Call by need computations in orthogonal term rewriting systems

Irène A. Durand

No d'ordre:

HABILITATION À DIRIGER LES RECHERCHES

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par Irène Durand

Call by need computations in orthogonal term rewriting systems

À soutenir le :	01 juillet 2005		
Après avis de :	MM.	Hubert Comon	Rapporteurs
		Jan Willem Klop	
		Sophie Tison	

Devant la commission d'examen formée de :

MM.	Robert Strandh	Professeur	Président
	Pascal Weil	Directeur de Recherche	Rapporteur
	Hubert Comon	Professeur	Examinateurs
	Aart Middeldorp	Professeur	
	Sophie Tison	Professeur	

– 2005 –

Call by need computations in orthogonal term rewriting systems

Irène Durand

June 8, 2011

Preamble

After my PHD thesis [Dur86] at the university of Toulouse on parallel implementation of logic programs, I deepened my knowledge on that domain by working on the PRISM system (parallel inference system for problem solving) developed at that time by the Professor Jack Minker research group at the university of Maryland (USA)[GKMD90].

After getting a research and teaching position at the university of Bordeaux (1989) and meeting Professor Robert Strandh [Str88] (former PHD student of Professor Mike O'Donnell [O'D77]), my interests moved to term rewriting systems (1990).

This document describes my research and programming work in computer science between 1990 and 2005. All this work takes place in the framework of term rewriting systems and call by need strategies for these systems. It is both theoretical and practical; most of the theoretical notions and algorithms discussed have been implemented in a software tool call `Autowrite`.

This document is an opportunity to present all our work related to call-by-need (published or unpublished) in a unified framework. Thanks to no severe limits on available space, the presentation may contain more details and examples than what can be given in a usual publication where space is always scarce. In some cases, the presentation given in this document departs significantly from the one previously given in our publications. It is for instance the case for the description of the algorithm for deciding whether a system is forward-branching. Also in more than ten years our writing has evolved; consequently older work has sometimes been rewritten with improved style.

We have tried to make this document as self-contained as possible so that a reader with some background in theoretical computer science could read it without looking for definitions elsewhere. All the given proofs are ours (with or without co-authors). An index of notation together with a general index can be found at the end of this document. Also Appendix A contains examples of systems that are used all along the document.

Acknowledgements

I would like to thank Aart Middeldorp for his collaboration over the years, in particular for his suggestions for improvements and ways to move forward. Thanks to Robert Strandh who introduced me to the domain of term rewriting and always encouraged my work.

I would like to thank the university of Bordeaux 1 for providing me with the conditions which made the conduct of that work possible. Thanks also to the universities of Warwick and Stuttgart for hosting me during one year and giving me good conditions to continue my research.

Thanks to the referees who have accepted to review this work.

Contents

Chapter 1

Introduction

The foundation of term rewriting is equational logic but for the sake of efficiency, the equations are oriented and become the rules of a term rewriting system. Term rewriting form a model of computation on algebraic data structures (terms).

Term rewriting systems play an important role in various domains of computer science such as automated theorem proving, functional programming, code generation, problem formalization (security of cryptographic protocols).

Rewriting starts with a ground term, and consists of repeatedly replacing a redex (an instance of a left-hand side) by its contractum (the corresponding right-handside after applying the substitution). Rewriting may eventually yield a term in normal form which is a term containing no redex.

Natural questions in term rewriting are:

- is the system terminating" (*i.e.* there are no infinite rewrite sequences)?
- "is the system confluent" (if a term rewrites independently to two terms t_1 and t_2, there exists a term s such that both t_1 and t_2 rewrite to s)?

We are interested in systems which can be used as programs so we want to allow non-terminating computations.

Confluence implies unicity of normal forms but does not imply termination. Confluent systems form a good framework for deterministic programming. They have the power of Turing machines. However confluence is not a decidable property for term rewriting systems. Orthogonal systems (*i.e.* linear and non-overlapping left-hand sides) which are always confluent form the framework of all this work, although some results may apply to the more general class of left-linear systems (linear left-hand sides).

The first point we want to address is "how to compute the normal form?" and not end up in an infinite computation when the normal form exists. The second is "how to do that efficiently?".

The following theorem of Huet and Lévy [HL91] forms the basis of all results on optimal normalizing rewrite strategies for orthogonal term rewrite systems:

"Every reducible term contains a needed redex, i.e., a redex which is contracted in every rewrite sequence to normal form, and repeated contraction of

needed redexes results in a normal form, if the term under consideration has a normal form".

Unfortunately, needed redexes are not computable in general. Hence, in order to obtain a *computable* optimal rewrite strategy, we are left to find (1) decidable approximations of neededness and (2) decidable properties of rewrite systems which ensure that every reducible term has a needed redex identified by (1). Starting with the seminal work of Huet and Lévy [HL91] on *strong sequentiality*, these issues have been extensively investigated in the literature [Com00, Jac96b, JS95, KM91, NST95, Oya93, Toy92]. In all these works Huet and Lévy's notions of index, ω-reduction, and sequentiality figure prominently. We present here our contributions to this domain.

The first part of the work is about extensions of the strongly sequential class. In a unified framework we define classes that admit call-by-need strategies (Chapter 3). Our framework is parameterized by the concept of approximation mapping and we show that recognizability preservation is the key to decidability, which is obtained by applying simple term automata techniques. Because of the high complexity of these classes (from EXPTIME to 3-EXPTIME) it was natural to investigate the question of modularity. In Chapter 4, we perform a detailed study of the modularity aspects of our framework. Root-stability has been shown to be the right notion when dealing with infinite normal forms but there is very little published work in that field. In Chapters 6 and 7, we extend our work to the problem of computing root-stable forms.

In second part, we go back to the original notion of strong sequentiality and compare it with our CBN classes. We show that our framework provides a better approximation to neededness than the sequentiality notions originating from the seminal paper of Huet and Lévy [HL79].

The strongly sequential class (SS) is known to be in EXPTIME but the problems whether it is in NP, co-NP, NP-complete, co-NP-complete remain open. The only known polynomial classes are subclasses of SS. In the third part, we present our work concerning such subclasses. This guideline for this work was the quest of finding classes that transform into the strongly sequential constructor class for which efficient stragegies exist.

The last part presents the `Autowrite` software. It was initially started as a tool to help us verify properties (in particular CBN properties) of systems. It has been really of help for the study of modularity helping us verify or find examples and counter-examples for our theories. First this software implemented mainly call-by-need theory to normal forms. To our knowledge, it is the only software to do so. For implementing call-by-need algorithms, it was necessary to also implement many tools concerning term automata and term rewriting systems. Now graphical interface gives the possibility to manipulate easily systems and term(tree) automata broadening the initial targets of Autowrite.

This work occurs in a rather restricted domain. However we have addressed many theoretical aspects (definition of new classes, decidability, complexity) as well as practical aspects through the development of the software `Autowrite`.

Chapter 2

Terminology

Here we give terminology, notation and well-known results that are useful all along this document. Additional specific terminology will be given locally when needed. Recall that an index of notation can be found at the end of the document.

2.1 Terms

2.1.1 Signature

A *signature* is a set of *function* symbols, each associated with an *arity* (a natural integer), which is the number of arguments accepted by the function. By $\mathsf{arity}(f)$ we denote the arity of a function symbol f. Given a signature $\mathcal{F}$ we denote by $\mathcal{F}_n$ be the subset of function symbols of $\mathcal{F}$ of arity n, $\mathcal{F} = \bigcup\{\mathcal{F}_n | n \geq 0\}$. Our signatures will often be denoted by the letters $\mathcal{F}$ or $\mathcal{G}$. A function symbol in $\mathcal{F}_0$ is called a *constant symbol*.

Example. *Let us take for instance the signature induced by the system of Example A.0.1* $\mathcal{F} = \{\mathsf{a}^{(0)}, \mathsf{b}^{(0)}, \mathsf{f}^{(2)}, \mathsf{g}^{(2)}\}$. *The arity of the symbols is indicated by the number between parentheses. We have* $\mathsf{arity}(\mathsf{f}) = 2$.

2.1.2 Variables

Let $\mathcal{V}$ be a denumerable set of *variables*.

Example. *The system of A.0.1 uses a single variable x. Variables will commonly be named, $x, y, z, \ldots$.*

2.1.3 Terms

Our expression language is the set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of first order *terms* formed from $\mathcal{F}$ and $\mathcal{V}$ defined as follows:

$\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$,

$f \in \mathcal{F}_n$ and $\forall i, 1 \leq i \leq n$, $M_i \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \Rightarrow f(M_1, \ldots, M_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.
We will use the usual equality symbol $=$ to denote the syntactic equality of terms.

Example. $M = \mathsf{g}(\mathsf{f}(\mathsf{a}, \mathsf{b}), x) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.

Let us denote the empty sequence by ε.

For any term M we define its set of *positions* $\mathcal{P}\mathsf{os}(M)$ as a finite subset of the set of finite sequences of positive integers as follows:
$\varepsilon \in \mathcal{P}\mathsf{os}(M)$,
$u \in \mathcal{P}\mathsf{os}(M_i) \Rightarrow iu \in \mathcal{P}\mathsf{os}(f(M_1, \ldots, ...M_n))$ for $1 \leq i \leq n$.
In the literature positions are also called *occurrences* or *paths*. Intuitively, a position of M names a subterm of M by its access path.

We write $\mathcal{P}\mathsf{os}^+(M)$ for $\mathcal{P}\mathsf{os}(M) \setminus \{\varepsilon\}$. This notation extends to any set of positions P: $P^+ = P \setminus \{\varepsilon\}$.

Example. $\mathcal{P}\mathsf{os}(M) = \{\varepsilon, 1, 1.1, 1.2, 2\}$ $\mathcal{P}\mathsf{os}^+(M) = \{1, 1.1, 1.2, 2\}$

If $u \in \mathcal{P}\mathsf{os}(M)$, we define the *subterm of M at u* as the term M/u defined inductively by
$M/\varepsilon = M$,
$f(M_1, \ldots, M_n)/iu = M_i/u$,
and we define the *root symbol* of M as the symbol $\mathsf{root}(M)$ defined by
$\mathsf{root}(M) = M$ if $M \in \mathcal{V} \cup \mathcal{F}_0$,
$\mathsf{root}(M) = f$ if $M = f(M_1, \ldots, M_n)$.

Example.

$M/\varepsilon = M$	$M/1 = \mathsf{f}(\mathsf{a}, \mathsf{b})$	$M/1.2 = \mathsf{b}$	$M/2 = x$
$\mathsf{root}(M) = \mathsf{g}$	$\mathsf{root}(\mathsf{f}(\mathsf{a}, \mathsf{b})) = \mathsf{f}$	$\mathsf{root}(\mathsf{b}) = \mathsf{b}$	$\mathsf{root}(x) = x$

A term M such that $\mathsf{root}(M) \in \mathcal{V}$ is called a *variable term* or simply a *variable*. A term M such that $\mathsf{root}(M) \in \mathcal{F}_0$ is called a *constant term* or simply a *constant*.

Finally, if $u \in \mathcal{P}\mathsf{os}(M)$, we define for every term N the *replacement in M at u by N* as the term $M[N]_u$ defined by
$M[N]_\varepsilon = N$,
$f(M_1, \ldots, M_n)[N]_{iu} = f(M_1, \ldots, M_i[N]_u, \ldots, M_n)$.

Example. $M[\mathsf{g}(\mathsf{a}, \mathsf{a})]_2 = \mathsf{g}(\mathsf{f}(\mathsf{a}, \mathsf{b}), \mathsf{g}(\mathsf{a}, \mathsf{a}))$ $M[\mathsf{a}]_1 = \mathsf{g}(\mathsf{a}, x)$

The set of positions is partially ordered by the *prefix order* $\leq$ given by: $u \leq v$ iff $\exists w$ such that $uw = v$. In this case we define v/u as w. We write $u < v$ if $u \leq v$ and $u \neq v$. We write $u \perp v$ (and call u and v *disjoint*) when $u \not\leq v$ and $v \not\leq u$.

Example. $1 < 1.2$, $2 \not\leq 1$, $1 \not\leq 2$, $1 \perp 2$, $1.1.2.2/1.1 = 2.2$.

Given a term M, a set of pairwise disjoint positions $\{u_1, \ldots, u_n\}$ of M and a set of terms $N_1, \ldots, N_n$, we can extend the concept of replacement to *parallel replacement*:

$$M[N_1, \ldots, N_n]_{u_1, \ldots, u_n} = M[N_1]_{u_1} \ldots [N_n]_{u_n}.$$

Example. $\mathsf{f}(\mathsf{a},\mathsf{b})[\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{a})]_{1,2} = \mathsf{f}(\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{a}))$

Given a term M and a symbol f, $\mathcal{P}\mathsf{os}_f(M)$ denotes the set of positions labelled f in M.

$\mathcal{P}\mathsf{os}_f(M) = \{u \in \mathcal{P}\mathsf{os}(M) | \mathsf{root}(M/u) = f\}$.

Example. $\mathcal{P}\mathsf{os}_f(M) = \{1.1\}$.

Given a term M, $\mathcal{V}\mathsf{ar}(M)$ denotes the set of variables of M. We also use the notation $\mathcal{P}\mathsf{os}_{\mathcal{V}}(M)$ (resp. $\mathcal{P}\mathsf{os}_{\overline{\mathcal{V}}}(M)$) to denote the set of variable (resp. nonvariable) positions in M.

$\mathcal{P}\mathsf{os}_{\mathcal{V}}(M) = \{u \in \mathcal{P}\mathsf{os}(M) | M/u \in \mathcal{V}\}$.

$\mathcal{P}\mathsf{os}_{\overline{\mathcal{V}}}(M) = \{u \in \mathcal{P}\mathsf{os}(M) | M/u \notin \mathcal{V}\}$.

Example.

$$\mathcal{V}\mathsf{ar}(M) = \{x\}$$
$$\mathcal{P}\mathsf{os}_{\mathcal{V}}(M) = \{2\} \quad \mathcal{P}\mathsf{os}_{\overline{\mathcal{V}}}(M) = \{\varepsilon, 1, 1.1, 1.2\}$$

We denote by $\mathcal{S}\mathsf{ub}(M)$ the set of subterms of a term M:

$\mathcal{S}\mathsf{ub}(M) = \{M/u \mid u \in \mathcal{P}\mathsf{os}(M)\}$.

This notation extends to a set of terms S: $\mathcal{S}\mathsf{ub}(S) = \bigcup_{M \in S} \mathcal{S}\mathsf{ub}(M)$.

A subterm M/u of M with $u > \varepsilon$ is called a *proper subterm* of M.

Example. $\mathcal{S}\mathsf{ub}(M) = \{\mathsf{a}, \mathsf{b}, \mathsf{f}(\mathsf{a},\mathsf{b}), x, M\}$ *and* $\mathsf{f}(\mathsf{a},\mathsf{b})$ *is a proper subterm of* M.

A *substitution* σ is a mapping from $\mathcal{T}(\mathcal{F},\mathcal{V})$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$ satisfying $\sigma(f(M_1,\ldots,M_n)) = f(\sigma(M_1),\ldots,\sigma(M_n))$. So, σ is determined by its restriction to the set of variables $\mathcal{V}$. For convenience, $\sigma(M)$ is often written $M\sigma$.

We say that the term M is an *instance* of the term N if there exists a substitution σ such that $M = \sigma(N)$.

Example. $\mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{a})$ *is an instance of* $\mathsf{g}(\mathsf{f}(x,y),x)$ *as* $\mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{a}) = \mathsf{g}(\mathsf{f}(x,y),x)\sigma$ *with the substitution* σ *such that* $x\sigma = \mathsf{a}$ *and* $y\sigma = \mathsf{b}$.

We say that two terms M and N *unify* (or are *unifiable*) if there exists a substitution σ such that $M\sigma = N\sigma$. Note that to check whether two terms are unifiable their variables should be first renamed in order that the two terms do not share any variable.

Example. $\mathsf{g}(\mathsf{f}(\mathsf{a},x),\mathsf{b})$ *and* $\mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{a}),y)$ *are unifiable with the substitution* σ *such that* $x\sigma = \mathsf{a}$ *and* $y\sigma = \mathsf{b}$.

A *ground term* does not contain variables. The set of ground terms is written $\mathcal{T}(\mathcal{F})$ often abbreviated $\mathcal{T}$ whenever $\mathcal{F}$ is understood.

A *linear term* does not contain multiple occurrences of the same variable.

Example. $\mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{a})$ *is ground hence linear.* $\mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{b}),x)$ *is linear, not ground.* $\mathsf{g}(\mathsf{f}(x,\mathsf{b}),x)$ *is not linear, not ground.*

2.1.4 Contexts

Let $\Box$ be a special constant symbol which cannot appear in any signature. $\Box$ denotes an empty position (a *hole*) in a term. A *context* is a term of $\mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$. A context with one hole at position u is often denoted by $C[]_u$ or $C[]$ when u is not relevant. If $\Box$ is replaced by a term M of $\mathcal{T}(\mathcal{F}, \mathcal{V})$, we obtain the term $C[M]_u$ or $C[M]$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A context can also have several holes at positions $u_1, \ldots, u_n$ often denoted by $C[, \ldots,]_{u_1,\ldots,u_n}$ or $C[, \ldots,]$. If all the $\Box$ are replaced by terms $M_1, \ldots, M_n$ of $\mathcal{T}(\mathcal{F}, \mathcal{V})$ we obtain the term $C[M_1, \ldots, M_n]_{u_1,\ldots,u_n}$ or $C[M_1, \ldots, M_n]$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

2.2 Term Rewriting Systems

2.2.1 Term rewriting rules

A *rewrite rule* is a pair $L \to R$ of terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ that satisfy $L \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(R) \subseteq \mathcal{V}\mathsf{ar}(L)$. If the second condition is not imposed we find it useful to speak of *extended rewrite rule*. L (resp. R) is called the *left-hand side* (resp. *right-handside*) of the rule. A rule is *linear* (resp. *ground*) when both sides are linear (resp. ground). A rule is *left-linear* (resp. *left-ground*) when its left-hand side is linear (resp. ground). A rule is *right-linear* (resp. *right-ground*) when its right-handside is linear (resp. ground).

2.2.2 Term rewriting systems

A *term rewrite system* (*system* for short) is a pair $(\mathcal{R}, \mathcal{F})$ where $\mathcal{F}$ is a signature and $\mathcal{R}$ a set of rewrite rules built upon the signature $\mathcal{F}$. When $\mathcal{R}$ contains extended rewrite rules we will call it an *extended system* (*esystem*). Such esystems arise naturally when we approximate systems, as explained in Section 3.3. When $\mathcal{F}$ contains exactly the function symbols appearing in $\mathcal{R}$ or when $\mathcal{F}$ is understood, we may write $\mathcal{R}$ instead of $(\mathcal{R}, \mathcal{F})$. We write $\mathsf{LHS}_\mathcal{R}$ (or LHS whenever $\mathcal{R}$ is understood) to denote the set of the left-hand sides of $\mathcal{R}$.

2.2.3 Redexes and Normal forms

We consider the system of Example A.0.1 to illustrate most of our definitions.

A *redex* is a term which is an instance of the left-hand side of a rewrite rule. A position u in a term t such that M/u is a redex is called a *redex position*. The set of redex positions of a term M with regard to a system $\mathcal{R}$ is denoted by $\mathsf{REDEX}_\mathcal{R}(M)$ or $\mathsf{REDEX}(M)$ whenever $\mathcal{R}$ is understood.

When $\mathsf{REDEX}(M) \neq \varnothing$ we say that M is *reducible*. A term M is a *normal form* (or *in normal form*, or *irreducible*) if $\mathsf{REDEX}_\mathcal{R}(M) = \varnothing$.

The set of all ground normal forms of a system $(\mathcal{R}, \mathcal{F})$ is denoted by $\mathsf{NF}(\mathcal{R}, \mathcal{F})$. We may write also $\mathsf{NF}(\mathcal{R})$, $\mathsf{NF}(\mathcal{F})$, NF whenever $\mathcal{F}$, $\mathcal{R}$ or both $\mathcal{F}$ and $\mathcal{R}$ are understood.

Example. $\mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{a}),\mathsf{a})$ *is a redex.* $\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{a}),\mathsf{a}),\mathsf{a})$ *is reducible.* $\mathsf{REDEX}(\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{a}),\mathsf{a}),\mathsf{a})) = \{1\}$. $\mathsf{f}(\mathsf{a},\mathsf{a}) \in \mathsf{NF}(\mathcal{R})$.

A redex in a term is *innermost* if it does not contain smaller redexes. A redex in a term is *outermost* if it is not a proper subterm of another redex in the same term.

A normal form is *external* if it is not an instance of a proper non-variable subterm of a left-hand side of a rewrite rule in $\mathcal{R}$. The set of all ground external normal forms of a system $\mathcal{R}$ is denoted by $\mathsf{ENF}(\mathcal{R})$.

Example. *The term* $\mathsf{g}(\mathsf{a},\mathsf{a})$ *is not an external normal form as it is an instance of* $\mathsf{g}(x,\mathsf{a})$ *but* $\mathsf{g}(\mathsf{g}(\mathsf{a},\mathsf{a}),\mathsf{g}(\mathsf{a},\mathsf{a}))$ *is one.*

2.2.4 Reduction (Rewriting)

We say that a term M *reduces* (or *rewrites*) *to* N *at position* u *using rule* $L \to R$ iff there exists a substitution σ such that $M/u = L\sigma$ and $N = M[R\sigma]_u$. We write $M \to N$ when M reduces to N. We use $\stackrel{*}{\to}$ to denote the reflexive and transitive closure of $\to$. $R\sigma$ is called the *contractum* of the redex $L\sigma$.

