in Revue d'Intelligence Artificielle **4** (3), 1990, pp. 9-52.

[KK+93] J.R. Kennaway, J.W. Klop, M.R. Sleep, F.J. de Vries, *Event Structures and Orthogonal Term Graph Rewriting*, in [EPS93], pp. 141-155.

[KK+94] J.R. Kennaway, J.W. Klop, M.R. Sleep, F.J. de Vries, *The Adequacy of Term Graph Rewriting for Simulating Term Rewriting*, ACM Transaction on Programming Languages and Systems **12** (1), 1994, pp. 493-523.

[KK+95] J.R. Kennaway, J.W. Klop, M.R. Sleep, F.J. de Vries, *Transfinite Reductions in Orthogonal Term Rewriting System*, Information and Computation **119** (1), pp. 18-38.

[KKV95] C. Kirchner, H. Kirchner, M. Vittek, *Designing Constraint Logic Programming Languages Using Computational Systems*, in Principles and Practice of Constraint Programming, eds. P. Van Hentenryck and V. Saraswat, MIT Press, 1995, pp. 131-158.

[Klo91] J.W. Klop, *Term Rewriting Systems*, to appear in Handbook of Logic in Computer Science, Vol. I, eds. S. Abramsky, D. Gabbay, and T. Maibaum, Oxford University Press, 1991.

[KR77] A. Kock, G.E. Reyes, *Doctrines in Categorical Logic*, in Handbook of Mathematical Logic, ed. John Bairwise, North Holland, 1977, pp. 283-313.

[KS74] G.M. Kelly, R.H. Street, *Review of the Elements of 2-categories*, Lecture Notes in Mathematics 420, 1974, pp. 75-103.

[Laf95] J. Lafont, *Equational Reasoning with 2-dimensional Diagrams* in Term Rewriting, French Spring School of Theoretical Computer Science, Font-Romeu, May 1993, LNCS 909, 1995, pp. 170-195.

[Lam68] J. Lambek, *Deductive Systems and Categories I*, Mathematical Systems Theory **2**, 1968, pp. 287-318.

[Lam69] J. Lambek, *Deductive Systems and Categories II*, in Category Theory, Homology Theory and their Applications I, Lecture Notes in Mathematics 86, 1969, pp. 76-122.

[Lam72] J. Lambek, *Deductive Systems and Categories III*, in Toposes, Algebraic Geometry and Logic, Lecture Notes in Mathematics 274, 1972, pp. 57-82.

[Lan94] C. Laneve, *Distributive Evaluations of λ-calculus*, Fundamenta Informaticae **20** (4), 1994, pp. 333-352.

[Law63] F. W. Lawvere, *Functorial Semantics of Algebraic Theories*, Proc. National Academy of Science **50**, 1963, pp. 869-872.

[Law68] F. W. Lawvere, *Some Algebraic Problems in the Context of Functorial Semantics of Algebraic Theories*, in Reports of the 2^{th} Midwest Category Seminar, Lecture Notes in Mathematics 61, 1968, pp. 41-61.

[Lev80] J.J. Lévy, *Optimal Reductions in the λ-calculus*, in To H.B. Curry, Essays in Combinatory Logic, Lambda Calculus and Formalism, eds. J.P. Seldin and J.R. Hindley, Academic Press, 1980, pp. 159-191.

[LM92] C. Laneve, U. Montanari, *Axiomatizing Permutation Equivalence in the λ-calculus*, in Proc. 3^{rd} International Conference on Algebraic and Logic Programming, LNCS 632, 1992, pp. 350-363. To appear in Mathematical Structures in Computer Science.

[LMR94] P. Lincoln, J. Meseguer, L. Ricciulli, *The Rewrite Rule Machine Node Architecture and its Performance*, in Proc. CONPAR'94, LNCS 854, 1994, pp. 509-520.

[LX90] K.G. Larsen, L. Xinxin, *Compositionality Through an Operational Semantics of Contexts*, in Proc. ICALP'90, LNCS 443, 1990, pp. 526-539.

[Mar91] L. Maranget, *Optimal Derivations in Weak λ-calculi and in Orthogonal Term Rewriting Systems*, in Proc. POPL'91, 1991, pp. 255-269.