When applying an extended rewrite rule $L \to R$ of an esystem, the variables in $\mathcal{V}\mathsf{ar}(R) \setminus \mathcal{V}\mathsf{ar}(L)$ may be instantiated by arbitrary ground terms.

In all this work, we are dealing with **finite** systems only. Moreover, we consider rewriting on **ground** terms only, except in Chapter 9 for reasons explained there.

We write $N \Vdash M$ if M can be obtained from N by contracting a set (possibly empty) of redexes at pairwise disjoint positions in N. In other words, $N = C[N_1,\ldots,N_n]$ and $M = C[M_1,\ldots,M_n]$ for some context C and terms $N_1,\ldots,N_n$, $M_1,\ldots,M_n$ with $N_i \to M_i$ for all $1 \leqslant i \leqslant n$. The relation $\Vdash$ is called *parallel rewriting*.

A binary relation R on ground terms is called *parallel relation* if $C[s_1,\ldots,s_n]\ R\ C[t_1,\ldots,t_n]$ whenever $s_1\ R\ t_1,\ \ldots,\ s_n\ R\ t_n$, for all contexts C and terms $s_1,\ldots,s_n,t_1,\ldots,t_n$. Note that relation $\Vdash$ defined above is parallel. Actually, $\Vdash$ is the smallest parallel relation that contains $\to$, i.e., the *parallel closure* of $\to$.

A term is $\mathcal{R}$*-root-stable* or simply *root-stable* (whenever $\mathcal{R}$ is understood) if it cannot be rewritten to a redex. The set of root-stable terms with regards to a system $\mathcal{R}$ is denoted by $\mathsf{RS}_{\mathcal{R}}$.

Example. *The term* $\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{b},\mathsf{a}),\mathsf{a}),\mathsf{g}(\mathsf{b},\mathsf{b}))$ *is not root-stable as it reduces to the redex* $\mathsf{g}(\mathsf{b},\mathsf{b})$*:*

$$\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{b},\mathsf{a}),\mathsf{a}),\mathsf{g}(\mathsf{b},\mathsf{b})) \to \mathsf{g}(\mathsf{b},\mathsf{g}(\mathsf{b},\mathsf{b})) \to \mathsf{g}(\mathsf{b},\mathsf{b})$$

but the term $\mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{a}),\mathsf{a}),\mathsf{g}(\mathsf{b},\mathsf{b}))$ *is root-stable.*

We denote by $\mathsf{WN}(\mathcal{R},\mathcal{F})$ the set of all ground terms in $\mathcal{T}(\mathcal{F})$ that rewrite in $\mathcal{R}$ to a normal form in $\mathsf{NF}(\mathcal{R},\mathcal{F})$. If no confusion can arise, we just write $\mathsf{WN}(\mathcal{R})$.

Example. *Just for this example let us take* $\mathcal{R} = \{\mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(\mathsf{a}, x), \mathsf{f}(x, \mathsf{b}) \to \mathsf{b}\}$. *We have* $\mathsf{f}(\mathsf{b}, \mathsf{a}) \in \mathsf{WN}(\mathcal{R}, \mathcal{F})$ *and* $\mathsf{f}(\mathsf{a}, \mathsf{b}) \notin \mathsf{WN}(\mathcal{R}, \mathcal{F})$.

Let $\mathcal{F} \subseteq \mathcal{G}$. We denote by $\mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F})$ the set of terms in $\mathcal{T}(\mathcal{F})$ that have a normal form with respect to $(\mathcal{R}, \mathcal{G})$. This useful only when we consider esystems and when a free variable of a right-handside unifies with a term containing a symbol in $\mathcal{G} \setminus \mathcal{F}$.

Example. *Just for this example let us take the esytem* $\mathcal{R} = \{\mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(\mathsf{a}, y)\}$ *with its implicit signature* $\mathcal{F} = \{\mathsf{a}^{(0)}, \mathsf{f}^{(2)}\}$. *We have* $\mathsf{f}(\mathsf{a}, \mathsf{a}) \notin \mathsf{WN}(\mathcal{R}, \mathcal{F})$. *Consider now the extended signature* $\mathcal{G} = \mathcal{F} \cup \{\mathsf{b}^{(0)}\}$. *This time, we have* $\mathsf{f}(\mathsf{a}, \mathsf{a}) \in \mathsf{WN}(\mathcal{R}, \mathcal{F})$ *because* $\mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{f}(\mathsf{a}, \mathsf{b}) \in \mathsf{NF}(\mathcal{R}, \mathcal{G})$. *So* $\mathsf{WN}(\mathcal{R}, \mathcal{F}) \neq \mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F})$. *Note that* $\mathsf{f}(\mathsf{b}, \mathsf{a}) \notin \mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F})$ *because* $\mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F})$ *contains only terms in* $\mathcal{T}(\mathcal{F})$ *by definition.*

2.2.5 Properties of term rewriting systems

A system is *linear* (resp. *ground*, *left-linear*, *left-ground*, *right-linear*, *right-ground*) if all its rules are linear (resp. ground, left-linear, left-ground, right-linear, right-ground).

A system $\mathcal{R}$ is *overlapping* if there exists two distinct rules $L \to R$ and $L' \to R'$ such that L and L' are unifiable or if there exists two rules $L \to R$ and $L' \to R'$ and $u \in \mathcal{P}\mathsf{os}_{\overline{\mathcal{V}}}(L)$ such that L/u and L' are unifiable.

A left-linear system which is non-overlapping is called *orthogonal*.

Example.
$\mathcal{R} = \{\mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to x, \mathsf{g}(\mathsf{a}, x) \to \mathsf{b}\}$ *is overlapping hence not orthogonal.*
$\mathcal{R} = \{\mathsf{g}(x, x) \to \mathsf{a}\}$ *is not left-linear hence not orthogonal.*
$\mathcal{R} = \{\mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to \mathsf{g}(x, x), \mathsf{g}(\mathsf{a}, \mathsf{b}) \to \mathsf{b}\}$ *is left-linear and non-overlapping hence orthogonal.*

A system $\mathcal{R}$ is *confluent* if forall terms M, N_1, N_2 such that $M \to^* N_1$ and $M \to^* N_2$ ther exists a term N such that $N_1 \to^* N$ and $N_2 \to^* N$.

[Ros73] showed that for orthogonal systems the relation $\to$ is confluent (has the *Church-Rosser* property). Orthogonal systems have the property that every term has at most one normal form.

A rewrite rule is *collapsing* if its right-handside is a variable. A redex with respect to a collapsing rewrite rule is also called *collapsing* and so is called an esystem that contains a collapsing rewrite rule.

Let $(\mathcal{R}, \mathcal{F})$ be a system. A function symbol in $\mathcal{F}$ is called a *defined symbol* if it is the root symbol of a left-hand side of a rewrite rule in $\mathcal{R}$. The function symbols in $\mathcal{F}$ that are not defined are called *constructor symbol*s. We use $\mathcal{F}_{\mathcal{D}}$ and $\mathcal{F}_{\mathcal{C}}$ to denote the set of defined and constructor symbols. A term is said *constructor* if all its inner symbols are constructors. A system $\mathcal{R}$ such that all terms of $\mathsf{LHS}_{\mathcal{R}}$ are constructor is called a *constructor system*. The class of constructor systems is denoted by C.

Given a term M, we denote by $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(M)$ the set of subterms of M having a defined symbol at their root:

$\mathcal{S}\mathsf{ub}_{\mathcal{D}}(M) = \{T \in \mathcal{S}\mathsf{ub}(M) \mid \mathsf{root}(T) \in \mathcal{F}_{\mathcal{D}}\}$. We also write $\mathcal{S}\mathsf{ub}^{+}_{\mathcal{D}}(M) = \mathcal{S}\mathsf{ub}_{\mathcal{D}}(M) \setminus \{M\}$.
These notations extend to a set of terms S: $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(S) = \bigcup_{M\in S} \mathcal{S}\mathsf{ub}_{\mathcal{D}}(M)$ and $\mathcal{S}\mathsf{ub}^{+}_{\mathcal{D}}(S) = \mathcal{S}\mathsf{ub}_{\mathcal{D}}(S) \setminus S$. Note that if S is orthogonal (S the set of left-hand sides of an orthogonal system) $\mathcal{S}\mathsf{ub}^{+}_{\mathcal{D}}(S) = \mathcal{S}\mathsf{ub}_{\mathcal{D}}(S) \setminus S = \bigcup_{M\in S} \mathcal{S}\mathsf{ub}^{+}_{\mathcal{D}}(M)$.

2.3 Prefixes of Terms: Ω-Terms

We represent *prefixes* of terms by Ω-*term*, i.e, by terms where the new constant Ω can occur. Let $\mathcal{F}_\Omega = \mathcal{F} \cup \{\Omega\}$ and let $\mathcal{T}(\mathcal{F}_\Omega, \mathcal{V})$ be the set of these Ω-terms. Most of the time we will deal with *ground* Ω-terms (members of $\mathcal{T}(\mathcal{F}_\Omega)$).
Let us consider the *prefix order* $\preceq$ on $\mathcal{T}(\mathcal{F}_\Omega, \mathcal{V})$ defined by
$\Omega \preceq M$ for all $M \in \mathcal{T}(\mathcal{F}_\Omega, \mathcal{V})$,
$f(M_1, \ldots, M_n) \preceq f(N_1, \ldots, N_n)$ iff $M_i \preceq N_i$ for $1 \leq i \leq n$,
$x \preceq x$ for each variable.
Given a set of Ω-terms S we write $\mathsf{Max}(S)$ (resp. $\mathsf{Min}(S)$) for the maximal (resp. minimal) elements of S. We also write S^* for the set $S \setminus \{\Omega\}$.

All the previously defined operations on terms extend obiously to Ω-terms. One just adds the following notions:

A position $u \in \mathcal{P}\mathsf{os}(M)$ such that $\mathsf{root}(M/u) = \Omega$ is called an Ω-*position*. We write $\mathcal{P}\mathsf{os}_\Omega$ (resp. $\mathcal{P}\mathsf{os}_{\overline{\Omega}}$) to denote the set of Ω-positions (resp. non-Ω-positions) of M:
$\mathcal{P}\mathsf{os}_\Omega(M) = \{u \in \mathcal{P}\mathsf{os}(M) | M/u = \Omega\}$,
$\mathcal{P}\mathsf{os}_{\overline{\Omega}}(M) = \{u \in \mathcal{P}\mathsf{os}(M) | M/u \neq \Omega\}$.

If $M \preceq T$ and $N \preceq T$ for some T, then M and N are said to be *compatible* which is written $M \uparrow N$. If S is a set of terms, we also write $M \uparrow S$ (resp. $M \succ S$, $M \prec S$) when there exists $N \in S$ such that $M \uparrow N$ (resp. $M \succ N$, $M \prec N$).

We will write M_Ω for $\sigma(M)$ where σ is the substitution such that $\sigma(x) = \Omega$ for all variables x. We extend this notation to a set of terms S: $S_\Omega = \{M_\Omega \mid M \in S\}$.

Let M be an Ω-term. By $M^\prec$ we denote the set of strict ($\neq \Omega$) and proper prefixes of M. Similarly, if S is a set of Ω-terms $S^\prec = \cup_{M\in S} M^\prec$.

An Ω-term N such that $\mathsf{REDEX}(N) = \varnothing$ will be said in Ω-*normal form*. We reserve the phrase *normal form* for terms containing neither redexes nor Ω's.

An element of LHS_Ω is called a *redex scheme* or simply *scheme*.

Example. *For the system of Example A.0.1, we obtain the set of redex schemes:* $\mathsf{LHS}_\Omega = \{\mathsf{f}(\mathsf{g}(\mathsf{a}, \Omega), \mathsf{a}), \mathsf{f}(\mathsf{g}(\Omega, \mathsf{a}), \mathsf{b}), \mathsf{g}(\mathsf{b}, \mathsf{b})\}$.

Lemma 2.3.1. *If* $\mathcal{R} \in \mathsf{C}$ *then* $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega) = \mathsf{LHS}_\Omega$.

An Ω-term M is *redex compatible* if it can be refined to a redex (*i.e.* $M \uparrow \mathsf{LHS}_\Omega$).

We call an Ω-term M a *preredex* if $M \prec \mathsf{LHS}_\Omega$. A *strict preredex* is a preredex not equal to Ω. The set of strict preredexes is then $\mathsf{LHS}^\prec_\Omega$, An element

of $\mathcal{S}\mathsf{ub}^+_{\mathcal{D}}(\mathsf{LHS}_\Omega) = \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega) \setminus \mathsf{LHS}_\Omega$ is called a *subscheme*. Note that from the definition of $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$, $\mathsf{LHS}_\Omega \subseteq \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$.

Example. $\mathsf{f}(\mathsf{g}(\Omega, \mathsf{b}), \mathsf{a})$ *is redex compatible.* $\mathsf{f}(\Omega, \mathsf{b})$ *is a proper preredex.* $\mathsf{g}(\mathsf{a}, \Omega)$ *is a subscheme.*

The notation $f(\vec{\Omega})$ denotes the term $f(\Omega, \ldots, \Omega)$. Given an Ω-term M and $u \in \mathcal{P}\mathsf{os}_\Omega(M)$, we define the *extension* of M at u by f, written $\mathsf{ext}(M, u, f)$ as the Ω-term $M[f(\vec{\Omega})]_u$.

Example. $\mathsf{ext}(\mathsf{f}(\Omega, \Omega), 1, \mathsf{g}) = \mathsf{f}(\mathsf{g}(\Omega, \Omega), \Omega)$.

2.4 Term Automata and Ground Term Transducers

We now recall some basic definitions and results concerning finite term automata[1]. Much more information can be found in [CDG+02]. Term automata are also called *tree automata* but as all this work is based on terms we prefer to call them term automata. A (finite bottom-up) *term automaton* is a quadruple $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Gamma)$ consisting of a finite signature $\mathcal{F}$, a finite set Q of *states*, disjoint from $\mathcal{F}$, a subset $Q_f \subseteq Q$ of final states, and a set of transition rules Γ. Every transition rule is of the form $f(q_1, \ldots, q_n) \to q$ with $f \in \mathcal{F}$ and $q_1, \ldots, q_n, q \in Q$ or $q \to q'$ with $q, q' \in Q$. The latter rules are called ϵ-transitions. So a term automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Gamma)$ is simply a finite ground term rewriting system Γ over the signature $\mathcal{F} \cup Q$ whose rewrite rules have a special shape, together with a subset Q_f of Q. The induced rewrite relation on $\mathcal{T}(\mathcal{F} \cup Q)$ is denoted by $\to_{\mathcal{A}}$. A ground term $M \in \mathcal{T}(\mathcal{F})$ is accepted by $\mathcal{A}$ if $M \to^+_{\mathcal{A}} q$ for some $q \in Q_f$. The set of all such terms is denoted by $L(\mathcal{A})$. A subset $L \subseteq \mathcal{T}(\mathcal{F})$ is called *recognizable* if there exists a term automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Gamma)$ such that $L = L(\mathcal{A})$.

Given a term t and a deterministic automaton $\mathcal{A} = (\mathcal{F}, Q, Q^f, \Gamma)$, let $t{\downarrow_{\mathcal{A}}}$ denote the states of $\mathcal{A}$ accessible from t using Γ. This notation extends to terms in $\mathcal{T}(\mathcal{G}, Q)$.
The same notation is used when $\mathcal{A}$ is non-deterministic: in that case $t{\downarrow_{\mathcal{A}}}$ denotes the set of all the states of $\mathcal{A}$ accessible from t using the rules of $\mathcal{A}$. This notation can be generalized to terms in $\mathcal{T}(\mathcal{F}, 2^Q)$: let $S_1, \ldots, S_n$ be subsets of Q, $f(S_1, \ldots, S_n){\downarrow_{\mathcal{A}}} = \{q \in Q, f(q_1, \ldots, q_n) \to_\Gamma q, \text{for some } q_1, \ldots, q_n \in S_1, \ldots, S_n\}$.

It is well-known that the set $\mathcal{T}(\mathcal{F})$ of all ground terms is recognizable. Other well-known properties are stated in the following two lemmata.

Lemma 2.4.1.

1. Recognizable languages are effectively closed under Boolean operations.

[1] To avoid using the two equivalent words, *terms* and *trees* to refer to terms we talk about *term automata* and *ground term transducers* instead of *tree automata* and *ground tree transducers* instead. The word *tree* will be used for the notion of *index tree*

2. Membership and emptiness are decidable for recognizable languages.

Lemma 2.4.2. *If $\mathcal{R}$ is a finite left-linear system then* $\mathsf{REDEX}(\mathcal{R})$ *and* $\mathsf{NF}(\mathcal{R})$ *are recognizable.*

Ground term transducers are defined in [DHLT90]. A *ground term transducer* (GTT for short) is a pair $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ of term automata over the same signature $\mathcal{F}$. Let $N, M \in \mathcal{T}(\mathcal{F})$. We say that the pair (N, M) is accepted by $\mathcal{G}$ if $N \to^*_{\mathcal{A}} u$ and $M \to^*_{\mathcal{B}} U$ for some term $U \in \mathcal{T}(\mathcal{F} \cup Q)$ where Q is the set of common states of $\mathcal{A}$ and $\mathcal{B}$. The set of all such pairs is denoted by $L(\mathcal{G})$. Observe that $L(\mathcal{G})$ is a binary relation on $\mathcal{T}(\mathcal{F})$. A binary relation on ground terms is called *gtt-recognizable* if there exists a ground term transducer that accepts it. Every gtt-recognizable relation R is parallel.

Ground term transducers were introduced by Dauchet and Tison [DT85] in order to prove that confluence is a decidable property of ground systems. In this work we make use of the following closure properties. They can be proved by adding appropriate ϵ-transitions. Part (2) originates from [CG90].

Lemma 2.4.3. *Let R be a gtt-recognizable relation on $\mathcal{T}(\mathcal{F})$.*

1. *The inverse relation R^{-1} of R is gtt-recognizable.*
2. *The transitive closure R^+ of R is gtt-recognizable.*
3. *If $L \subseteq \mathcal{T}(\mathcal{F})$ is recognizable then $R[L]^2 = \{N \mid N\ R\ M$ for some $M \in L\}$ is recognizable.*

We would like to emphasize that there are other notions of recognizability for binary relations in the literature. The gtt-recognizability defined above suffices for our purposes.

Weak Second-Order Monadic Logic

The material in this subsection is only used in Chapter 6. We assume familiarity with WSkS, the weak second-order monadic logic with k successors. See Thomas [Tho90, Section 11] for a discussion of WS2S.

Let $\mathcal{F}$ be a finite signature. Let k be the maximal arity of function symbols in $\mathcal{F}$ and let n be the cardinality of $\mathcal{F}$. A term $t \in \mathcal{T}(\mathcal{F})$ is represented in WSkS using $n+1$ set variables X and X_f for every $f \in \mathcal{F}$. In the following we write $\vec{X}$ for the sequence X, X_f for $f \in \mathcal{F}$. The WSkS formula $\mathsf{term}(\vec{X})$:

$$X = \bigcup_{f \in \mathcal{F}} X_f \ \wedge \bigwedge_{f \in \mathcal{F}} \forall x \in X_f \left[\bigwedge_{i=1}^{arity(f)} x{\cdot}i \in X \ \wedge \bigwedge_{i=arity(f)+1}^{k} x{\cdot}i \notin X \right] \wedge$$
$$\bigwedge_{f \neq g \in \mathcal{F}} X_f \cap X_g = \varnothing \ \wedge \ \forall x \in X \, \forall y < x \ [y \in X]$$

[2] In the literature $R[L]$ often denotes the different set $\{M \mid N\ R\ M$ for some $N \in L\}$. We find our choice more convenient.

expresses that $\vec{X}$ encodes a term in $\mathcal{T}(\mathcal{F})$. If $\mathsf{term}(\vec{T})$ holds for the sequence $\vec{T}$ of sets of positions then we define $t_{\vec{T}}$ to be the term in $\mathcal{T}(\mathcal{F})$ uniquely determined by $\mathcal{P}\mathsf{os}(t) = T$ and $\mathsf{root}(t_{|p}) = f$ if $p \in T_f$, for all $p \in T$. A subset L of $\mathcal{T}(\mathcal{F})$ is called WSkS *definable* if there exists a WSkS formula ϕ with free variables $\vec{X}$ such that $L = \{t_{\vec{T}} \mid \mathsf{term}(\vec{T}) \wedge \phi(\vec{T})\}$.

Theorem 2.4.4 ((Doner [Don70], Thatcher and Wright [TW68]))**.** *A set $L \subseteq \mathcal{T}(\mathcal{F})$ is WSkS definable if and only if it is recognizable.* □

Part I

Beyond strong sequentiality

Chapter 3

Call by need Strategies

Definition 3.0.5. *Given a term rewriting system and a reducible term t, a redex in t is* needed *if it is contracted in every rewrite sequence from t to normal form.*

The following theorem of Huet and Lévy [HL91] forms the basis of all results on optimal normalizing rewrite strategies for orthogonal term rewrite systems:

Theorem 3.0.6. *Given an orthogonal term rewriting system,*

1. *every reducible term contains a needed redex,*
2. *repeated contraction of needed redexes results in a normal form, if the term under consideration has a normal form.*

Unfortunately, needed redexes are not computable in general. In order to obtain a *computable* optimal rewrite strategy, we are left to find (1) decidable approximations of neededness and (2) decidable properties of systems which ensure that every reducible term has a needed redex identified by (1). Starting with the seminal work of Huet and Lévy [HL79] on strong sequentiality, these issues have been extensively investigated in the literature [Com00, Jac96b, JS95, KM91, NST95, Oya93, Toy92]. In all these works Huet and Lévy's concepts of index, ω-reduction, and sequentiality figure prominently.

In this chapter we present an approach to decidable call-by-need computations in which issues (1) and (2) above are addressed directly. Besides facilitating understanding this enables us to cover larger classes of rewrite systems than the ones based on the sequentiality approach. For instance, a trivial consequence of our work is that every orthogonal right-ground rewrite system admits a computable call-by-need strategy whereas none of the sequentiality-based approaches cover all such systems. Our approach is based on the easy but fundamental observation that needed redexes are *uniform* but not *independent* of other redexes in the same term. Uniformity means that only the position of a redex in a term counts for determining neededness.

From [HL79, Oya93, Com00] we extract the important concept of *approximation mapping*, which is used to parameterize our framework.

An approximation mapping transforms a rewrite system into a simpler one such that every rewrite step in the former can be simulated in the latter. We identify recognizability preservation as the key property that an approximation mapping α must have in order to obtain a decidable class CBN_α consisting of all rewrite systems that have the property that at least one of the needed redexes in every reducible term can be computed by α. Consequently, every rewrite system in CBN_α admits a computable call-by-need strategy. Inspired by Comon [Com00], our decidability results heavily rely on term automata techniques. However, by assigning a greater role to *ground term transducers* (GTT) we do not need to rely on weak second-order monadic logic.

The remainder of this chapter is organized as follows. In Section 3.1 we give a brief introduction to call-by-need strategies. In Section 3.2 we present sufficient conditions for neededness in terms of approximations. Several approximations are defined in Section 3.3. In Section 3.4 we present our framework for decidable call-by-need computations to normal form.

The results presented in this chapter are also published in [DM05].

3.1 Strategies

Given a system and a term, a rewrite strategy specifies which part(s) of the term to evaluate. If a system admits infinite computations, certain rewrite strategies may fail to reduce terms to their normal forms.

Example 3.1.1. *Consider the system $\mathcal{R}$ consisting of the rewrite rules*

$$\begin{aligned} 0 + y &\to y & \mathsf{fib} &\to \mathsf{f}(0, \mathsf{s}(0)) \\ \mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{f}(x, y) &\to x : \mathsf{f}(y, x + y) \\ \mathsf{nth}(0, y : z) &\to y & \mathsf{nth}(\mathsf{s}(x), y : z) &\to \mathsf{nth}(x, z) \end{aligned}$$

for computing Fibonacci numbers. The term $t = \mathsf{nth}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))), \mathsf{fib})$ admits the normal form $\mathsf{s}(\mathsf{s}(0))$:[1]

$$\begin{aligned} t &\to \mathsf{nth}(3, \mathsf{f}(0, 1)) \to \mathsf{nth}(3, 0 : \mathsf{f}(1, 0 + 1)) \to \mathsf{nth}(2, \mathsf{f}(1, 0 + 1)) \\ &\to \mathsf{nth}(2, \mathsf{f}(1, 1)) \to \mathsf{nth}(2, 1 : \mathsf{f}(1, 1 + 1)) \to \mathsf{nth}(1, \mathsf{f}(1, 1 + 1)) \\ &\to \mathsf{nth}(1, \mathsf{f}(1, \mathsf{s}(0 + 1))) \to \mathsf{nth}(1, \mathsf{f}(1, 2)) \to \mathsf{nth}(1, 1 : \mathsf{f}(2, 1 + 2)) \\ &\to \mathsf{nth}(0, \mathsf{f}(2, 1 + 2)) \to \mathsf{nth}(0, \mathsf{f}(2, \mathsf{s}(0 + 2))) \to \mathsf{nth}(0, \mathsf{f}(2, 3)) \\ &\to \mathsf{nth}(0, 2 : \mathsf{f}(3, 2 + 3)) \to 2 \end{aligned}$$

[1] In the rewrite sequences we denote $\mathsf{s}^n(0)$ by n for $n = 1, 2, 3, 5$.

but an eager (innermost) strategy will produce an infinite rewrite sequence:

$$\begin{aligned}
t &\to \mathsf{nth}(3,\mathsf{f}(0,1)) \to \mathsf{nth}(3,0:\mathsf{f}(1,0+1)) \to \mathsf{nth}(3,0:\mathsf{f}(1,1)) \\
&\to \mathsf{nth}(3,0:(1:\mathsf{f}(1,1+1))) \to^2 \mathsf{nth}(3,0:(1:\mathsf{f}(1,2))) \\
&\to \mathsf{nth}(3,0:(1:(1:\mathsf{f}(2,1+2)))) \to^2 \mathsf{nth}(3,0:(1:(1:\mathsf{f}(2,3)))) \\
&\to \mathsf{nth}(3,0:(1:(1:(2:\mathsf{f}(3,2+3))))) \to^3 \mathsf{nth}(3,0:(1:(1:(2:\mathsf{f}(3,5))))) \\
&\to \cdots
\end{aligned}$$

If a term t has a normal form then we can always compute a normal form of t by computing the reducts of t in a breadth-first manner until we encounter a normal form. However, this is a highly inefficient way to compute normal forms. In practice, normal forms are computed by adopting a suitable strategy for selecting the redexes which are to be contracted in each step. A strategy is called *normalizing* if it succeeds in computing normal forms for all terms that admit a normal form. For the class of *orthogonal* systems several normalization results are known (see e.g. Klop [Klo92]). For instance, O'Donnell [O'D77] proved that the *parallel-outermost strategy* (which contracts in a single step all outermost redexes in parallel) is normalizing for all orthogonal systems. However, parallel-outermost is not an *optimal*[2] strategy as it may perform useless steps.

Example 3.1.2. *Consider the system $\mathcal{R}$ consisting of the rewrite rules*

$$\begin{array}{cc}
0+y \to y & 0\times y \to 0 \\
\mathsf{s}(x)+y \to \mathsf{s}(x+y) & \mathsf{s}(x)\times y \to (x\times y)+y
\end{array}$$

Faced with the term $t = (0 \times \mathsf{s}(0)) \times (0 + \mathsf{s}(0))$, the parallel-outermost strategy computes its normal form 0 by contracting three redexes in two steps:

$$\underline{(0\times \mathsf{s}(0))} \times \underline{(0+\mathsf{s}(0))} \between \underline{0\times \mathsf{s}(0)} \to 0$$

The normal form 0 can also be reached by contracting just two redexes:

$$\underline{(0\times \mathsf{s}(0))} \times (0+\mathsf{s}(0)) \to \underline{0\times (0+\mathsf{s}(0))} \to 0$$

So redex $0 + \mathsf{s}(0)$ in t is not needed to reach the normal form.

An optimal strategy selects only needed redexes. Formally, a redex Δ in a term t is needed if in every rewrite sequence from t to normal form a *descendant* of Δ is contracted. The latter concept is defined as follows. Let $A\colon s = s[l\sigma]_p \to s[r\sigma]_p = t$ be a rewrite step in an system and let $q \in \mathcal{P}\mathsf{os}(s)$. The set $q\backslash A$ of descendants of q in t is defined as follows:

$$q\backslash A = \begin{cases} \{q\} & \text{if } q < p \text{ or } q \perp p, \\ \{pp_3p_2 \mid r|_{p_3} = l|_{p_1}\} & \text{if } q = pp_1p_2 \text{ with } p_1 \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(l), \\ \varnothing & \text{otherwise.} \end{cases}$$

[2] An optimal strategy uses the least number of redex contractions to normalize terms.

The concept of descendant extends naturally to rewrite sequences. Orthogonal esystems have the property that descendants of redex positions are again redex positions.

Example 3.1.3. *In the displayed rewrite sequence* $\mathsf{nth}(3,\mathsf{fib}) \to^* 2$ *in Example 3.1.1 non-needed redexes are contracted. For instance, redex* $1+2$ *in the term* $\mathsf{nth}(0,\mathsf{f}(2,1+2))$ *is non-needed:*

$$\mathsf{nth}(0,\underline{\mathsf{f}(2,1+2)}) \to \underline{\mathsf{nth}(0,2:\mathsf{f}(1+2,2+(1+2)))} \to 2$$

The following theorem of Huet and Lévy [HL91] forms the basis of all results on optimal normalizing reduction strategies for orthogonal systems.

Theorem 3.1.4. *Let* $\mathcal{R}$ *be an orthogonal system.*

1. *Every reducible term contains a needed redex.*
2. *Repeated contraction of needed redexes results in a normal form, whenever the term under consideration has a normal form.*

So, for orthogonal systems, the strategy that always selects a needed redex for contraction is normalizing and optimal.[3] Unfortunately, needed redexes are not computable in general. Hence, in order to obtain a *computable* optimal strategy, we need to find (1) decidable approximations of neededness and (2) (decidable) classes of rewrite systems which ensure that every reducible term has a needed redex identified by (1).

In the sequentiality-based approach (see Chapter 9) issue (1) is addressed as follows. Basically, to determine whether an outermost redex Δ in a term $t = C[\Delta]$ is needed, Δ is replaced by a fresh symbol $\bullet$ and all other outermost redexes in t are replaced by Ω which represents an unknown term. It is then investigated whether $\bullet$ can disappear from the resulting Ω-term t' by using some computable notion of partial reduction. If this is not the case, then we may conclude that redex Δ in t is needed. Since neededness of redex Δ in t is solely determined by its position in t (cf. Lemma 3.2.1), replacing redex Δ in t by $\bullet$ induces no loss of generality. However, by replacing all other outermost redexes by Ω, essential information may be lost for determining the neededness of Δ. This is illustrated in the following example, which shows that needed redexes are not independent of other redexes.

Example 3.1.5. *Consider again the system of Example 3.1.2. An arbitrary redex* Δ *is needed in the term* $(0+\mathsf{s}(0))\times\Delta$ *but not in the term* $(0\times\mathsf{s}(0))\times\Delta$*:*

$$\underline{(0\times\mathsf{s}(0))}\times\Delta \to \underline{0\times\Delta} \to 0$$

In the next section we present a new approach to the problem of determining neededness of a given redex in a term which does not abstract from the other redexes in the term.

[3] We ignore here the problem of duplication of (needed) redexes, which can be solved if common subterms are shared.

3.2 Decidable Approximations of Neededness

We assume that the set of ground terms is non-empty. It is undecidable whether a redex in a term is needed with respect to a given (orthogonal) esystem. In this section we present decidable sufficient conditions for a redex to be needed.

We start with an easy lemma that provides an alternative definition of neededness, not depending on the concept of descendant. Consider a system $(\mathcal{R}, \mathcal{F})$. We assume the existence of a constant $\bullet$ not appearing in $\mathcal{F}$. We consider the extended signature $\mathcal{F}_\bullet = \mathcal{F} \cup \{\bullet\}$ and the extended set of redex rules $\mathcal{R}_\bullet = \mathcal{R} \cup \{\bullet \to \bullet\}$ and the resulting system $(\mathcal{R}_\bullet, \mathcal{F}_\bullet)$ that we will call $\mathcal{R}_\bullet$ whenever $\mathcal{F}$ is understood.

Note that $\mathsf{NF}(\mathcal{R}_\bullet, \mathcal{F}_\bullet) = \mathsf{NF}(\mathcal{R}, \mathcal{F})$

Lemma 3.2.1. *Let $(\mathcal{R}, \mathcal{F})$ be an orthogonal esystem. Redex Δ in term $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is needed if and only if there is no term $t \in \mathsf{NF}(\mathcal{R}_\bullet)$ such that $C[\bullet] \to^*_{\mathcal{R}} t$.*

Proof. Let $A\colon s \to^* t$ be a rewrite sequence and Δ a redex in s. We write $\Delta \perp A$ if no descendant of Δ is contracted in A. So a redex Δ in a term s is needed if and only if $A\colon s \to^* t$ with $\Delta \perp A$ implies that t is not a normal form.

For the "only if" direction we suppose there is a term $t \in \mathsf{NF}(\mathcal{R}_\bullet)$ such that $C[\bullet] \to^*_{\mathcal{R}} t$. Replacing every position of $\bullet$ by Δ yields a sequence $A\colon C[\Delta] \to^*_{\mathcal{R}} t$ with $\Delta \perp A$. Hence Δ is not needed.

For the "if" direction we suppose that Δ is not needed. So there exists a rewrite sequence $A\colon C[\Delta] \to^*_{\mathcal{R}} t$ with $t \in \mathsf{NF}(\mathcal{R}_\bullet)$ and $\Delta \perp A$. Replacing every descendant of Δ in A by $\bullet$ yields a sequence $C[\bullet] \to^*_{\mathcal{R}} t$. (Here we use orthogonality. Note that because t is a normal form there are no descendants of Δ in t left.) □

An immediate consequence of this lemma is the folklore result that only the position of a redex in a term is important for determining neededness. So if redex Δ in term $C[\Delta]$ is needed then so is redex Δ' in $C[\Delta']$.

Using the notation introduced p21, the preceding lemma can be rephrased as follows: redex Δ in $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is needed if and only if $C[\bullet] \notin (\to^*_{\mathcal{R}})[\mathsf{NF}(\mathcal{R}_\bullet)]$. Since membership for recognizable languages is decidable but neededness undecidable, it follows that $(\to^*_{\mathcal{R}})[\mathsf{NF}(\mathcal{R}_\bullet)]$ is not recognizable in general. The key to decidability is to extend $\to^*_{\mathcal{R}}$ to $\to^*_{\mathcal{S}}$ for some suitable esystem $\mathcal{S}$ such that $(\to^*_{\mathcal{S}})[\mathsf{NF}(\mathcal{R})]$ becomes recognizable.

Definition 3.2.2. *Let $\mathcal{R}$ and $\mathcal{S}$ be esystems over the same signature. We say that $\mathcal{S}$* approximates *$\mathcal{R}$ if $\to_{\mathcal{R}} \subseteq \to^*_{\mathcal{S}}$ and $\mathsf{LHS}(\mathcal{R}) = \mathsf{LHS}(\mathcal{S})$.*

Definition 3.2.3. *An* approximation mapping *is a mapping α from esystems to esystems with the property that $\alpha(\mathcal{R})$ approximates $\mathcal{R}$ for all esystems $\mathcal{R}$. We write $\mathcal{R}_\alpha$ for $\alpha(\mathcal{R})$. We define a partial order $\leqslant$ on approximation mappings as follows: $\alpha \leqslant \beta$ if and only if $\mathcal{R}_\beta$ approximates $\mathcal{R}_\alpha$, for every esystem $\mathcal{R}$. Note that the identity mapping is the minimum element of this partial order.*

Definition 3.2.4. *We say that an approximation mapping α is* recognizability preserving *if $(\to^*_{\mathcal{R}_\alpha})[L]$ is recognizable for all esystems $\mathcal{R}$ and recognizable L.*

Needless to say, we are only interested in *computable approximation mappings* that are *effectively* recognizability preserving. This means that there is an algorithm which, given a term automaton for L, constructs a term automaton for $(\to^*_{\mathcal{R}_\alpha})[L]$. The recognizability preserving approximation mappings that we introduce in the next section have this property.

Definition 3.2.5. *Let $(\mathcal{R}, \mathcal{F})$ be a system and α an approximation mapping. We say that redex Δ in $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is* α-needed *if $C[\bullet] \notin (\to^*_{\mathcal{R}_\alpha})[\mathsf{NF}(\mathcal{R}_\bullet)]$. The set of all such terms $C[\bullet]$ is denoted by $\mathsf{NEED}(\mathcal{R}_\alpha)$.*

In the following we abbreviate $\to_{\mathcal{R}_\alpha}$ to $\to_\alpha$ when the $\mathcal{R}$ can be inferred from the context.

Lemma 3.2.6. *Let $\mathcal{R}$ be an orthogonal system and α an approximation mapping. Every α-needed redex is needed.*

Proof. Let Δ be an α-needed redex in $C[\Delta]$. So $C[\bullet] \notin (\to^*_{\mathcal{R}_\alpha})[\mathsf{NF}(\mathcal{R}_\bullet)]$. Since $\mathcal{R}_\alpha$ approximates $\mathcal{R}$, we have $\to_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}_\alpha}$ by definition and thus also $\to^*_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}_\alpha}$. Hence $C[\bullet] \notin (\to^*_{\mathcal{R}})[\mathsf{NF}(\mathcal{R}_\bullet)]$. Because $\mathcal{R}$ is orthogonal, we obtain the neededness of Δ from Lemma 3.2.1. □

Only in Lemma 3.2.6 do we require orthogonality. For decidability issues, left-linearity suffices. The following example shows that both left-linearity and non-overlappingness are required for Lemmata 3.2.1 and 3.2.6.

Example 3.2.7. *First of all, consider the left-linear overlapping system consisting of the single rewrite rule*

$$\mathsf{f}(\mathsf{f}(x)) \to \mathsf{a}$$

and the term $\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a})))$. Since contracting either of the two redexes immediately gives a normal form, neither of the two redexes is needed. On the other hand, for any approximation mapping α, including the identity mapping, redex $\mathsf{f}(\mathsf{f}(\mathsf{a}))$ is α-needed since $\bullet$ is an $\mathcal{R}_\alpha$-normal form which does not belong to $\mathsf{NF}(\mathcal{R}_\bullet)$.

Next consider the non-left-linear non-overlapping system consisting of the three rewrite rules

$$\mathsf{f}(x, x) \to \mathsf{a} \qquad \mathsf{b} \to \mathsf{c} \qquad \mathsf{c} \to \mathsf{b}$$

and the term $\mathsf{f}(\mathsf{b}, \mathsf{c})$. Again, it is easy to see that neither of the two redexes is needed. Replacing either of them by $\bullet$ yields a term which, for two of the three approximation mappings α defined in the next section as well as for the identity mapping, does not $\mathcal{R}_\alpha$-rewrite to a normal form in $\mathsf{NF}(\mathcal{R}_\bullet)$.

Lemma 3.2.8. *Let $\mathcal{R}$ be a left-linear system and α an approximation mapping. If α is recognizability preserving then $\mathsf{NEED}(\mathcal{R}_\alpha)$ is recognizable.*

Proof. We have

$$\mathsf{NEED}(\mathcal{R}_\alpha) = (\to^*_{\mathcal{R}_\alpha})[\mathsf{NF}(\mathcal{R}_\bullet)]^c \cap \mathsf{M}_\bullet$$ [4]

[4] Here $(\to^*_{\mathcal{R}_\alpha})[\mathsf{NF}(\mathcal{R}_\bullet)]^c$ denotes the complement of $(\to^*_{\mathcal{R}_\alpha})[\mathsf{NF}(\mathcal{R}_\bullet)]$ (with respect to $\mathcal{T}(\mathcal{F}_\bullet)$).

where $\mathsf{M}_\bullet$ is the subset of $\mathcal{T}(\mathcal{F}_\bullet)$ consisting of all terms that contain exactly one position of $\bullet$. The recognizability of $\mathsf{M}_\bullet$ is easily shown. Hence the recognizability of $\mathsf{NEED}(\mathcal{R}_\alpha)$ is a consequence of Lemmata 2.4.1 and 2.4.2. □

Since membership for recognizable term languages is decidable, we obtain the following result.

Corollary 3.2.9. *Let $\mathcal{R}$ be a left-linear system and α a recognizability preserving approximation mapping. It is decidable whether a redex in a term is α-needed.*

Naturally, a better approximation can identify more needed redexes.

Lemma 3.2.10. *Let α and β be approximation mappings. If $\alpha \leqslant \beta$ then $\mathsf{NEED}(\mathcal{R}_\beta) \subseteq \mathsf{NEED}(\mathcal{R}_\alpha)$, for every system $\mathcal{R}$.*

3.3 Approximations

In this section we define four approximation mappings that are known to be recognizability preserving. We give new proofs for two of these results. The approximations differ in the way they treat the right-handsides of the rewrite rules of the original system. The left-hand sides are not affected, and hence the second requirement in the definition of approximation is trivially satisfied.

Definition 3.3.1. *Let $\mathcal{R}$ be a system. The* strong approximation $\mathcal{R}_\mathsf{s}$ *is obtained from $\mathcal{R}$ by replacing the right-handside of every rewrite rule by a fresh variable.*

Example 3.3.2. *For the system $\mathcal{R}$ of Example 3.1.2, the esystem $\mathcal{R}_\mathsf{s}$ consists of the following rules:*

$$\begin{array}{rr} \mathsf{0} + y \to z & \mathsf{0} \times y \to z \\ \mathsf{s}(x) + y \to z & \mathsf{s}(x) \times y \to z \end{array}$$

The idea of approximating a system by ignoring the right-handsides of its rewrite rules is due to Huet and Lévy [HL91]. Reduction with the strong approximation of a system replaces redexes by arbitrary terms and is sometimes called *arbitrary reduction*. A better approximation is obtained by preserving the *n*on-*v*ariable parts of the right-handsides of the rewrite rules.

Definition 3.3.3. *Let $\mathcal{R}$ be a system. The* nv approximation $\mathcal{R}_\mathsf{nv}$ *is obtained from $\mathcal{R}$ by replacing all positions of variables in the right-hand side of every rewrite rule by distinct fresh variables.*

Example 3.3.4. *For the system $\mathcal{R}$ of Example 3.1.2, the esystem $\mathcal{R}_\mathsf{nv}$ consists of the following rules:*

$$\begin{array}{rr} \mathsf{0} + y \to y' & \mathsf{0} \times y \to \mathsf{0} \\ \mathsf{s}(x) + y \to \mathsf{s}(x' + y') & \mathsf{s}(x) \times y \to (x' \times y') + y'' \end{array}$$

The idea of approximating a system by ignoring the variables in the right-handsides of the rewrite rules is due to Oyamaguchi [Oya93]. Note that $\mathcal{R}_{\mathsf{nv}} = \mathcal{R}$ whenever $\mathcal{R}$ is right-ground. Hence for every orthogonal right-ground system $\mathcal{R}$, a redex is needed if and only if it is nv-needed.