[Mit72] B. Mitchell, *Rings with Several Objects*, in Advances in Mathematics **8**, 1972, pp. 1-161.

[Mes90] J. Meseguer, *Rewriting as a Unified Model of Concurrency*, SRI Technical Report, CSL-93-02R, 1990. See in particular the appendix on *Functorial Semantics of Rewrite Systems*.

[Mes92] J. Meseguer, *Conditional Rewriting Logic as a Unified Model of Concurrency*, in Selected Papers of 2^{th} Workshop on Concurrency and Compositionality, Theoretical Computer Science **96**, 1992, pp. 73-155.

[Mil89] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[ML71] S. MacLane, *Categories for the Working Mathematician*, Springer, 1971.

[MM90] J. Meseguer, U. Montanari, *Petri Nets are Monoids*, Information and Computation **88**, 1990, pp. 105-154.

[MOM91] N. Martí-Oliet, J. Meseguer, *From Petri Nets to Linear Logic through Categories: a Survey*, International Journal of Foundations in Computer Science **2** (4), 1991, pp. 297-399.

[MOM93] N. Martí-Oliet, J. Meseguer, *Rewriting Logic as a Logical and Semantic Framework*, SRI Technical Report, CSL-93-05, 1993.

[Pfe74] Michael Pfender, *Universal Algebra in S-Monoidal Categories*, Algebra-Berichte 20, Mathematisches Institut der Universität Munchen, 1974.

[Plo81] G. Plotkin, *A Structural Approach to Operational Semantics*, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pow89] A.J. Power, *An Abstract Formulation for Rewrite Systems*, in Proc. CTCS'89, LNCS 389, 1989, pp. 300-312.

[Pow90] A.J. Power, *A 2-categorical Pasting Theorem*, Journal of Algebra **129**, 1990, pp. 439-445.

[Rei85] W. Reisig, *Petri Nets*, Springer-Verlag, 1985.

[Rei87] H. Reichel, *Initial Computability, Algebraic Specification and Partial Algebras*, Oxford University Press, 1987.

[RS87] D.E. Rydeheard and J.G. Stell, *Foundations of Equational Deduction: A Categorical Treatment of Equational Proofs and Unification Algorithm*, in Proc. CTCS'87, LNCS 283, 1987, pp. 114-339.

[See87] R.A.G. Seely, *Modelling Computations: a 2-categorical Framework*, in Proc. 2^{nd} Symposium on Logic in Computer Science, 1987, pp. 65-71.

[Ste92] J. G. Stell, *Categorical Aspects of Unification and Rewriting*, Ph.D. Thesis, Faculty of Science, University of Manchester, 1992.

[Ste94] J. G. Stell, *Modelling Term Rewriting System by Sesqui-categories*, Technical Report TR94-02, Keele University, 1994. An abstract appear also in Actes des Journées Mathématiques "Catégories, Algèbres, Esquisses and Néo-Esquisses" (C.A.E.N.), 1994, pp. 121-126.

[Str92] R.H. Street, *Categorical Structures*, in Handbook of Algebra, eds M. Hazewinkel et al., Elsevier, preprint 1992.

[Wad71] C.P. Wadsworth, *Semantics and Pragmatics of the λ-Calculus*, Ph.D. Thesis, Oxford University, 1971.

[Wel89] C. Wells, *Path Grammars*, in Proc. Conference on Algebraic Methods and Software Technology, University of Iowa, May 1989.

[Vir95] P.Viry, *Rewrite modulo a Rewrite System*, Technical Report TR-20/85, Computer Science Department, University of Pisa, 1995.

[Vit94] M. Vittek, ELAN*: Un Cadre Logique pour le Prototypage de Langages de Programmation avec Contraintes*, Ph.D. thesis, University of Nancy, 1994.

[Win87] G. Winskel, *Event Structures*, in Advanced Course on Petri Nets, LNCS 255, 1987, pp. 325-392.

[Win89] G. Winskel, *An Introduction to Event Structures*, Lecture Notes for the REX Summerschool in Temporal Logic, LNCS 354, 1989, pp. 285-363.