Definition 3.3.5. *An esystem is called* growing *if for every rewrite rule $l \to r$ the variables in $\mathcal{V}\mathsf{ar}(l) \cap \mathcal{V}\mathsf{ar}(r)$ occur at depth 1 in l. Let $\mathcal{R}$ be a system. The* growing approximation $\mathcal{R}_{\mathsf{g}}$ *is defined as the growing esystem that is obtained from $\mathcal{R}$ by renaming the variables in the right-handsides that occur at a depth greater than 1 in the corresponding left-hand sides.*

Example 3.3.6. *For the system $\mathcal{R}$ of Example 3.1.2, the esystem $\mathcal{R}_{\mathsf{g}}$ consists of the following rules:*

$$\begin{array}{cc} \mathsf{0} + y \to y & \mathsf{0} \times y \to \mathsf{0} \\ \mathsf{s}(x) + y \to \mathsf{s}(x' + y) & \mathsf{s}(x) \times y \to (x' \times y) + y \end{array}$$

Note that the positions of y in the right-handsides of the rules of $\mathcal{R}$ are not renamed since they occur at depth 1 in the corresponding left-hand sides.

The growing approximation defined above stems from Nagaya and Toyama [NT02]. It extends Jacquemard's linear-growing approximation [Jac96b] which imposes a right-linearity requirement.

Definition 3.3.7. *An esystem is called* linear-growing *if it is growing and linear. The* linear-growing approximation $\mathcal{R}_{\mathsf{lg}}$ *can be obtained from $\mathcal{R}_{\mathsf{g}}$ by renaming in the right-handsides the variables causing non-linearity in a left to right manner. Note that other convention could be taken in order to define uniquely $\mathcal{R}_{\mathsf{lg}}$.*

Linear-growing systems introduced by Jacquemard [Jac96b], are a proper extension of the shallow systems considered by Comon [Com00].

Example 3.3.8. *For the system $\mathcal{R}$ of Example 3.1.2, the esystem $\mathcal{R}_{\mathsf{lg}}$ consists of the following rules:*

$$\begin{array}{cc} \mathsf{0} + y \to y & \mathsf{0} \times y \to \mathsf{0} \\ \mathsf{s}(x) + y \to \mathsf{s}(x' + y) & \mathsf{s}(x) \times y \to (x' \times y) + y' \end{array}$$

Note that the second occurrence of y in the right-handside of the fourth rule has been renamed to achieve linearity.

The mapping s that assigns to every esystem $\mathcal{R}$ the esystem $\mathcal{R}_{\mathsf{s}}$ is an approximation mapping. In the same fashion, Definitions 3.3.3, 3.3.7 and 3.3.5 define approximation mappings nv, lg and g. We clearly have $\mathsf{id} \leqslant \mathsf{g} \leqslant \mathsf{lg} \leqslant \mathsf{nv} \leqslant \mathsf{s}$ where id is the identity mapping.

Example 3.3.9. *Consider again the system $\mathcal{R}$ of Example 3.1.2. Let Δ_1 and Δ_2 be arbitrary redexes and consider the term*

$$t = (\underbrace{\mathsf{0} + \mathsf{s}(\Delta_1)}_{\Delta_3}) + \Delta_2$$

All three redexes are needed (since the rules of $\mathcal{R}$ involved are non-erasing). The following rewrite sequences show that Δ_1 and Δ_2 are not s*-needed:*

$$(0 + \mathsf{s}(\bullet)) + \Delta_2 \to_{\mathsf{s}} 0 + \Delta_2 \to_{\mathsf{s}} 0$$
$$(0 + \mathsf{s}(\Delta_1)) + \bullet \to_{\mathsf{s}} 0 + \bullet \to_{\mathsf{s}} 0$$

Redex Δ_3 is s*-needed since all* s*-reducts of $\bullet + \Delta_2$ are of the form $\bullet + t'$. For the* nv *approximation the situation is the same. Redexes Δ_1 and Δ_2 are not* nv*-needed—the above* s*-rewrite sequences are also* nv*-rewrite sequences—but Δ_3 is. With respect to the growing approximation, Δ_1 is not* g*-needed:*

$$(0 + \mathsf{s}(\bullet)) + \Delta_2 \to_{\mathsf{g}} \mathsf{s}(\bullet) + \Delta_2 \to_{\mathsf{g}} \mathsf{s}(0 + \Delta_2) \to_{\mathsf{g}} \mathsf{s}(\Delta_2) \to_{\mathsf{g}}^* t'$$

for some normal form t' (which depends on redex Δ_2). However, Δ_2 is g*-needed. The reason is that we cannot get rid of $\bullet$ in the term $(0 + \mathsf{s}(\Delta_1)) + \bullet$ since the second argument of $+$ is never erased by the rules in $\mathcal{R}_{\mathsf{g}}$.*

Theorem 3.3.10. *The approximation mappings* s*,* nv*,* lg *and* g *are recognizability preserving.*

Nagaya and Toyama [NT02] proved the above result for the growing approximation; the term automaton that recognizes $(\to_{\mathsf{g}}^*)[L]$ is defined as the limit of a finite saturation process(see Section 5.2.2. This saturation process is similar to the ones defined in Comon [Com00] and Jacquemard [Jac96b] (see Section 5.2.1). However, by working exclusively with deterministic term automata, Nagaya and Toyama can handle non-right-linear rewrite rules. The construction of these automata is detailed in Chapter 5 as they are useful for the complexity study.

Below we give a very simple proof of Theorem 3.3.10 for the s and nv approximations, using ground term transducers.

Lemma 3.3.11. *Let $\mathcal{R}$ be a left-linear system. The relations $\to_{\mathsf{s}}^*$ and $\to_{\mathsf{nv}}^*$ are gtt-recognizable.*

Proof. According to Lemma 2.4.3(1) gtt-recognizable relations are closed under transitive closure. Since $\Vdash^+ = \to^*$ it therefore suffices to show that $\Vdash_{\mathsf{s}}$ and $\Vdash_{\mathsf{nv}}$ are gtt-recognizable. First we show the gtt-recognizability of $\Vdash_{\mathsf{nv}}$. Let $\mathcal{R}_{\mathsf{nv}} = \{l_i \to r_i \mid 1 \leqslant i \leqslant n\}$. Define the ground term transducer $\mathcal{G}_{\mathsf{nv}}$ as the pair of term automata $\mathcal{A}$ and $\mathcal{B}$ that accept in state i all instances of l_i and r_i, respectively. Moreover, we may assume that the two term automata share no other states. Hence $L(\mathcal{G}_{\mathsf{nv}}) = \Vdash_{\mathsf{nv}}$. The gtt-recognizability of $\Vdash_{\mathsf{s}}$ is obtained by replacing $\mathcal{B}$ by the term automaton $\mathcal{C}$ that accepts in state i all terms. □

We illustrate the construction of $\mathcal{G}_{\mathsf{nv}}$ and $\mathcal{G}_{\mathsf{s}}$ in the proof of the above lemma on a small example.

Example 3.3.12. *Table 3.1 shows the term automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ used in the proof of the above lemma for the following system $\mathcal{R}$:*

$$\begin{array}{lrcl} 1\colon & \mathsf{f}(\mathsf{g}(x), \mathsf{a}) & \to & \mathsf{f}(\mathsf{h}(\mathsf{h}(x)), x) \\ 2\colon & \mathsf{h}(\mathsf{a}) & \to & \mathsf{h}(\mathsf{b}) \\ 3\colon & \mathsf{h}(\mathsf{f}(x, \mathsf{b})) & \to & x \end{array}$$

Table 3.1: The term automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ in the proof of Lemma 3.3.11.

$\mathsf{a} \to [x]$	$\mathsf{a} \to \{x\}$	$\mathsf{a} \to \{x\}$
$\mathsf{b} \to [x]$	$\mathsf{b} \to \{x\}$	$\mathsf{b} \to \{x\}$
$\mathsf{f}([x],[x]) \to [x]$	$\mathsf{f}(\{x\},\{x\}) \to \{x\}$	$\mathsf{f}(\{x\},\{x\}) \to \{x\}$
$\mathsf{g}([x]) \to [x]$	$\mathsf{g}(\{x\}) \to \{x\}$	$\mathsf{g}(\{x\}) \to \{x\}$
$\mathsf{h}([x]) \to [x]$	$\mathsf{h}(\{x\}) \to \{x\}$	$\mathsf{h}(\{x\}) \to \{x\}$
$\bullet \to [x]$	$\bullet \to \{x\}$	$\bullet \to \{x\}$
$\mathsf{a} \to [\mathsf{a}]$	$\mathsf{b} \to \{\mathsf{b}\}$	
$\mathsf{b} \to [\mathsf{b}]$	$\mathsf{h}(\{x\}) \to \{\mathsf{h}(x)\}$	
$\mathsf{g}([x]) \to [\mathsf{g}(x)]$	$\mathsf{h}(\{\mathsf{h}(x)\}) \to \{\mathsf{h}(\mathsf{h}(x))\}$	
$\mathsf{f}([x],[\mathsf{b}]) \to [\mathsf{f}(x,\mathsf{b})]$		
$\mathsf{f}([\mathsf{g}(x)],[\mathsf{a}]) \to \{1\}$	$\mathsf{f}(\{\mathsf{h}(\mathsf{h}(x))\},\{x\}) \to \{1\}$	$\{x\} \to \{1\}$
$\mathsf{h}([\mathsf{a}]) \to \{2\}$	$\mathsf{h}(\{\mathsf{b}\}) \to \{2\}$	$\{x\} \to \{2\}$
$\mathsf{h}([\mathsf{f}(x,\mathsf{b})]) \to \{3\}$	$\{x\} \to \{3\}$	$\{x\} \to \{3\}$

Note that only states $\{1\}$, $\{2\}$, *and* $\{3\}$ *are shared by* $\mathcal{A}$ *and* $\mathcal{B}$ *and by* $\mathcal{A}$ *and* $\mathcal{C}$. *Consider the term automaton* $\mathcal{A}$. *Its states are* $[x]$, $[\mathsf{a}]$, $[\mathsf{b}]$, $[g(x)]$, *and* $[f(x,b)]$. *In state* $[x]$ *all ground terms are accepted. The purpose of the second group of transition rules is to recognize all ground instances of proper non-variable subterms of the left-hand sides of* $\mathcal{R}$. *So in state* $[\mathsf{a}]$ *only the term* a *is accepted, whereas in state* $[f(x,b)]$ *all ground terms of the form* $\mathsf{f}(t,\mathsf{b})$ *are accepted. The third group of transition rules corresponds to the left-hand sides of* $\mathcal{R}$.

The recognizability preservation of s and nv is an immediate consequence of Lemmata 3.3.11 and 2.4.3(2). (Since $\to_{\mathsf{g}}^*$ need not be a gtt-recognizable relation,[5] ground term transducers are not useful for obtaining the recognizability preservation of g.)

It is easy to see that s-needed redexes in a term are always outermost. The same is true for nv-needed redexes in terms that have a normal form. However, g-needed redexes in normalizing terms need not be outermost. For instance, the system $\mathcal{R}$:

$$\mathsf{f}(x) \to \mathsf{g}(x) \qquad\qquad \mathsf{a} \to \mathsf{b}$$

is growing and hence $\mathcal{R}_{\mathsf{g}} = \mathcal{R}$. Innermost redex a in the term $\mathsf{f}(\mathsf{a})$ is g-needed because there is no term $t \in \mathsf{NF}(\mathcal{R}_\bullet)$ such that $\mathsf{f}(\bullet) \to_{\mathcal{R}}^* t$. Note that a is not nv-needed as $\mathsf{f}(\bullet) \to_{\mathsf{nv}} \mathsf{g}(\mathsf{b})$ with $\mathsf{g}(\mathsf{b}) \in \mathsf{NF}(\mathcal{R}_\bullet)$.

Takai *et al.* [TKS00] introduced the class of left-linear inverse finite path overlapping rewrite systems and showed that Theorem 3.3.10 is true for the corresponding approximation mapping. Growing rewrite systems constitute a proper subclass of the class of inverse finite path overlapping rewrite systems.

[5] It is not difficult to show that $\to_{\mathsf{g}}^*$ is not gtt-recognizable for the system $\mathcal{R} = \{\mathsf{f}(x) \to x\}$ over the signature consisting of unary function symbols f and g, and a constant a.

Since the definition of this class is rather difficult, we do not consider the inverse finite path overlapping approximation here. We note however that our results easily extend. Another complicated recognizability preserving approximation mapping can be extracted from the paper by Seki *et al.* [STFK02].

3.4 Call-by-Need Computations to Normal Form

A system $\mathcal{R}$ admits decidable call-by-need computations to normal form if there exists an approximation mapping α such that α-needed redexes are computable and, moreover, every reducible term has an α-needed redex. In Section 3.2 we addressed the first issue. This section is devoted to the second issue. The following definition is readily understood.

Definition 3.4.1. *Let α be an approximation mapping. The class of systems $(\mathcal{R}, \mathcal{F})$ such that every reducible term in $\mathcal{T}(\mathcal{F})$ has an α-needed redex is denoted by CBN_α.*

Lemma 3.4.2. *Let $\mathcal{R}$ be an orthogonal system.*

1. *If $\mathcal{R}$ is right-ground then $\mathcal{R} \in \mathsf{CBN}_{\mathsf{nv}}$.*
2. *If $\mathcal{R}$ is linear-growing then $\mathcal{R} \in \mathsf{CBN}_{\mathsf{lg}}$.*
3. *If $\mathcal{R}$ is growing then $\mathcal{R} \in \mathsf{CBN}_{\mathsf{g}}$.*

Proof. According to Theorem 3.1.4(1) every reducible term contains a needed redex. If $\mathcal{R}$ is right-ground then $\mathcal{R} = \mathcal{R}_{\mathsf{nv}}$ and thus all needed redexes are nv-needed. Hence $\mathcal{R} \in \mathsf{CBN}_{\mathsf{nv}}$. If $\mathcal{R}$ is linear-growing then $\mathcal{R} = \mathcal{R}_{\mathsf{lg}}$ and thus all needed redexes are lg-needed. Hence $\mathcal{R} \in \mathsf{CBN}_{\mathsf{lg}}$. The same argument applies to g. □

The next lemma is an easy consequence of Lemma 3.2.10.

Lemma 3.4.3. *Let α and β be approximation mappings. If $\alpha \leqslant \beta$ then $\mathsf{CBN}_\beta \subseteq \mathsf{CBN}_\alpha$.*

Proof. Let $\mathcal{R}$ be a system over a signature $\mathcal{F}$ that belongs to CBN_β. So every reducible term t in $\mathcal{T}(\mathcal{F})$ has a β-needed redex. So $t = C[\Delta]$ with Δ a β-needed redex. By definition $C[\bullet] \in \mathsf{NEED}(\mathcal{R}_\beta)$. Lemma 3.2.10 yields $C[\bullet] \in \mathsf{NEED}(\mathcal{R}_\alpha)$. Hence redex Δ is α-needed in t. It follows that $\mathcal{R}$ belongs to CBN_α. □

Below we show that membership of a left-linear system in CBN_α is decidable for any recognizability preserving approximation mapping α. The proof is a straightforward consequence of the following result.

Theorem 3.4.4. *Let $\mathcal{R}$ be a left-linear system and let α be a recognizability preserving approximation mapping. The set of terms that have an α-needed redex is recognizable.*

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R}$. Define the relation $\mathsf{mark}^{\bullet}_{\mathcal{R}}$ on $\mathcal{T}(\mathcal{F}_{\bullet})$ as the parallel closure of $\{(\Delta, \bullet) \mid \Delta \in \mathcal{T}(\mathcal{F}) \text{ is a redex}\}$. The set of terms that have an α-needed redex coincides with

$$\mathsf{mark}^{\bullet}_{\mathcal{R}}[\mathsf{NEED}(\mathcal{R}_\alpha)] \cap \mathcal{T}(\mathcal{F})$$

If we can show that the relation $\mathsf{mark}^{\bullet}_{\mathcal{R}}$ is gtt-recognizable then the result follows from Lemmata 2.4.1, 2.4.3(2), and 3.2.8. Let $\mathcal{A}$ be a term automaton with a unique final state ! that accepts $\mathsf{REDEX}(\mathcal{R}) \cap \mathcal{T}(\mathcal{F})$ and let $\bullet \to !$ be the single transition rule of the term automaton $\mathcal{B}$. It is not difficult to see that the ground term transducer $(\mathcal{A}, \mathcal{B})$ accepts $\mathsf{mark}^{\bullet}_{\mathcal{R}}$. □

Theorem 3.4.5. *Let $\mathcal{R}$ be a left-linear system and let α be a recognizability preserving approximation mapping. It is decidable whether $\mathcal{R} \in \mathsf{CBN}_\alpha$.*

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R}$. The system $\mathcal{R}$ belongs to CBN_α if and only if the set

$$A = \mathsf{NF}(\mathcal{R})^c \setminus \{t \in \mathcal{T}(\mathcal{F}) \mid t \text{ has an } \alpha\text{-needed redex}\}$$

is empty. According to Lemmata 2.4.1, 2.4.2 and Theorem 3.4.4, A is recognizable. Hence the emptiness of A is decidable by Lemma 2.4.1. □

Because $\mathcal{R}_\alpha$-needed redexes need not be needed for a left-linear system $\mathcal{R}$ (Example 3.2.7), membership in CBN_α does not guarantee that $\mathcal{R}$ admits a computable call-by-need strategy; orthogonality is needed to draw that conclusion.

It should not come as a surprise that a better approximation covers a larger class of systems. This is expressed formally in the next lemma.

Lemma 3.4.6. *We have $\mathsf{CBN}_\mathsf{s} \subsetneq \mathsf{CBN}_\mathsf{nv} \subsetneq \mathsf{CBN}_\mathsf{lg} \subsetneq \mathsf{CBN}_\mathsf{g}$, even when these classes are restricted to orthogonal systems.*

Proof. From Lemma 3.4.3 we obtain $\mathsf{CBN}_\mathsf{s} \subseteq \mathsf{CBN}_\mathsf{nv} \subseteq \mathsf{CBN}_\mathsf{lg} \subseteq \mathsf{CBN}_\mathsf{g}$. Consider the orthogonal systems

$$\begin{array}{lllll}
\mathcal{R}_a: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{a} & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{b} & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to \mathsf{c} \\
\mathcal{R}_b: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{a} & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{b} & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to x \\
\mathcal{R}_c: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{a} & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{a} & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to \mathsf{a} \\
& \mathsf{f}(\mathsf{b},\mathsf{b},\mathsf{b}) \to \mathsf{b} & & \mathsf{h}(x) \to \mathsf{f}(x,x,x)
\end{array}$$

According to Lemma 3.4.2 $\mathcal{R}_a \in \mathsf{CBN}_\mathsf{nv}$, $\mathcal{R}_b \in \mathsf{CBN}_\mathsf{lg}$ and $\mathcal{R}_c \in \mathsf{CBN}_\mathsf{g}$. So it remains to show that $\mathcal{R}_a \notin \mathsf{CBN}_\mathsf{s}$, $\mathcal{R}_b \notin \mathsf{CBN}_\mathsf{nv}$ and $\mathcal{R}_c \notin \mathsf{CBN}_\mathsf{lg}$. We have

$$\begin{array}{lllll}
(\mathcal{R}_a)_\mathsf{s}: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to y & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to y & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to y \\
(\mathcal{R}_b)_\mathsf{nv}: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{a} & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{b} & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to y \\
(\mathcal{R}_c)_\mathsf{lg}: & \mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{a} & \mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{a} & \mathsf{f}(x,\mathsf{a},\mathsf{b}) \to \mathsf{a} \\
& \mathsf{f}(\mathsf{b},\mathsf{b},\mathsf{b}) \to \mathsf{b} & & \mathsf{h}(x) \to \mathsf{f}(x,y,z)
\end{array}$$

Let Δ be the redex $\mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{b})$. In $(\mathcal{R}_a)_{\mathsf{s}}$ and $(\mathcal{R}_b)_{\mathsf{nv}}$ we have $\Delta \to t$ for every term t. The following rewrite sequences in $(\mathcal{R}_a)_{\mathsf{s}}$ show that none of the redexes in $\mathsf{f}(\Delta,\Delta,\Delta)$ is s-needed:

$$\mathsf{f}(\bullet,\Delta,\Delta) \to \mathsf{f}(\bullet,\mathsf{a},\Delta) \to \mathsf{f}(\bullet,\mathsf{a},\mathsf{b}) \to \mathsf{a}$$
$$\mathsf{f}(\Delta,\bullet,\Delta) \to \mathsf{f}(\mathsf{b},\bullet,\Delta) \to \mathsf{f}(\mathsf{b},\bullet,\mathsf{a}) \to \mathsf{a}$$
$$\mathsf{f}(\Delta,\Delta,\bullet) \to \mathsf{f}(\mathsf{a},\Delta,\bullet) \to \mathsf{f}(\mathsf{a},\mathsf{b},\bullet) \to \mathsf{a}$$

Hence $\mathcal{R}_a \notin \mathsf{CBN}_{\mathsf{s}}$. The following rewrite sequences in $(\mathcal{R}_b)_{\mathsf{nv}}$ show that none of the redexes in $\mathsf{f}(\Delta,\Delta,\Delta)$ is nv-needed:

$$\mathsf{f}(\bullet,\Delta,\Delta) \to \mathsf{f}(\bullet,\mathsf{a},\Delta) \to \mathsf{f}(\bullet,\mathsf{a},\mathsf{b}) \to \mathsf{a}$$
$$\mathsf{f}(\Delta,\bullet,\Delta) \to \mathsf{f}(\mathsf{b},\bullet,\Delta) \to \mathsf{f}(\mathsf{b},\bullet,\mathsf{a}) \to \mathsf{b}$$
$$\mathsf{f}(\Delta,\Delta,\bullet) \to \mathsf{f}(\mathsf{a},\Delta,\bullet) \to \mathsf{f}(\mathsf{a},\mathsf{b},\bullet) \to \mathsf{a}$$

Consequently, $\mathcal{R}_b \notin \mathsf{CBN}_{\mathsf{nv}}$. Now let Δ be the redex $\mathsf{f}(\mathsf{h}(\mathsf{b}),\mathsf{h}(\mathsf{b}),\mathsf{h}(\mathsf{b}))$. Δ can lg-reduce do both a and b. Note that would not be the case if we used g. The following rewrite sequences in $(\mathcal{R}_c)_{\mathsf{lg}}$ show that none of the redexes in $\mathsf{f}(\Delta,\Delta,\Delta)$ is lg-needed:

$$\mathsf{f}(\bullet,\Delta,\Delta) \to^* \mathsf{f}(\bullet,\mathsf{a},\mathsf{b}) \to \mathsf{a}$$
$$\mathsf{f}(\Delta,\bullet,\Delta) \to^* \mathsf{f}(\mathsf{b},\bullet,\mathsf{a}) \to \mathsf{b}$$
$$\mathsf{f}(\Delta,\Delta,\bullet) \to^* \mathsf{f}(\mathsf{a},\mathsf{b},\bullet) \to \mathsf{a}$$

Consequently, $\mathcal{R}_c \notin \mathsf{CBN}_{\mathsf{lg}}$. □

A reducible term without $(\mathcal{R}_\alpha,\mathcal{F})$-needed redexes is called $(\mathcal{R}_\alpha,\mathcal{F})$-*free*. A *minimal* free term has the property that none of its proper subterms is free.

Chapter 4

Signature extension and Modularity

Comon [Com00] showed that strong sequentiality of a left-linear rewrite system can be decided in exponential time. That result also applies to $\mathsf{CBN}_{\mathsf{s}}$. For classes higher in the hierarchy the known upper bounds range from double ($\mathsf{CBN}_{\mathsf{nv}}, \mathsf{CBN}_{\mathsf{lg}}$) to triple exponential ($\mathsf{CBN}_{\mathsf{g}}$) (see Chapter 5, [DM98]). Consequently, it is of obvious importance to have results available that enable to split a rewrite system into smaller components such that membership in CBN_α of the components implies membership of the original system in CBN_α.

Such *modularity* results have been extensively studied for basic properties like confluence and termination, see [Ohl02] for a recent overview. The simplest kind of modularity results are concerned with enriching the signature. Most properties of rewrite systems are preserved under *signature extension*. Two notable exceptions are the normal form property and the unique normal form property (with respect to reduction), see Kennaway *et al.* [KKSdV96]. Also some properties dealing with ground terms are not preserved under signature extension. Consider for instance the property that every ground term is innermost terminating, the rewrite system consisting of the two rewrite rules $\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{f}(x))$ and $\mathsf{f}(\mathsf{a}) \to \mathsf{a}$, and add a new constant b. It turns out that for no α, membership in CBN_α is preserved under signature extension. We present several sufficient conditions which guarantee preservation under signature extension. The study of preservation of membership to CBN_α under signature extension was a necessary step towards modularity. We next consider modularity for systems over disjoint signatures and finally constructor-sharing combinations.

The remainder of this chapter is organized as follows. In Section 4.1 we present our signature extension results and in Section 4.2 these results are extended to modularity. The proofs of most of the results in these two sections are given in Appendix B. The results presented in this chapter are also published in [DM05].

4.1 Signature Extension

In this section we study the question whether membership in CBN_α is preserved after adding new function symbols. This entails that we need to be a bit more precise about the underlying signature in our notation. From now on we write $\mathsf{NF}(\mathcal{R},\mathcal{F})$ for the set of ground normal forms of an esystem $\mathcal{R}$ over a signature $\mathcal{F}$. Furthermore, an α-needed redex with respect to a system $\mathcal{R}$ over the signature $\mathcal{F}$ will often be called $(\mathcal{R}_\alpha,\mathcal{F})$-needed in the sequel.

Many of the examples presented in this and the next section have been verified by `Autowrite`. This tool, described in Chapter 15, checks membership in CBN_α for $\alpha \in \{\mathsf{s},\mathsf{nv},\mathsf{g}\}$ by using the direct term automata constructions described in Chapter 5 as opposed to the ground term transducer constructions of Sections 3.3 and 3.4 which work only up to the nv-case.

Definition 4.1.1. *We say that a class $\mathcal{C}$ of systems is* preserved under signature extension *if $(\mathcal{R},\mathcal{G}) \in \mathcal{C}$ for all $(\mathcal{R},\mathcal{F}) \in \mathcal{C}$ and $\mathcal{F} \subseteq \mathcal{G}$.*

Our first example shows that CBN_s is not preserved under signature extension.

Example 4.1.2. *Consider the system $\mathcal{R} = \mathcal{R}_5$ of Example A.0.5 over the signature $\mathcal{F}$ consisting of all the symbols appearing in the rewrite rules. Let $\mathcal{G} = \mathcal{F} \cup \{@\}$ with $@$ a fresh constant. We have $(\mathcal{R},\mathcal{G}) \notin \mathsf{CBN}_\mathsf{s}$ as the term $\mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{a})$ has no $(\mathcal{R}_\mathsf{s},\mathcal{G})$-needed redex:*

$$\begin{aligned}
&\mathsf{f}(\bullet,\mathsf{a},\mathsf{a}) \to_\mathsf{s} \mathsf{f}(\bullet,\mathsf{g}(\mathsf{a}),\mathsf{a}) \to_\mathsf{s} \mathsf{f}(\bullet,\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a})) \to_\mathsf{s} @\\
&\mathsf{f}(\mathsf{a},\bullet,\mathsf{a}) \to_\mathsf{s} \mathsf{f}(\mathsf{h}(\mathsf{a}),\bullet,\mathsf{a}) \to_\mathsf{s} \mathsf{f}(\mathsf{h}(\mathsf{a}),\bullet,\mathsf{g}(\mathsf{a})) \to_\mathsf{s} @\\
&\mathsf{f}(\mathsf{a},\mathsf{a},\bullet) \to_\mathsf{s} \mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{a},\bullet) \to_\mathsf{s} \mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a}),\bullet) \to_\mathsf{s} @
\end{aligned}$$

One may wonder whether there are any nontrivial counterexamples, where nontrivial means that the set of ground normal forms is non-empty. Surprisingly, the answer is yes, provided we consider an approximation mapping α that is at least as good as nv.

Example 4.1.3. *Consider the system $\mathcal{R}$*

$$\begin{array}{lll}
\mathsf{f}(x,\mathsf{a},\mathsf{b}) \to \mathsf{g}(x) & \mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{a}) \to \mathsf{g}(\mathsf{a}) & \mathsf{g}(\mathsf{a}) \to \mathsf{g}(\mathsf{a})\\
\mathsf{f}(\mathsf{b},x,\mathsf{a}) \to \mathsf{g}(x) & \mathsf{f}(\mathsf{b},\mathsf{b},\mathsf{b}) \to \mathsf{g}(\mathsf{a}) & \mathsf{g}(\mathsf{b}) \to \mathsf{g}(\mathsf{b})\\
\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{g}(x) & \mathsf{e}(x) \to x &
\end{array}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules. First we show that $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_\mathsf{nv}$. It is not difficult to show that the only $(\mathcal{R}_\mathsf{nv},\mathcal{F})$-normalizable terms are a, b, and $\mathsf{e}(t)$ for every $t \in \mathcal{T}(\mathcal{F})$. Since a and b are normal forms, we only have to show that every $\mathsf{e}(t)$ contains an $(\mathcal{R}_\mathsf{nv},\mathcal{F})$-needed redex, which is easy since $\mathsf{e}(t)$ itself is an $(\mathcal{R}_\mathsf{nv},\mathcal{F})$-needed redex. Let $\mathcal{G} = \mathcal{F} \cup \{@\}$ with $@$ a constant. We have $(\mathcal{R},\mathcal{G}) \notin \mathsf{CBN}_\mathsf{nv}$ as the term

$\mathsf{f}(\mathsf{e}(\mathsf{a}),\mathsf{e}(\mathsf{a}),\mathsf{e}(\mathsf{a}))$ *has no* $(\mathcal{R}_{\mathsf{nv}},\mathcal{G})$*-needed redex:*

$$\begin{aligned}
&\mathsf{f}(\bullet,\mathsf{e}(\mathsf{a}),\mathsf{e}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\bullet,\mathsf{a},\mathsf{e}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\bullet,\mathsf{a},\mathsf{b}) \to_{\mathsf{nv}} \mathsf{g}(@)\\
&\mathsf{f}(\mathsf{e}(\mathsf{a}),\bullet,\mathsf{e}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{b},\bullet,\mathsf{e}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{b},\bullet,\mathsf{a}) \to_{\mathsf{nv}} \mathsf{g}(@)\\
&\mathsf{f}(\mathsf{e}(\mathsf{a}),\mathsf{e}(\mathsf{a}),\bullet) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{a},\mathsf{e}(\mathsf{a}),\bullet) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{a},\mathsf{b},\bullet) \to_{\mathsf{nv}} \mathsf{g}(@)
\end{aligned}$$

For $\alpha = \mathsf{s}$ there is no nontrivial counterexample.

Theorem 4.1.4. *The subclass of* $\mathsf{CBN}_{\mathsf{s}}$ *consisting of all orthogonal systems* $(\mathcal{R},\mathcal{F})$ *such that* $\mathsf{NF}(\mathcal{R},\mathcal{F}) \neq \varnothing$ *is preserved under signature extension.*

We refrain from giving the proof at this point since the statement easily follows from Theorem 4.1.9 below, whose proof is presented in detail in the Appendix B (see also the discussion following Corollary 4.2.5). We just show the necessity of the orthogonality condition.

Example 4.1.5. *Consider the left-linear (overlapping) system* $\mathcal{R}$

$$\begin{aligned}
&\mathsf{f}(x,\mathsf{a}) \to \mathsf{a} && \mathsf{g}(\mathsf{f}(\mathsf{a},x),y) \to \mathsf{a}\\
&\mathsf{g}(x,\mathsf{a}) \to \mathsf{a} && \mathsf{g}(x,\mathsf{f}(y,z)) \to \mathsf{a}\\
& && \mathsf{g}(x,\mathsf{g}(y,z)) \to \mathsf{a}
\end{aligned}$$

over the signature $\mathcal{F} = \{\mathsf{a},\mathsf{f},\mathsf{g}\}$.
`Autowrite` *is able to verify that* $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_{\mathsf{s}}$. *Let* $\mathcal{G} = \mathcal{F} \cup \{@\}$ *with* $@$ *a constant. The system* $(\mathcal{R},\mathcal{G})$ *does not belong to* $\mathsf{CBN}_{\mathsf{s}}$ *because the term* $\mathsf{g}(\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{f}(\mathsf{a},\mathsf{a})),@)$ *lacks* $(\mathcal{R}_{\mathsf{s}},\mathcal{G})$*-needed redexes:*

$$\begin{aligned}
&\mathsf{g}(\mathsf{f}(\bullet,\mathsf{f}(\mathsf{a},\mathsf{a})),@) \to_{\mathsf{s}} \mathsf{g}(\mathsf{f}(\bullet,\mathsf{a}),@) \to_{\mathsf{s}} \mathsf{g}(\mathsf{a},@)\\
&\mathsf{g}(\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\bullet),@) \to_{\mathsf{s}} \mathsf{g}(\mathsf{f}(\mathsf{a},\bullet),@) \to_{\mathsf{s}} \mathsf{a}
\end{aligned}$$

Note that here only the rewrite rules $\mathsf{f}(x,\mathsf{a}) \to \mathsf{a}$ *and* $\mathsf{g}(\mathsf{f}(\mathsf{a},x),y) \to \mathsf{a}$ *are used. The remaining rules of* $\mathcal{R}$ *are needed to ensure that* $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_{\mathsf{s}}$.

Our second result states that for any approximation mapping α the subclass of CBN_{α} consisting of all left-linear systems $\mathcal{R}$ with the property defined below is preserved under signature extension.

Definition 4.1.6. *We say that a system* $\mathcal{R}$ *has an* external normal form *if there exists a ground normal form which is not an instance of a proper non-variable subterm of a left-hand sides of a rewrite rule in* $\mathcal{R}$.

Note that the system of Example 4.1.3 lacks external normal forms as both ground normal forms a and b appear in the left-hand sides of the rewrite rules. Further note that it is decidable whether a left-linear system has external normal forms by straightforward term automata techniques. Finally note that the external normal form property is satisfied whenever there exists a constant not occurring in the left-hand sides of the rewrite rules.

Theorem 4.1.7. *Let α be an approximation mapping. The subclass of CBN_α consisting of all left-linear systems with external normal forms is preserved under signature extension.*

The proof is given in Appendix B. Note that for $\alpha = \mathsf{s}$ the above theorem is a special case of Theorem 4.1.4 since the existence of an external normal form implies the existence of a ground normal form.

Our final signature extension result is about systems without external normal form. Such systems are quite common.

Example 4.1.8. *Consider the system $\mathcal{R}$ of Example 3.1.2 (p27) over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules. Since every ground normal form is of the form $\mathsf{s}^n(\mathsf{0})$ for some $n \geqslant 0$, it follows that $\mathcal{R}$ lacks external normal forms.*

The condition $\mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ in Theorem 4.1.9 expresses that the set of $\mathcal{R}_\alpha$-normalizable terms in $\mathcal{T}(\mathcal{F})$ is not enlarged by allowing terms in $\mathcal{T}(\mathcal{G})$ to be substituted for the variables in the rewrite rules. We stress that this condition is decidable for left-linear $\mathcal{R}$ and recognizability preserving α by standard term automata techniques.

Theorem 4.1.9. *Let $(\mathcal{R}, \mathcal{F})$, $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$, and $\mathcal{G} \supseteq \mathcal{F}$ such that $\mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$. If $(\mathcal{R}, \mathcal{F}) \in \mathsf{CBN}_\alpha$ and $\mathcal{R}_\alpha$ is collapsing then $(\mathcal{R}, \mathcal{G}) \in \mathsf{CBN}_\alpha$.*

The necessity of the $\mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ condition for collapsing $\mathcal{R}_\alpha$ is a consequence of Example 4.1.3. The system $\mathcal{R}$ in that example is a collapsing orthogonal system with $(\mathcal{R}, \mathcal{F}) \in \mathsf{CBN}_{\mathsf{nv}}$, $(\mathcal{R}, \mathcal{G}) \notin \mathsf{CBN}_{\mathsf{nv}}$, and $\mathsf{WN}(\mathcal{R}_{\mathsf{nv}}, \mathcal{F}) \neq \mathsf{WN}(\mathcal{R}_{\mathsf{nv}}, \mathcal{G}, \mathcal{F})$ as witnessed by the term $\mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{b})$. The following example shows the necessity of the collapsing condition.

Example 4.1.10. *Consider system $\mathcal{R}$*

$$
\begin{aligned}
\mathsf{f}(x, \mathsf{a}, \mathsf{b}(y, z)) &\to \mathsf{c}(\infty) & \mathsf{g}(x) &\to \mathsf{b}(x, \infty)\\
\mathsf{f}(x, \mathsf{a}, \mathsf{c}(y)) &\to \infty & \mathsf{h}(\mathsf{a}) &\to \infty\\
\mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{a}) &\to \infty & \mathsf{h}(\mathsf{b}(\mathsf{a}, x)) &\to \mathsf{a}\\
\mathsf{f}(\mathsf{a}, \mathsf{b}(x, y), z) &\to \mathsf{a} & \mathsf{h}(\mathsf{b}(\mathsf{b}(x, y), z)) &\to \mathsf{b}(\infty, \infty)\\
\mathsf{f}(\mathsf{a}, \mathsf{c}(x), y) &\to \infty & \mathsf{h}(\mathsf{b}(\mathsf{c}(x), y)) &\to \infty\\
\mathsf{f}(\mathsf{b}(x, y), z, \mathsf{a}) &\to \mathsf{a} & \mathsf{h}(\mathsf{c}(x)) &\to \infty\\
\mathsf{f}(\mathsf{b}(x, y), \mathsf{b}(z, u), \mathsf{b}(v, w)) &\to \infty & \mathsf{i}(\mathsf{a}, \mathsf{a}) &\to \infty\\
\mathsf{f}(\mathsf{b}(x, y), \mathsf{b}(z, u), \mathsf{c}(v)) &\to \infty & \mathsf{i}(\mathsf{a}, \mathsf{b}(x, y)) &\to \infty\\
\mathsf{f}(\mathsf{b}(x, y), \mathsf{c}(z), \mathsf{b}(u, v)) &\to \infty & \mathsf{i}(\mathsf{a}, \mathsf{c}(x)) &\to \infty\\
\mathsf{f}(\mathsf{b}(x, y), \mathsf{c}(z), \mathsf{c}(u)) &\to \infty & \mathsf{i}(\mathsf{b}(x, y), z) &\to \infty\\
\mathsf{f}(\mathsf{c}(x), \mathsf{a}, \mathsf{a}) &\to \infty & \mathsf{i}(\mathsf{c}(x), y) &\to \mathsf{a}\\
\mathsf{f}(\mathsf{c}(x), \mathsf{b}(y, z), \mathsf{a}) &\to \infty & \infty &\to \infty\\
\mathsf{f}(\mathsf{c}(x), \mathsf{b}(y, z), \mathsf{c}(u)) &\to \infty
\end{aligned}
$$

$$\mathsf{f}(\mathsf{c}(x), \mathsf{b}(y,z), \mathsf{b}(u,v)) \to \infty$$
$$\mathsf{f}(\mathsf{c}(x), \mathsf{c}(y), z) \to \infty$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules and let $\mathcal{G} = \mathcal{F} \cup \{@\}$ with $@$ a constant. One easily checks that the term $\mathsf{i}(\mathsf{f}(\Delta,\Delta,\Delta),@)$ with $\Delta = \mathsf{h}(\mathsf{g}(@))$ lacks $(\mathcal{R}_{\mathsf{nv}},\mathcal{G})$-needed redexes and hence $(\mathcal{R},\mathcal{G}) \notin \mathsf{CBN}_{\mathsf{nv}}$. `Autowrite` is able to verify $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_{\mathsf{nv}}$ and $\mathsf{WN}(\mathcal{R}_{\mathsf{nv}},\mathcal{F}) = \mathsf{WN}(\mathcal{R}_{\mathsf{nv}},\mathcal{G},\mathcal{F})$.

The next example shows the necessity of the restriction to $\alpha \in \{\mathsf{s},\mathsf{nv}\}$.

Example 4.1.11. *Consider the orthogonal system $\mathcal{R}$*

$$\begin{aligned}
\mathsf{f}(x,\mathsf{a},\mathsf{b}(y),z) &\to \mathsf{g}(z) & \mathsf{g}(\mathsf{a}) &\to \infty\\
\mathsf{f}(\mathsf{b}(x),y,\mathsf{a},z) &\to \mathsf{g}(z) & \mathsf{g}(\mathsf{b}(x)) &\to \infty\\
\mathsf{f}(\mathsf{a},\mathsf{b}(x),y,z) &\to \mathsf{g}(z) & \mathsf{h}(\mathsf{b}(x)) &\to \mathsf{j}(\infty,x)\\
\mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{a},z) &\to \infty & \mathsf{h}(\mathsf{a}) &\to \infty\\
\mathsf{f}(\mathsf{b}(x),\mathsf{b}(y),\mathsf{b}(z),u) &\to \infty & \mathsf{j}(x,\mathsf{a}) &\to \mathsf{a}\\
\infty &\to \infty & \mathsf{j}(x,\mathsf{b}(y)) &\to \mathsf{b}(\mathsf{a})
\end{aligned}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules. Note that the growing approximation only modifies the rule $\mathsf{h}(\mathsf{b}(x)) \to \mathsf{j}(\infty,x)$ into $\mathsf{h}(\mathsf{b}(x)) \to \mathsf{j}(\infty,y)$. Let $\mathcal{G} = \mathcal{F} \cup \{@\}$ with $@$ a constant. As the term $\mathsf{f}(\mathsf{h}(\mathsf{b}(@)),\mathsf{h}(\mathsf{b}(@)),\mathsf{h}(\mathsf{b}(@))),@)$ lacks $(\mathcal{R}_{\mathsf{g}},\mathcal{G})$-needed redexes, $(\mathcal{R},\mathcal{G}) \notin \mathsf{CBN}_{\mathsf{g}}$. `Autowrite` is able to verify $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_{\mathsf{g}}$ and $\mathsf{WN}(\mathcal{R}_{\mathsf{g}},\mathcal{F}) = \mathsf{WN}(\mathcal{R}_{\mathsf{g}},\mathcal{G},\mathcal{F})$. Note that $\mathcal{R}$ is not collapsing. This is not essential, since adding the single collapsing rule $\mathsf{k}(x) \to x$ to $\mathcal{R}$ does not affect any of the above properties.

We show that Theorem 4.1.4 is a special case of Theorem 4.1.9 by proving that for $\alpha = \mathsf{s}$ the condition $\mathsf{WN}(\mathcal{R}_\alpha,\mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha,\mathcal{G},\mathcal{F})$ is a consequence of $\mathsf{NF}(\mathcal{R},\mathcal{F}) \neq \varnothing$.

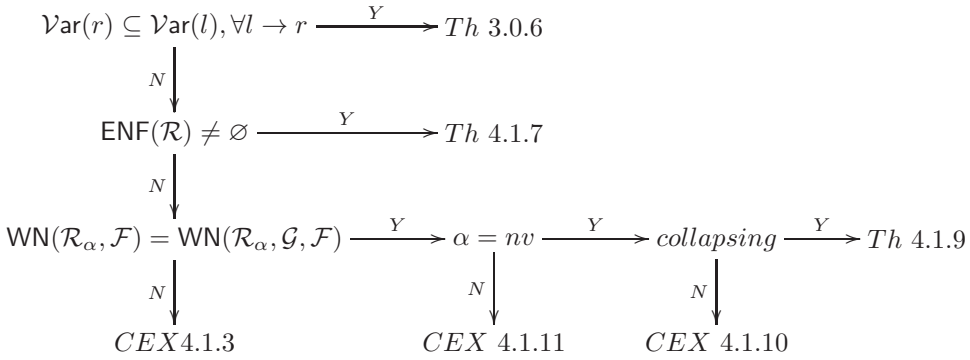
Lemma 4.1.12. *Let $(\mathcal{R},\mathcal{F})$ be a system. If $\mathsf{NF}(\mathcal{R},\mathcal{F}) \neq \varnothing$ then $\mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{F}) = \mathcal{T}(\mathcal{F})$.*

Proof. If $\mathsf{NF}(\mathcal{R},\mathcal{F}) \neq \varnothing$ then there must be a constant $c \in \mathsf{NF}(\mathcal{R},\mathcal{F})$. Define the system $\mathcal{R}' = \{l \to c \mid l \to r \in \mathcal{R}\}$ over the signature $\mathcal{F}$. Clearly $\to_{\mathcal{R}'} \subseteq \to_{\mathsf{s}}$. The system $\mathcal{R}'$ is terminating since every rewrite step reduces the number of function symbols in $\mathcal{F} \setminus \{c\}$. Since $\mathcal{R}_{\mathsf{s}}$ and $\mathcal{R}'$ have the same normal forms, it follows that $\mathcal{R}_{\mathsf{s}}$ is weakly normalizing. □

Proof. of Theorem 4.1.4. Let $\mathcal{R}$ be an orthogonal system over a signature $\mathcal{F}$ such that $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_{\mathsf{s}}$. Let $\mathcal{F} \subseteq \mathcal{G}$. We have to show that $(\mathcal{R},\mathcal{G}) \in \mathsf{CBN}_{\mathsf{s}}$. If $\mathcal{R} = \varnothing$, this is trivial. Otherwise $\mathcal{R}_{\mathsf{s}}$ is collapsing and the result follows from Theorem 4.1.9 provided that $\mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{F}) = \mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{G},\mathcal{F})$. From Lemma 4.1.12 we obtain $\mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{F}) = \mathcal{T}(\mathcal{F})$ and $\mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{G},\mathcal{F}) = \mathsf{WN}(\mathcal{R}_{\mathsf{s}},\mathcal{G}) \cap \mathcal{T}(\mathcal{F}) = \mathcal{T}(\mathcal{G}) \cap \mathcal{T}(\mathcal{F}) = \mathcal{T}(\mathcal{F})$. □

Remark that we have to use Theorem 4.1.9 only once. After adding a single new function symbol we obtain an external normal form and hence we can apply Theorem 4.1.7 for the remaining new function symbols.

The following picture summarizes the results concerning signature extension.



To conclude this section, Proposition 4.1.15 gives an unpublished folklore result whose main consequence is that, in order to check whether signature extension holds one just has to add a single constant symbol. We have taken advantage of this property in `Autowrite` (see Chapter 15) where we consider signature extension with just the constant symbol @. Two additional lemmata are needed for the proof of Proposition 4.1.15.

Lemma 4.1.13. *Let $(\mathcal{R}, \mathcal{F})$ be a left-linear esystem, $\mathcal{G} \supseteq \mathcal{F}$, $t, s \in \mathcal{T}(\mathcal{G})$ and $c \in \mathcal{T}(\mathcal{F})$. Let t^c and s^c be t and s where every outermost symbol in $\mathcal{G} \setminus \mathcal{F}$ have been replaced by c.*
If $t \to^*_{\mathcal{R},\mathcal{G}} s$ then $t^c \to^*_{\mathcal{R},\mathcal{F}} s^c$.

Proof. Obvious as the symbols in $\mathcal{G} \setminus \mathcal{F}$ do not participate to the left-hand side of any redex. □

Lemma 4.1.14. *Let $(\mathcal{R}, \mathcal{F})$ be an esystem. Let $\mathcal{G} \supseteq \mathcal{F}$. Let @ be a fresh constant symbol. Let $\mathcal{F}_@ = \mathcal{F} \cup \{@\}$. Let $t \in \mathcal{T}(\mathcal{G})$. Let $t^@$ obtained from t by replacing every outermost subterm with root in $\mathcal{G} \setminus \mathcal{F}$ by @.*
(1) If $t \in \mathsf{NF}(\mathcal{R}, \mathcal{G})$ then $t^@ \in \mathsf{NF}(\mathcal{R}, \mathcal{F}_@)$.
(2) If $t \in \mathsf{WN}(\mathcal{R}, \mathcal{G})$ then $t^@ \in \mathsf{WN}(\mathcal{R}, \mathcal{F}_@)$.

Proof. (1) $t \in \mathsf{NF}(\mathcal{R}, \mathcal{G})$. If $\mathsf{root}(t) \in \mathcal{G} \setminus \mathcal{F}$ then $t^@ = @ \in \mathsf{NF}(\mathcal{R}, \mathcal{F}_@)$ otherwise $t^@$ cannot contain a redex otherwise t would also contain a redex. So (1) holds. (2) $t \in \mathsf{WN}(\mathcal{R}, \mathcal{G})$. Then $t \to^*_{\mathcal{R},\mathcal{G}} s$ for some $s \in \mathsf{NF}(\mathcal{R}, \mathcal{G})$. Trivially, $t \to^*_{\mathcal{R},\mathcal{G}_@} s$ and $s \in \mathsf{NF}(\mathcal{R}, \mathcal{G}_@)$. As $@ \in \mathcal{F}_@$, Lemma 4.1.13 yields $t^@ \to^*_{\mathcal{R},\mathcal{F}_@} s^@$. (1) yields $s^@ \in \mathsf{NF}(\mathcal{R}, \mathcal{F}_@)$. It follows that $t^@ \in \mathsf{WN}(\mathcal{R}, \mathcal{F}_@)$. □

Proposition 4.1.15. *Let α be an arbitrary approximation mapping. Let $(\mathcal{R}, \mathcal{F})$ a system such that $\mathcal{R} \in \mathsf{CBN}_\alpha$. Let $\mathcal{G} \supseteq \mathcal{F}$. Let @ be fresh constant symbol (not in $\mathcal{G}$). Let $\mathcal{F}_@ = \mathcal{F} \cup \{@\}$. If $(\mathcal{R}, \mathcal{G}) \notin \mathsf{CBN}_\alpha$ then $(\mathcal{R}, \mathcal{F}_@) \notin \mathsf{CBN}_\alpha$.*

Proof. Let t be a minimal $(\mathcal{R}_\alpha, \mathcal{G})$-free term. By Lemma B.0.6, t is not $(\mathcal{R}_\alpha, \mathcal{G})$-root-stable. Let $t^{@}$ obtained from t by replacing every outermost subterm with root in $\mathcal{G} \setminus \mathcal{F}$ by @. $t^{@}$ is necessarily reducible otherwise t would be $(\mathcal{R}_\alpha, \mathcal{G})$-root-stable. Let p be a redex in $t^{@}$. p is also a redex in t and as t is free we get $t[\bullet]_p \in \mathsf{WN}(\mathcal{R}_{\alpha\bullet}, \mathcal{G}_\bullet)$. By Lemma 4.1.14, we get $t^{@}[\bullet]_p \in \mathsf{WN}(\mathcal{R}_{\alpha\bullet}, \mathcal{F}_{@\bullet})$. We conclude that $t^{@}$ is a $(\mathcal{R}_\alpha, \mathcal{F}_@)$-free term so that $(\mathcal{R}, \mathcal{F}_@) \notin \mathsf{CBN}_\alpha$. □

4.2 Modularity

The results obtained in the previous section form the basis for the modularity results presented in this section. We first consider disjoint combinations.

Definition 4.2.1. *We say that a class $\mathcal{C}$ of systems is* modular (for disjoint combinations) *if $(\mathcal{R}_1 \cup \mathcal{R}_2, \mathcal{F}_1 \cup \mathcal{F}_2) \in \mathcal{C}$ for all $(\mathcal{R}_1, \mathcal{F}_1), (\mathcal{R}_2, \mathcal{F}_2) \in \mathcal{C}$ such that $\mathcal{F}_1 \cap \mathcal{F}_2 = \varnothing$.*

To simplify notation, in the remainder of this section we write $\mathcal{S}$ for $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{G}$ for $\mathcal{F}_1 \cup \mathcal{F}_2$.

The condition in Theorem 4.1.7 is insufficient for modularity as shown by the following example.

Example 4.2.2. *Consider the system $\mathcal{R}_1$*

$$\mathsf{f}(x, \mathsf{a}, \mathsf{b}) \to \mathsf{a} \qquad \mathsf{f}(\mathsf{b}, x, \mathsf{a}) \to \mathsf{a} \qquad \mathsf{f}(\mathsf{a}, \mathsf{b}, x) \to \mathsf{a}$$

over the signature $\mathcal{F}_1$ consisting of all symbols appearing in the rewrite rules and the system $\mathcal{R}_2 = \{\mathsf{g}(x) \to x\}$ over the signature $\mathcal{F}_2$ consisting of a constant c *in addition to* g*. Both systems have external normal forms and belong to $\mathsf{CBN}_{\mathsf{nv}}$, as one easily shows. Their union does not belong to $\mathsf{CBN}_{\mathsf{nv}}$ as the term $\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a}))$ has no $(\mathcal{S}_{\mathsf{nv}}, \mathcal{G})$-needed redex:*

$$\begin{aligned}
&\mathsf{f}(\bullet, \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\bullet, \mathsf{a}, \mathsf{g}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\bullet, \mathsf{a}, \mathsf{b}) \to_{\mathsf{nv}} \mathsf{a} \\
&\mathsf{f}(\mathsf{g}(\mathsf{a}), \bullet, \mathsf{g}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{b}, \bullet, \mathsf{g}(\mathsf{a})) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{b}, \bullet, \mathsf{a}) \to_{\mathsf{nv}} \mathsf{a} \\
&\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a}), \bullet) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{a}, \mathsf{g}(\mathsf{a}), \bullet) \to_{\mathsf{nv}} \mathsf{f}(\mathsf{a}, \mathsf{b}, \bullet) \to_{\mathsf{nv}} \mathsf{a}
\end{aligned}$$

If we forbid collapsing rules like $\mathsf{g}(x) \to x$, modularity holds. The following theorem is proved along the lines of the proof of Theorem 4.1.7; because there are no collapsing rules and the esystems are left-linear, aliens (see Appendix B) cannot influence the possibility to perform a rewrite step in the non-alien part of a term.

Theorem 4.2.3. *Let α be an arbitrary approximation mapping. The subclass of CBN_α consisting of all left-linear systems $\mathcal{R}$ with external normal forms such that $\mathcal{R}_\alpha$ is non-collapsing is modular.*

The following result is the modularity counterpart of Theorem 4.1.9. The proof is given in the appendix.

Theorem 4.2.4. *Let* $(\mathcal{R}_1, \mathcal{F}_1)$ *and* $(\mathcal{R}_2, \mathcal{F}_2)$ *be disjoint orthogonal systems and* $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$ *such that both* $\mathsf{WN}(\mathcal{R}_{1\alpha}, \mathcal{G}, \mathcal{F}_1) = \mathsf{WN}(\mathcal{R}_{1\alpha}, \mathcal{F}_1)$ *and* $\mathsf{WN}(\mathcal{R}_{2\alpha}, \mathcal{G}, \mathcal{F}_2) = \mathsf{WN}(\mathcal{R}_{2\alpha}, \mathcal{F}_2)$. *If* $(\mathcal{R}_1, \mathcal{F}_1), (\mathcal{R}_2, \mathcal{F}_2) \in \mathsf{CBN}_\alpha$ *and both* $\mathcal{R}_{1\alpha}$ *and* $\mathcal{R}_{2\alpha}$ *are collapsing then* $(\mathcal{S}, \mathcal{G}) \in \mathsf{CBN}_\alpha$.

It is rather surprising that the presence of collapsing rules helps to achieve modularity; for most properties of systems collapsing rules are an obstacle for modularity (see e.g. Middeldorp [Mid90]).

The necessity of the collapsing condition is shown already for signature extension so a counterexample showing the necessity of the collapsing condition could be $(\mathcal{R}_1, \mathcal{F}_1) = (\mathcal{R}, \mathcal{F})$ of Example 4.1.11 and $(\mathcal{R}_2, \mathcal{F}_2) = (\varnothing, \{@\})$.

The next result is the modularity counterpart of Theorem 4.1.4. It is an easy corollary of the preceding theorem.

Corollary 4.2.5. *The subclass of* CBN_s *consisting of all orthogonal systems* $(\mathcal{R}, \mathcal{F})$ *such that* $\mathsf{NF}(\mathcal{R}, \mathcal{F}) \neq \varnothing$ *is modular.*

Using Huet and Lévy's characterization of strong sequentiality by means of increasing indexes, Klop and Middeldorp [KM91] showed that strong sequentiality is a modular property of orthogonal systems. Since membership in CBN_s coincides with strong sequentiality for orthogonal systems with ground normal forms (Lemma 9.2.1), this provides another proof of Corollary 4.2.5. Actually, in [KM91] it is remarked that it is sufficient that the left-hand sides of the two strongly sequential rewrite systems do not share function symbols. One easily verifies that for our modularity results it is sufficient that $\mathcal{R}_{1\alpha}$ and $\mathcal{R}_{2\alpha}$ do not share function symbols. Actually, we can go a step further by considering so-called *constructor-sharing* combinations. In such combinations the participating systems may share constructors but not defined symbols.

Definition 4.2.6. *Two systems* $(\mathcal{R}_1, \mathcal{F}_1)$ *and* $(\mathcal{R}_2, \mathcal{F}_2)$ share constructors *if* $\mathcal{F}_{1\mathcal{D}} \cap \mathcal{F}_2 = \mathcal{F}_{2\mathcal{D}} \cap \mathcal{F}_1 = \varnothing$. *We say that a class* $\mathcal{C}$ *of systems is* constructor-sharing modular *if* $(\mathcal{R}_1 \cup \mathcal{R}_2, \mathcal{F}_1 \cup \mathcal{F}_2) \in \mathcal{C}$ *for all systems* $(\mathcal{R}_1, \mathcal{F}_1), (\mathcal{R}_2, \mathcal{F}_2) \in \mathcal{C}$ *that share constructors.*

It can be shown that the results obtained in this section extend to constructor-sharing combinations, provided we strengthen the requirements in Theorems 4.2.3 and 4.2.4 by forbidding the presence of *constructor-lifting* rules. A rewrite rule $l \to r$ is called constructor-lifting if $\mathsf{root}(r)$ is a shared constructor. In the appendix we give a detailed proof of the extension of Theorem 4.2.3. The proof of Theorem 4.2.4 is easily extended to constructor-sharing combinations and hence omitted.

Theorem 4.2.7. *Let* $(\mathcal{R}_1, \mathcal{F}_1)$ *and* $(\mathcal{R}_2, \mathcal{F}_2)$ *be left-linear constructor-sharing systems with external normal forms and without constructor-lifting rules and let* α *be an approximation mapping such that* $\mathcal{R}_{1\alpha}$ *and* $\mathcal{R}_{2\alpha}$ *are non-collapsing. If* $(\mathcal{R}_1, \mathcal{F}_1), (\mathcal{R}_2, \mathcal{F}_2) \in \mathsf{CBN}_\alpha$ *then* $(\mathcal{S}, \mathcal{G}) \in \mathsf{CBN}_\alpha$.

The reason for excluding constructor-lifting rules in Theorem 4.2.7 is shown in the following example.

Example 4.2.8. *Consider the system* $\mathcal{R}_1$

$$\mathsf{f}(x,\mathsf{c}(\mathsf{a}),\mathsf{c}(\mathsf{b}))\to\mathsf{a}\qquad \mathsf{f}(\mathsf{c}(\mathsf{b}),x,\mathsf{c}(\mathsf{a}))\to\mathsf{a}\qquad \mathsf{f}(\mathsf{c}(\mathsf{a}),\mathsf{c}(\mathsf{b}),x)\to\mathsf{a}$$

over the signature $\mathcal{F}_1$ *consisting of all symbols appearing in the rewrite rules and the system* $\mathcal{R}_2=\{\mathsf{g}(x)\to\mathsf{c}(x)\}$ *over the signature* $\mathcal{F}_2$ *consisting of a constant* d *in addition to* g *and* c*. Both systems have external normal forms, lack collapsing rules, and belong to* $\mathsf{CBN}_{\mathsf{nv}}$*. Their union does not belong to* $\mathsf{CBN}_{\mathsf{nv}}$ *as the term* $\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a}))$ *has no* $(\mathcal{S}_{\mathsf{nv}},\mathcal{G})$*-needed redex. Note that* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *share the constructor* c *and hence* $\mathsf{g}(x)\to\mathsf{c}(x)$ *is constructor-lifting.*

Theorem 4.2.9. *Let* $(\mathcal{R}_1,\mathcal{F}_1)$ *and* $(\mathcal{R}_2,\mathcal{F}_2)$ *be orthogonal constructor-sharing systems without constructor-lifting rules and* $\alpha\in\{\mathsf{s},\mathsf{nv}\}$ *such that* $\mathsf{WN}(\mathcal{R}_{1\alpha},\mathcal{G},\mathcal{F}_1)=\mathsf{WN}(\mathcal{R}_{1\alpha},\mathcal{F}_1)$ *and* $\mathsf{WN}(\mathcal{R}_{2\alpha},\mathcal{G},\mathcal{F}_2)=\mathsf{WN}(\mathcal{R}_{2\alpha},\mathcal{F}_2)$*. If* $(\mathcal{R}_1,\mathcal{F}_1),(\mathcal{R}_2,\mathcal{F}_2)\in\mathsf{CBN}_\alpha$ *and both* $\mathcal{R}_{1\alpha}$ *and* $\mathcal{R}_{2\alpha}$ *are collapsing then* $(\mathcal{S},\mathcal{G})\in\mathsf{CBN}_\alpha$*.*

Again, it is essential that constructor-lifting rules are excluded.

Example 4.2.10. *Consider the systems* $\mathcal{R}_1$

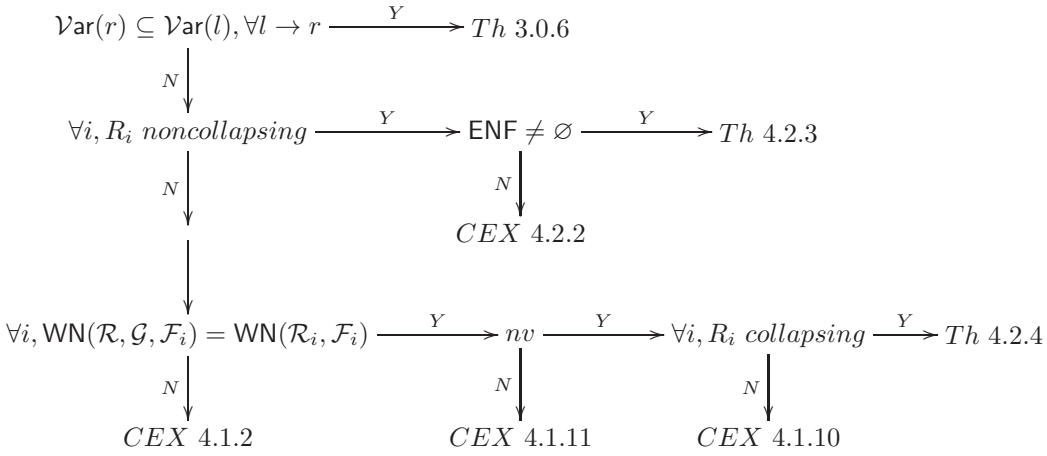
$$\begin{array}{ll}\mathsf{f}(x,\mathsf{a},\mathsf{b})\to\mathsf{c}(\mathsf{g}(x)) & \mathsf{g}(x)\to\mathsf{g}(\mathsf{a})\\ \mathsf{f}(\mathsf{b},x,\mathsf{a})\to\mathsf{c}(\mathsf{g}(x)) & \mathsf{h}(x)\to x\\ \mathsf{f}(\mathsf{a},\mathsf{b},x)\to\mathsf{c}(\mathsf{g}(x)) & \end{array}$$

and $\mathcal{R}_2=\{\mathsf{i}(\mathsf{a})\to\mathsf{a},\mathsf{i}(\mathsf{c}(x))\to x\}$ *over the signatures* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *consisting of function symbols that appear in their respective rewrite rules. The two systems are obviously collapsing and share the constructors* a *and* c*. One easily verifies that both systems belong to* $\mathsf{CBN}_{\mathsf{nv}}$ *and that* $\mathsf{WN}(\mathcal{R}_{1\mathsf{nv}},\mathcal{G},\mathcal{F}_1)=\mathsf{WN}(\mathcal{R}_{1\mathsf{nv}},\mathcal{F}_1)$ *and* $\mathsf{WN}(\mathcal{R}_{1\mathsf{nv}},\mathcal{G},\mathcal{F}_2)=\mathcal{T}(\mathcal{F}_2)=\mathsf{WN}(\mathcal{R}_{2\mathsf{nv}},\mathcal{F}_2)$*. However, the union of the two systems does not belong to* $\mathsf{CBN}_{\mathsf{nv}}$ *as the term* $\mathsf{i}(\mathsf{f}(\mathsf{h}(\mathsf{a}),\mathsf{h}(\mathsf{a}),\mathsf{h}(\mathsf{a})))$ *has no* $(\mathcal{S}_{\mathsf{nv}},\mathcal{G})$*-needed redex.*

For the strong approximation we need of course not exclude constructor-sharing rules. Moreover, the two conditions $\mathsf{WN}(\mathcal{R}_{1\mathsf{s}},\mathcal{G},\mathcal{F}_1)=\mathsf{WN}(\mathcal{R}_{1\alpha},\mathcal{F}_1)$ and $\mathsf{WN}(\mathcal{R}_{2\mathsf{s}},\mathcal{G},\mathcal{F}_2)=\mathsf{WN}(\mathcal{R}_{2\alpha},\mathcal{F}_2)$ are always satisfied (cf. the proof of Theorem 4.1.4). Hence we can state the final result of the chapter.

Corollary 4.2.11. *Let* $(\mathcal{R}_1,\mathcal{F}_1)$ *and* $(\mathcal{R}_2,\mathcal{F}_2)$ *be orthogonal constructor-sharing systems with ground normal forms. If* $(\mathcal{R}_1,\mathcal{F}_1),(\mathcal{R}_2,\mathcal{F}_2)\in\mathsf{CBN}_{\mathsf{s}}$ *then* $(\mathcal{S},\mathcal{G})\in\mathsf{CBN}_{\mathsf{s}}$*.*

The following picture summarizes the results concerning modularity.



Chapter 5

Complexity of CBN classes

The results of this chapter are unpublished although some preliminary results can be found in the technical report [DM98]. The ultimate aim of this chapter is to analyze the complexity of deciding membership to $\mathsf{CBN}_{\mathsf{g}}$ which contain all the CBN_{α} classes that we have defined. The case $\alpha = \mathsf{s}$ will be discussed again in Chapter 13. The intermediate case $\alpha = \mathsf{nv}$ has not been studied yet.

To study the complexity of deciding membership to $\mathsf{CBN}_{\mathsf{g}}$, we need to detail all the constructions of the automata used to decide membership to $\mathsf{CBN}_{\mathsf{g}}$. Thoughout the whole chapter, we will illustrate these constructions on the following example taken from [NT02]. Consider the orthogonal growing system $\mathcal{R}$ consisting of the rewrite rules

$$\begin{array}{lcl} \mathsf{g}(x) & \to & \mathsf{f}(x,x,x) \\ \mathsf{f}(x,\mathsf{a},\mathsf{b}) & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{b},x,\mathsf{a}) & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{a},\mathsf{b},x) & \to & \mathsf{b} \end{array}$$

We have $\mathcal{R}_{\mathsf{g}} = \mathcal{R}$ and

$$\mathcal{R}_{\mathsf{lg}} = \left\{ \begin{array}{lcl} \mathsf{g}(x) & \to & \mathsf{f}(x,y,z) \\ \mathsf{f}(x,\mathsf{a},\mathsf{b}) & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{b},x,\mathsf{a}) & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{a},\mathsf{b},x) & \to & \mathsf{b} \end{array} \right.$$

The first section gives basic easy constructions. The second section shows the construction of automata $\mathcal{C}_T(\mathcal{R})$ and $\mathcal{D}_T(\mathcal{R})$ regognizing $(\to^*_{\mathcal{R}})[T]$ given a recognizable term language T and a growing esystem $\mathcal{R}$. The third section gives constructions of automata $\mathcal{E}(\mathcal{R})$ recognizing the set of $\mathcal{R}$-free terms (having no $\mathcal{R}$-needed redex). With such automata one can decide whether a system $\mathcal{R}$ is in CBN_{α} by checking whether $\mathcal{E}(\mathcal{R}_{\alpha})$ recognizes the empty language. The fourth section contains the complexity analysis. Different constructions are given for $\alpha = \mathsf{lg}$ and $\alpha = \mathsf{g}$ which lead to different complexities for $\mathsf{CBN}_{\mathsf{lg}}$ and $\mathsf{CBN}_{\mathsf{g}}$.

5.1 Basic constructions

We consider finite bottom-up term automata without ϵ-transitions. Let $(\mathcal{R}, \mathcal{F})$ be a (left-linear) growing esystem. Let $\mathcal{G} \supseteq \mathcal{F}$. Let $T \subseteq \mathcal{T}(\mathcal{G})$ be a recognizable set of terms and $\mathcal{A}_T = (\mathcal{G}, Q_\mathcal{A}, Q_f, \Gamma_\mathcal{A})$ a term automaton recognizing T. We assume without loss of generality that all states of $\mathcal{A}_T$ are accessible.

5.1.1 Step 1

Let $S_\mathcal{R}$ be the set of all subterms of the arguments of the left-hand sides of $\mathcal{R}$. Construct the term automaton $\mathcal{B}(\mathcal{R}) = (\mathcal{G}, Q_\mathcal{B}, \varnothing, \Gamma_\mathcal{B})$ with $Q_\mathcal{B} = \{[t] \mid t \in S_\mathcal{R}\} \cup \{[x]\}$ and $\Gamma_\mathcal{B}$ consisting of the *matching* rules $f([t_1], \ldots, [t_n]) \to [t]$ for every term $t = f(t_1, \ldots, t_n)$ in $S_\mathcal{R}$ and *propagation* rules $f([x], \ldots, [x]) \to [x]$ for every $f \in \mathcal{G}$. Here $[t]$ denotes the equivalence class of the term t with respect to literal similarity. (So we identify $[s]$ and $[t]$ whenever s and t differ by variable renaming.) Note that all states of $\mathcal{B}(\mathcal{R})$ are accessible. From now on in this chapter, we write $\mathcal{B}$ for $\mathcal{B}(\mathcal{R})$ when $\mathcal{R}$ is understood. The set of ground instances of a term t is denoted by $\Sigma(t)$. Clearly, for $t \in S_\mathcal{R} \cup \{x\}$, we have $s \in \Sigma(t)$ if and only if $s \to^*_{\Gamma_\mathcal{B}} [t]$.

For $\mathcal{G} = \mathcal{F} \cup \{\bullet\}$, the automaton $\mathcal{B}(\mathcal{R})$ has states $Q_\mathcal{B} = \{[x], [\mathsf{a}], [\mathsf{b}]\}$ and transition rules $\Gamma_\mathcal{B}$:

propagation rules	a	$\to$	$[x]$	$\mathsf{g}([x])$	$\to$	$[x]$
	b	$\to$	$[x]$	$\mathsf{f}([x],[x],[x])$	$\to$	$[x]$
	$\bullet$	$\to$	$[x]$			
matching rules	a	$\to$	$[\mathsf{a}]$	b	$\to$	$[\mathsf{b}]$

5.1.2 Step 2

We assume that $\{t \mid t \to^+_{\Gamma_\mathcal{A}} q\} = \{t \mid t \to^+_{\Gamma_\mathcal{B}} q\}$ for every state $q \in Q_\mathcal{A} \cap Q_\mathcal{B}$. This can always be achieved by a renaming of states. Note that the automata $\mathcal{A}$ and $\mathcal{B}$ obtained for our example $\mathcal{R}$ share states $[a]$ and $[b]$. Let $\mathcal{U}_T(\mathcal{R}) = (\mathcal{G}, Q, Q^f, \Gamma_\mathcal{U})$ be the union of $\mathcal{A}_T$ and $\mathcal{B}$, so $Q = Q_\mathcal{A} \cup Q_\mathcal{B}$, $Q^f = Q^f_\mathcal{A}$ and $\Gamma_\mathcal{U} = \Gamma_\mathcal{A} \cup \Gamma_\mathcal{B}$.

Lemma 5.1.1.

1. *Let $t \in S_\mathcal{R} \cup \{x\}$. We have $s \in \Sigma(t)$ if and only if $s \to^*_{\Gamma_\mathcal{U}} [t]$.*

2. *$L(\mathcal{U}_T(\mathcal{R})) = T$.*

For our example, the set $T = \mathsf{NF}$ is accepted by the automaton $\mathcal{A}_\mathsf{NF}$ with states $Q_\mathcal{A} = Q^f_\mathcal{A} = \{[X], [\mathsf{a}], [\mathsf{b}]\}$ and transition rules $\Gamma_\mathcal{A}$: consisting of $\mathsf{a} \to [\mathsf{a}]$, $\mathsf{b} \to [\mathsf{b}]$ and rules $\mathsf{f}(q, q', q'') \to [X]$ for all triple $(q, q', q'') \in Q_\mathcal{A} \times Q_\mathcal{A} \times Q_\mathcal{A}$ except the ones of the form $(q, [\mathsf{a}], [\mathsf{b}]), ([\mathsf{b}], q', [\mathsf{a}])$ and $([\mathsf{a}], [\mathsf{b}], q'')$.

$$
\begin{array}{rrr}
\mathsf{a} \to [\mathsf{a}] & \mathsf{b} \to [\mathsf{b}] & \mathsf{f}([X],[X],[X]) \to [X] \\
\mathsf{f}([X],[X],[\mathsf{a}]) \to [X] & \mathsf{f}([X],[X],[\mathsf{b}]) \to [X] & \mathsf{f}([X],[\mathsf{a}],[X]) \to [X] \\
\mathsf{f}([X],[\mathsf{a}],[\mathsf{a}]) \to [X] & \mathsf{f}([X],[\mathsf{b}],[X]) \to [X] & \mathsf{f}([X],[\mathsf{b}],[\mathsf{a}]) \to [X] \\
\mathsf{f}([X],[\mathsf{b}],[\mathsf{b}]) \to [X] & \mathsf{f}([\mathsf{a}],[X],[X]) \to [X] & \mathsf{f}([\mathsf{a}],[X],[\mathsf{a}]) \to [X] \\
\mathsf{f}([\mathsf{a}],[X],[\mathsf{b}]) \to [X] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[X]) \to [X] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[\mathsf{a}]) \to [X] \\
\mathsf{f}([\mathsf{b}],[X],[X]) \to [X] & \mathsf{f}([\mathsf{b}],[X],[\mathsf{b}]) \to [X] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[X]) \to [X] \\
\mathsf{f}([\mathsf{b}],[\mathsf{b}],[X]) \to [X] & \mathsf{f}([\mathsf{b}],[\mathsf{b}],[\mathsf{b}]) \to [X] &
\end{array}
$$

Note that $\mathcal{A}_{\mathsf{NF}}$ and $\mathcal{B}$ share states $[\mathsf{a}]$ and $[\mathsf{b}]$, which may be allowed since both automata accept the same set of terms in those states ($\{\mathsf{a}\}$ and $\{\mathsf{b}\}$ respectively). Let $Q = Q_{\mathcal{A}} \cup Q_{\mathcal{B}}$ and $\Gamma_{\mathcal{U}} = \Gamma_{\mathcal{A}} \cup \Gamma_{\mathcal{B}}$.

5.2 Recognizability of $(\to_{\mathcal{R}}^*)[T]$

The goal of this section is to construct an automaton that recognizes the set $(\to_{\mathcal{R}}^*)[T]$ of ground terms that rewrite to a term in T. The construction of a non-deterministic such automaton $\mathcal{C}_T(\mathcal{R})$ was first given by Jacquemard [Jac96b] for linear-growing $\mathcal{R}$ and $T = \mathsf{NF}(\mathcal{R}, \mathcal{F})$ and by Comon [Com00] for arbitrary T and shallow $\mathcal{R}$. Nagaya and Toyama [NT02] give the construction of a deterministic automaton $\mathcal{D}_T(\mathcal{R})$ for arbitrary T and growing $\mathcal{R}$. The set of states of Jacquemard's non-deterministic automaton is Q whereas the set of states of Nagaya and Toyama's deterministic automaton is a subset of 2^Q.

5.2.1 Linear-growing case

We extend (and simplifiy) the construction given by Jacquemard in [Jac96b] for $\mathsf{NF}_{\mathcal{R}}$ to any regognizable term language T. The construction starts with $\mathcal{C}_T(\mathcal{R}) = (\mathcal{G}, Q, Q^f, \Gamma_{\mathcal{C}}) = \mathcal{U}_T(\mathcal{R})$ (constructed in Section 5.1.2). We saturate the transition rules $\Gamma_{\mathcal{C}}$ of $\mathcal{C}_T(\mathcal{R})$ under the following inference rule:

$$
\frac{f(l_1, \ldots, l_n) \to r \in \mathcal{R} \quad r\theta \to_{\Gamma_{\mathcal{C}}}^* q}{\Gamma_{\mathcal{C}} = \Gamma_{\mathcal{C}} \cup \{f(q_1, \ldots, q_n) \to q\}} \qquad (*)
$$

with θ mapping the variables in r to states in Q and

$$
q_i = \begin{cases} l_i\theta & \text{if } l_i \in \mathcal{V}\mathsf{ar}(r), \\ [l_i] & \text{otherwise.} \end{cases}
$$

Because Q is finite and no new state is added by $(*)$, the saturation process terminates. We now show that $L(\mathcal{C}_T(\mathcal{R})) = (\to_{\mathcal{R}}^*)[T]$ upon termination.

Lemma 5.2.1. $(\to_{\mathcal{R}}^*)[T] \subseteq L(\mathcal{C}_T(\mathcal{R}))$.

Proof. Let $s \in (\to_{\mathcal{R}}^*)[T]$. So there exists a term $t \in T$ such that $s \to_{\mathcal{R}}^* t$. We show that $s \in L(\mathcal{C}_T(\mathcal{R}))$ by induction on the length of $s \to_{\mathcal{R}}^* t$. If $s = t$ then $s \in T \subseteq L(\mathcal{C}_T(\mathcal{R}))$ according to Lemma 5.1.1(2). Let $s = C[l\sigma] \to_{\mathcal{R}} C[r\sigma] \to_{\mathcal{R}}^* t$ with $l = f(l_1, \ldots, l_n)$. The induction hypothesis yields $C[r\sigma] \in L(\mathcal{C}_T(\mathcal{R}))$. Hence there exists a final state q_f, a mapping θ from $\mathcal{V}\mathsf{ar}(r)$ to Q, and a state q such that $C[r\sigma] \to_{\Gamma_\mathcal{C}}^* C[r\theta] \to_{\Gamma_\mathcal{C}}^* C[q] \to_{\Gamma_\mathcal{C}}^* q_f$. By construction there exists a transition rule $f(q_1, \ldots, q_n) \to q \in \Gamma_\mathcal{C}$ such that $q_i = l_i\theta$ if $l_i \in \mathcal{V}\mathsf{ar}(r)$ and $q_i = [l_i]$ otherwise. We claim that $l\sigma \to_{\Gamma_\mathcal{C}}^* f(q_1, \ldots, q_n)$. Let $i \in \{1, \ldots, n\}$. If $l_i \in \mathcal{V}\mathsf{ar}(r)$ then $l_i\sigma \to_{\Gamma_\mathcal{C}}^* l_i\theta = q_i$, otherwise $l_i\sigma \to_{\Gamma_\mathcal{C}}^* [l_i] = q_i$ by Lemma 5.1.1(1). Consequently $s \to_{\Gamma_\mathcal{C}}^* C[f(q_1, \ldots, q_n)] \to_{\Gamma_\mathcal{C}} C[q] \to_{\Gamma_\mathcal{C}}^* q_f$ and hence $s \in L(\mathcal{C}_T(\mathcal{R}))$. □

Note that we don't use the growing assumption in the above proof; right-linearity of $\mathcal{R}$ is sufficient.

Lemma 5.2.2. *Let $s \in \mathcal{T}(\mathcal{G})$ with $s \to_{\Gamma_\mathcal{C}}^+ q$.*

1. *If $q = [t]$ with $t \in S_\mathcal{R} \cup \{x\}$ then $s \in (\to_{\mathcal{R}}^*)[\Sigma(t)]$.*
2. *If $q \in Q_f$ then $s \in (\to_{\mathcal{R}}^*)[T]$.*

Proof. Let $\Gamma_\mathcal{C}^k$ denote the value of $\Gamma_\mathcal{C}$ after the k-th transition rule has been added by $(*)$. We have $s \to_{\Gamma_\mathcal{C}^k}^+ q$ for some $k \geqslant 0$. We prove statements (1) and (2) by induction on k. If $k = 0$ then the result follows from Lemma 5.1.1. Let $s \to_{\Gamma_\mathcal{C}^{k+1}}^+ q$. We use a second induction on the number of steps that use the (unique) transition rule $f(q_1, \ldots, q_n) \to q' \in \Gamma_\mathcal{C}^{k+1} \setminus \Gamma_\mathcal{C}^k$. Suppose this rule is created from $l = f(l_1, \ldots, l_n) \to r \in \mathcal{R}$ and $r\theta \to_{\Gamma_\mathcal{C}^k}^* q'$. If this number is zero then the result follows from the first induction hypothesis. Otherwise we may write $s = C[f(s_1, \ldots, s_n)] \to_{\Gamma_\mathcal{C}^k}^* C[f(q_1, \ldots, q_n)] \to C[q'] \to_{\Gamma_\mathcal{C}^{k+1}}^* q$. We will define a substitution τ such that $s \to_{\mathcal{R}}^* C[l\tau] \to_{\mathcal{R}} C[r\tau] \to_{\Gamma_\mathcal{C}^k}^+ C[q']$. The second induction hypothesis applied to $C[r\tau] \to_{\Gamma_\mathcal{C}^{k+1}}^+ q$ then yields the desired result. We define τ as the (disjoint) union of $\tau_1, \ldots, \tau_n, \tau'$ such that $\mathcal{D}\mathsf{om}(\tau_i) = \mathcal{V}\mathsf{ar}(l_i)$ for $i = 1, \ldots, n$ and $\mathcal{D}\mathsf{om}(\tau') = \mathcal{V}\mathsf{ar}(r) \setminus \mathcal{V}\mathsf{ar}(l)$. Note that since l is a linear term, the union of $\tau_1, \ldots, \tau_n$ is well-defined. Fix $i \in \{1, \ldots, n\}$. If $l_i \in \mathcal{V}\mathsf{ar}(r)$ then we let $\tau_i = \{l_i \mapsto s_i\}$. Otherwise $q_i = [l_i]$ and thus $s_i \to_{\Gamma_\mathcal{C}^k}^+ [l_i]$. Part (1) of the first induction hypothesis yields $s_i \in (\to_{\mathcal{R}}^*)[\Sigma(l_i)]$. Hence there exists a substitution τ_i such that $s_i \to_{\mathcal{R}}^* l_i\tau_i$. We assume without loss of generality that $\mathcal{D}\mathsf{om}(\tau_i) = \mathcal{V}\mathsf{ar}(l_i)$. The substitution τ' is defined as $\{x \mapsto u_x \mid x \in \mathcal{V}\mathsf{ar}(r) \setminus \mathcal{V}\mathsf{ar}(l)\}$ where u_x is an arbitrary but fixed ground term such that $u_x \to_{\Gamma_\mathcal{C}^0}^+ x\theta$. (This is possible because all states of Q are accessible.) It remains to show that $s \to_{\mathcal{R}}^* C[l\tau]$ and $C[r\tau] \to_{\Gamma_\mathcal{C}^k}^+ C[q']$. The former is an immediate consequence of the definitions of $\tau_1, \ldots, \tau_n$. For the latter it is sufficient to show that $C[r\tau] \to_{\Gamma_\mathcal{C}^k}^* C[r\theta]$. Let $x \in \mathcal{V}\mathsf{ar}(r)$. If $x \in \mathcal{V}\mathsf{ar}(l)$ then, because $\mathcal{R}$ is growing and left-linear, there is a unique $i \in \{1, \ldots, n\}$ such that $x = l_i$. We have $x\tau = l_i\tau_i = s_i$ by construction of τ_i and $q_i = l_i\theta = x\theta$ by definition. Hence $x\tau = s_i \to_{\Gamma_\mathcal{C}^k}^+ q_i = x\theta$ by assumption.

If $x \notin \mathcal{V}\mathsf{ar}(l)$ then $x\tau = x\tau' \to^+_{\Gamma^0_{\mathcal{C}}} x\theta$ by construction of τ'. This completes the proof. The induction step is summarized in the following diagram:

$$\begin{array}{ccccccc} s & \xrightarrow[\Gamma^k_{\mathcal{C}}]{*} & C[f(q_1,\ldots,q_n)] & \xrightarrow[\Gamma^{k+1}_{\mathcal{C}}]{} & C[q'] & \xrightarrow[\Gamma^{k+1}_{\mathcal{C}}]{*} & q \\ {\scriptstyle\mathcal{R}}\downarrow{\scriptstyle *} & & & & \uparrow{\scriptstyle *}\,{\scriptstyle \Gamma^k_{\mathcal{C}}} & & \\ C[l\tau] & \xrightarrow[\mathcal{R}]{} & C[r\tau] & \xrightarrow[\Gamma^k_{\mathcal{C}}]{*} & C[r\theta] & & \end{array}$$

□

Corollary 5.2.3. $L(\mathcal{C}_T(\mathcal{R})) = (\to^*_{\mathcal{R}})[T]$.

Note that the previous construction can be refined in order to build an automaton with accessible states only. This later construction is implemented in the `Autowrite` tool (see Chapter 15).

For our example $\mathcal{R}$ and $T = \mathsf{NF}(\mathcal{R})$, we obtain the (reduced) set of non-deterministic rules for $\mathcal{C}_{\mathsf{NF}}(\mathcal{R}_{\mathsf{lg}})$ shown in Figure 5.1.

$$\begin{array}{rcl|rcl|rcl}
\mathsf{g}([\mathsf{a}]) & \to & [\mathsf{b}] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[\mathsf{b}]) & \to & [x] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[\mathsf{a}]) & \to & [X] \\
\mathsf{g}([x]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[\mathsf{b}]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[X]) & \to & [X] \\
\mathsf{g}([\mathsf{a}]) & \to & [x] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[\mathsf{b}]) & \to & [x] & \mathsf{f}([\mathsf{a}],[X],[\mathsf{b}]) & \to & [X] \\
\mathsf{g}([\mathsf{a}]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{a}],[\mathsf{a}],[\mathsf{b}]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{a}],[X],[\mathsf{a}]) & \to & [X] \\
\mathsf{g}([\mathsf{b}]) & \to & [x] & \mathsf{g}([\mathsf{a}]) & \to & [X] & \mathsf{f}([\mathsf{a}],[X],[X]) & \to & [X] \\
\mathsf{g}([\mathsf{b}]) & \to & [\mathsf{a}] & \mathsf{g}([\mathsf{b}]) & \to & [X] & \mathsf{f}([X],[\mathsf{b}],[\mathsf{b}]) & \to & [X] \\
\mathsf{g}([X]) & \to & [x] & \mathsf{g}([X]) & \to & [X] & \mathsf{f}([X],[\mathsf{b}],[\mathsf{a}]) & \to & [X] \\
\mathsf{g}([X]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{b}],[X],[\mathsf{a}]) & \to & [x] & \mathsf{f}([X],[\mathsf{b}],[X]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[X]) & \to & [x] & \mathsf{f}([\mathsf{b}],[X],[\mathsf{a}]) & \to & [\mathsf{a}] & \mathsf{f}([X],[\mathsf{a}],[\mathsf{a}]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[X]) & \to & [\mathsf{b}] & \mathsf{f}([\mathsf{b}],[x],[\mathsf{a}]) & \to & [x] & \mathsf{f}([X],[\mathsf{a}],[X]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[x]) & \to & [x] & \mathsf{f}([\mathsf{b}],[x],[\mathsf{a}]) & \to & [\mathsf{a}] & \mathsf{f}([X],[X],[\mathsf{b}]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[x]) & \to & [\mathsf{b}] & \mathsf{f}([\mathsf{b}],[\mathsf{b}],[\mathsf{a}]) & \to & [x] & \mathsf{f}([X],[X],[\mathsf{a}]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[\mathsf{b}]) & \to & [x] & \mathsf{f}([\mathsf{b}],[\mathsf{b}],[\mathsf{a}]) & \to & [\mathsf{a}] & \mathsf{f}([X],[X],[X]) & \to & [X] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[\mathsf{b}]) & \to & [\mathsf{b}] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[\mathsf{a}]) & \to & [x] & \mathsf{g}([x]) & \to & [x] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[\mathsf{a}]) & \to & [x] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[\mathsf{a}]) & \to & [\mathsf{a}] & \mathsf{f}([x],[x],[x]) & \to & [x] \\
\mathsf{f}([\mathsf{a}],[\mathsf{b}],[\mathsf{a}]) & \to & [\mathsf{b}] & \mathsf{f}([\mathsf{b}],[\mathsf{b}],[\mathsf{b}]) & \to & [X] & \mathsf{b} & \to & [x] \\
\mathsf{f}([X],[\mathsf{a}],[\mathsf{b}]) & \to & [x] & \mathsf{f}([\mathsf{b}],[\mathsf{b}],[X]) & \to & [X] & \mathsf{a} & \to & [x] \\
\mathsf{f}([X],[\mathsf{a}],[\mathsf{b}]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{b}],[\mathsf{a}],[X]) & \to & [X] & \bullet & \to & [x] \\
\mathsf{f}([x],[\mathsf{a}],[\mathsf{b}]) & \to & [x] & \mathsf{f}([\mathsf{b}],[X],[\mathsf{b}]) & \to & [X] & \mathsf{b} & \to & [\mathsf{b}] \\
\mathsf{f}([x],[\mathsf{a}],[\mathsf{b}]) & \to & [\mathsf{a}] & \mathsf{f}([\mathsf{b}],[X],[X]) & \to & [X] & \mathsf{a} & \to & [\mathsf{a}]
\end{array}$$

Figure 5.1: Rules of $\mathcal{C}_{\mathsf{NF}}(\mathcal{R}_{\mathsf{lg}})$

We consider the term $t = \mathsf{f}(\Delta,\Delta,\Delta)$ with $\Delta = \mathsf{g}(\mathsf{a})$ and the following computation of $\mathcal{C}_{\mathsf{NF}}(\mathcal{R}_{\mathsf{lg}})$ on the term $t[\bullet]_1$.

$$\mathsf{f}(\bullet,\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})) \to^+ \mathsf{f}([x],\mathsf{g}([\mathsf{a}]),\mathsf{g}([\mathsf{a}])) \to \mathsf{f}([x],[\mathsf{a}],[\mathsf{b}]) \to [\mathsf{a}]$$

As $[\mathsf{a}] \in Q^f$, we have $t[\bullet]_1 \in L(\mathcal{C}_{\mathsf{NF}}(\mathcal{R}_{\mathsf{lg}}))$. By Corollary 5.2.3, we get $t[\bullet]_1 \in (\to^*_{\mathcal{R}_{\mathsf{lg}}})[\mathsf{NF}]$. We conclude that redex Δ at position 1 in t is not $\mathcal{R}_{\mathsf{lg}}$-needed.

Similarly, we can check that the redexes at positions 2 and 3 are not $\mathcal{R}_{\mathsf{lg}}$-needed, so that t is a $\mathcal{R}_{\mathsf{lg}}$-free term.

5.2.2 (left-linear) Growing case

The construction (and proof of correctness) was first given by Nagaya and Toyama in [NT02]. We recall it here as we need the details of it for further analysis in subsequent sections.

The following construction is directly taken from [NT02]. It starts with $\mathcal{D}_T(\mathcal{R}) = (\mathcal{G}, Q_\mathcal{D}, Q^f_\mathcal{D}, \Gamma_\mathcal{D})$ the determinized version of $\mathcal{U}_T(\mathcal{R})$ (constructed in Section 5.1.2):
$Q_\mathcal{D} = 2^Q$, $Q^f = \{S \in Q_\mathcal{D} \mid S \cap Q^f \neq \varnothing\}$ and
$\Gamma_\mathcal{D} = \{f(S_1, \ldots, S_n) \to S \mid \exists q_1, \ldots, q_n, q \in S_1, \ldots S_n, S, f(q_1, \ldots q_n) \to_{\Gamma_\mathcal{U}} q\}$.
We saturate the transition rules $\Gamma_\mathcal{D}$ of $\mathcal{D}_T(\mathcal{R})$ under the following inference rule:

$$\frac{f(S_1, \ldots, S_n) \to S, f(l_1, \ldots, l_n) \to r \in \mathcal{R}, \quad r\theta \to^*_{\Gamma_\mathcal{D}} S', S \subsetneq S \cup S'}{\Gamma_\mathcal{D} = \Gamma_\mathcal{D} \setminus \{f(S_1, \ldots, S_n) \to S\} \cup \{f(S_1, \ldots, S_n) \to S \cup S'\}} \quad (*)$$

with θ mapping the variables in r to states in $Q_\mathcal{D}$ and such that $\forall x \in \mathcal{V}\mathsf{ar}(r)$, if $x = l_j$ for some j then $x\theta = S_j$
otherwise $t \to^*_{\Gamma_\mathcal{D}} x\theta$ for some $t \in \mathcal{T}(\mathcal{G})$.

Because $Q_\mathcal{D}$ is finite and no new state is added by $(*)$, the saturation process terminates.

Upon termination, one has $L(\mathcal{D}_T(\mathcal{R})) = (\to^*_\mathcal{R})[T]$ as shown in [NT02].

Note that the previous construction can be refined in order to build an automaton with accessible states only. This later construction is implemented in the `Autowrite` tool (see Chapter 15).

In our example $\mathcal{R} = \mathcal{R}_\mathsf{g}$. Let us construct $\mathcal{D}_\mathsf{NF}(\mathcal{R})$.
$Q_\mathcal{D} = \{\{[X][x]\}, \{[x]\}, \{[\mathsf{a}][x]\}, \{[\mathsf{b}][x]\}\}$ $\qquad Q^f_\mathcal{D} = Q_\mathcal{D} \setminus \{\{[x]\}\}$.
$\Gamma_\mathcal{D}$ contains 71 deterministic rules which we will not present in detail (but which can be obtained using `Autowrite`, see Chapter 15). Rather we show the computation of $\mathcal{D}_\mathsf{NF}(\mathcal{R})$ on the term $t[\bullet]_1 = \mathsf{f}(\bullet, \Delta, \Delta)$.

$$\mathsf{g}(\mathsf{a}) \to \mathsf{g}(\{[\mathsf{a}][x]\}) \to \{[X][x]\}$$
$$\mathsf{f}(\bullet, \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to \mathsf{f}(\{[x]\}, \{[X][x]\}, \{[X][x]\}) \to \{[x]\}$$

We do not reach a final state which means that $t[\bullet]_1 \notin (\to^*_\mathcal{R})[\mathsf{NF}]$ so that redex $\Delta = \mathsf{g}(\mathsf{a})$ at position 1 in t is not $\mathcal{R}$-needed.

As a side remark we mention that the result described above remains true if we drop the restriction that the left-hand side of a rewrite rule is a non-variable term; just add the following saturation rule:

$$\frac{f(S_1, \ldots, S_n) \to S, x \to r \in \mathcal{R}, \quad r\theta \to^*_{\Gamma_\mathcal{D}} S', S \subsetneq S \cup S'}{\Gamma_\mathcal{D} = \Gamma_\mathcal{D} \setminus \{f(S_1, \ldots, S_n) \to S\} \cup \{f(S_1, \ldots, S_n) \to S \cup S'\}} \quad (*)$$

with $x \in \mathcal{V}$ and θ mapping the variables in r to states in $Q_\mathcal{D}$ and such that $x\theta = S$ and $\forall y \neq x \in \mathcal{V}\mathsf{ar}(r)$, $t \to^*_{\Gamma_\mathcal{D}} x\theta$ for some $t \in \mathcal{T}(\mathcal{G})$.

Although this extension is useless when it comes to call-by-need (because no system that has a rewrite rule whose left-hand side is a single variable has normal forms), it is interesting to note that it generalizes Theorem 5.1 of Coquidé *et al.* [CDGV94]—the preservation of recognizability for linear semi-monadic rewrite systems.

5.3 Recognizability of the set of $\mathcal{R}$-free terms

In this section we assume that $\mathcal{G} = \mathcal{F} \cup \{\bullet\}$ and $T = \mathsf{NF}(\mathcal{R}, \mathcal{F})$, the set of ground normal forms of the growing esystem $\mathcal{R}$. Based on an automaton recognizing $(\to_{\mathcal{R}}^*)[\mathsf{NF}(\mathcal{R}, \mathcal{F})]$ (either the non-deterministic $\mathcal{C}_T(\mathcal{R})$ [or simply $\mathcal{C}$] of Subsection 5.2.1 or the deterministic automaton $\mathcal{D}_T(\mathcal{R})$ [or simply $\mathcal{D}$] of Subsection 5.2.2), we construct an automaton $\mathcal{E}(\mathcal{R})$ that accepts all reducible terms in $\mathcal{T}(\mathcal{F})$ that do not have an $\mathcal{R}$-needed redex *i.e.* the set of $\mathcal{R}$-free terms.

First we take the automaton $\mathcal{B}(\mathcal{R})$ (or simply $\mathcal{B}$) of Subsection 5.1.1 but with signature $\mathcal{F}$ (instead of $\mathcal{F}_\Omega$) and extended in such a way that it can be used to identify redexes and reducible terms with respect to $\mathcal{R}$. More precisely, we add the states $[r]$ (r for reducible) and $[l]$ (l for left-hand side) and the transition rules $f([l_1], \ldots, [l_n]) \to [l]$ and $f([l_1], \ldots, [l_n]) \to [r]$ for every left-hand side $f(l_1, \ldots, l_n)$ of a rewrite rule in $\mathcal{R}$ and all transition rules of the form $f([x], \ldots, [r], \ldots, [x]) \to [r]$ and $f([x], \ldots, [l] \ldots, [x]) \to [r]$. In state $[l]$, all redexes of $\mathcal{T}(\mathcal{F})$ are accepted. In state $[r]$ all reducible terms of $\mathcal{T}(\mathcal{F})$ are accepted.

So we can obtain this version of automaton $\mathcal{B}$ from the one given p50 by removing the rule $\bullet \to [x]$ and adding the following ones:

$$
\begin{array}{rcl@{\qquad}rcl}
\mathsf{g}([x]) & \to & [l] & \mathsf{f}([\mathsf{b}], [x], [\mathsf{a}]) & \to & [r] \\
\mathsf{f}([x], [\mathsf{a}], [\mathsf{b}]) & \to & [l] & \mathsf{f}([\mathsf{a}], [\mathsf{b}], [x]) & \to & [r] \\
\mathsf{f}([\mathsf{b}], [x], [\mathsf{a}]) & \to & [l] & \mathsf{f}([x], [x], [r]) & \to & [r] \\
\mathsf{f}([\mathsf{a}], [\mathsf{b}], [x]) & \to & [l] & \mathsf{f}([x], [r], [x]) & \to & [r] \\
\mathsf{g}([x]) & \to & [r] & \mathsf{f}([r], [x], [x]) & \to & [r] \\
\mathsf{f}([x], [\mathsf{a}], [\mathsf{b}]) & \to & [r] & \mathsf{g}([r]) & \to & [r]
\end{array}
$$

The signature of $\mathcal{E}(\mathcal{R})$ is $\mathcal{F}$. The construction of $\mathcal{E}(\mathcal{R})$ is related to the one in Comon [Com00, Lemma 31]. The main difference is that our constructions apply to the most general classes of linear-growing and growing esystems while Comon's only deals with strong esystems.

We have two slightly different constructions for the two cases, linear-growing and (left-linear) growing.

5.3.1 Linear-growing case

The set $Q_{\mathcal{E}}$ of states of $\mathcal{E}(\mathcal{R})$ consists of all triples $\langle S : B : P \rangle$ where $S \subseteq Q$, $P \subseteq Q$, $B \subseteq Q_{\mathcal{B}}$. The final states are the states $\langle S : B : P \rangle$ such that $[r] \in B$ and $P \subseteq Q^f$. Intuitively, the S component records the behaviour of automaton $\mathcal{C}$, the B component records the behaviour of the automaton $\mathcal{B}$ and

the P component is a subset of S containing for every redex, at least one state reachable with $\mathcal{C}$ when the redex is replaced by $\bullet$. This will be made precise in Lemmata 5.3.2 and 5.3.3 below. The set $\Gamma_{\mathcal{E}}$ consists of all transition rules of the form

$$f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_n : B_n : P_n\rangle) \to \langle S : B : P\rangle$$

where f is an n-ary function symbol in $\mathcal{F}$,
$S = f(S_1, \ldots, S_n)\downarrow_{\mathcal{C}}$, $B = f(B_1, \ldots, B_n)\downarrow_{\mathcal{B}}$, and $P = P' \cup P''$ where P' is a subset of

$$\bigcup_{i=1}^{n} f(S_1, \ldots, P_i, \ldots, S_n)\downarrow_{\mathcal{C}}$$

with the property that for all $i \in \{1, \ldots, n\}$ and $q_i \in P_i$

$$P' \cap f(S_1, \ldots, \{q_i\}, \ldots, S_n)\downarrow_{\mathcal{C}} \neq \varnothing$$

and

$$P'' = \begin{cases} \bullet\downarrow_{\mathcal{C}} = \{[x]\} & \text{if } [l] \in B, \\ \varnothing & \textit{otherwise.} \end{cases}$$

Note that the resulting automaton is non-deterministic due to the freedom of choosing P'. This is essential to obtain a double exponential complexity.

Lemma 5.3.1. *Let $s \in \mathcal{T}(\mathcal{F})$. If $s \to^{+}_{\Gamma_{\mathcal{E}}} \langle S : B : P\rangle$ then $S = s\downarrow_{\mathcal{C}}$ and $B = s\downarrow_{\mathcal{B}}$.*

Proof. Straightforward. □

Lemma 5.3.2. *Let $s \in \mathcal{T}(\mathcal{F})$ such that $s \to^{+}_{\Gamma_{\mathcal{E}}} \langle S : B : P\rangle$. Let $p \in \mathsf{REDEX}(s)$. We have $s[\bullet]_p\downarrow_{\mathcal{C}} \cap P \neq \varnothing$.*

Proof. By induction on the depth of the position p.
If $p = \varepsilon$ then according to Lemma 5.3.1 $[l] \in B$. By construction $P = P' \cup P''$ with $P'' = \bullet\downarrow_{\mathcal{C}} = \{[x]\}$. As $\bullet\downarrow_{\mathcal{C}} = s[\bullet]_{\varepsilon}\downarrow_{\mathcal{C}}$ it follows that $s[\bullet]_p\downarrow_{\mathcal{C}} \cap P = \{[x]\} \neq \varnothing$.
For the induction step we suppose that $s = f(s_1, \ldots, s_n)$ and $p = i{\cdot}p_i$ for some $i \in \{1, \ldots, n\}$. We may write

$$s \to^{*}_{\Gamma_{\mathcal{E}}} f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_i : B_i : P_i\rangle, \ldots) \to_{\Gamma_{\mathcal{E}}} \langle S : B : P\rangle$$

with $P = P' \cup P''$ as defined in the construction.
The induction hypothesis yields some $q_i \in s_i[\bullet]_{p_i}\downarrow_{\mathcal{C}} \cap P_i$.
By construction $f(S_1, \ldots, \{q_i\}, \ldots, S_n)\downarrow_{\mathcal{C}} \cap P' \neq \varnothing$.
It follows that $f(S_1, \ldots, s_i[\bullet]_{p_i}\downarrow_{\mathcal{C}}, \ldots, S_n)\downarrow_{\mathcal{C}} \cap P' \neq \varnothing$.
But $s[\bullet]_p\downarrow_{\mathcal{C}} = f(S_1, \ldots, s_i[\bullet]_{p_i}\downarrow_{\mathcal{C}}, \ldots, S_n)\downarrow_{\mathcal{C}}$ so $s[\bullet]_p\downarrow_{\mathcal{C}} \cap P' \neq \varnothing$. □

Lemma 5.3.3. *Let $s \in \mathcal{T}(\mathcal{F})$. If $P \subseteq \bigcup_{p\in\mathsf{REDEX}(s)} s[\bullet]_p\downarrow_{\mathcal{C}}$ and $\forall p \in \mathsf{REDEX}(s)$, $P \cap s[\bullet]_p\downarrow_{\mathcal{C}} \neq \varnothing$ then $s \to^{+}_{\Gamma_{\mathcal{E}}} \langle S : B : P\rangle$ with $S = s\downarrow_{\mathcal{C}}$ and $B = s\downarrow_{\mathcal{B}}$.*

Proof. By induction on the structure of s.
Base case: s is a constant
The construction gives $P' = \varnothing$. If s is a normal form then $\cup_{p\in\mathsf{REDEX}(s)} s[\bullet]_p\downarrow_{\mathcal{C}} = \varnothing$ so $P = \varnothing$. The construction gives $P'' = \varnothing$. So $P' \cup P'' = \varnothing = P$.
If s is a redex then $P \subseteq \bullet\downarrow_{\mathcal{C}} = \{[x]\}$ and $P \cap \bullet\downarrow_{\mathcal{C}} \neq \varnothing$ implies $P = \bullet\downarrow_{\mathcal{C}} = \{[x]\}$.

As s is a redex, we have $[l] \in B$. The construction gives $P'' = \bullet{\downarrow_{\mathcal{C}}} = \{[x]\}$. So $P' \cup P'' = \{[x]\} = P$. In both cases the contruction yields a rule $s \to_{\Gamma_\mathcal{E}} \langle S : B : P\rangle$.
Induction step: $s = f(s_1, \ldots, s_n)$ for some $n > 0$. Let $S_i = s_i{\downarrow_{\mathcal{C}}}$ and $B_i = s{\downarrow_{\mathcal{B}}}$, for all $i \in \{1, \ldots, n\}$. Define $P^{\neq\varepsilon} = P \cap \bigcup_{p\neq\varepsilon\in \mathsf{REDEX}(s)} s[\bullet]_p{\downarrow_{\mathcal{C}}}$. $P^{=\varepsilon} = P \cap \bullet{\downarrow_{\mathcal{C}}}$ if $\varepsilon \in \mathsf{REDEX}(s)$ and $P^{=\varepsilon} = \varnothing$ otherwise. Define
$P_i = \bigcup_{p_i\in\mathsf{REDEX}(s_i)}\{q_i \in s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}}\ s.t\ f(S1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}} \cap P^{\neq\varepsilon} \neq \varnothing\}$.
Clearly $P_i \subseteq \bigcup_{p_i\in\mathsf{REDEX}(s_i)} s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}}$.
We claim that $\forall p_i \in \mathsf{REDEX}(s_i), P_i \cap s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}} \neq \varnothing$.
Let $p = i{\cdot}p_i$. We have $s[\bullet]_p{\downarrow_{\mathcal{C}}} = f(S_1, \ldots, s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}}, \ldots, S_n){\downarrow_{\mathcal{C}}}$. By assumption, $s[\bullet]_p{\downarrow_{\mathcal{C}}} \cap P \neq \varnothing$ so $s[\bullet]_p{\downarrow_{\mathcal{C}}} \cap P^{\neq\varepsilon} \neq \varnothing$. Hence there exists $q_i \in s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}}$ such that $f(S_1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}} \cap P^{\neq\varepsilon} \neq \varnothing$. By definition, $q_i \in P_i$ which ends the proof of the claim.
The induction hypothesis yields $s_i \to^+_{\Gamma_\mathcal{E}} \langle S_i : B_i : P_i\rangle$. Since this holds for every $i \in \{1, \ldots, n\}$ we obtain $f(s_1, \ldots, s_n) \to^*_{\Gamma_\mathcal{E}} f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_n : B_n : P_n\rangle)$.
It suffices to show that $f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_n : B_n : P_n\rangle) \to \langle S : B : P\rangle$ is a transition rule of $\Gamma_\mathcal{E}$.
We claim that $P^{\neq\varepsilon} \subseteq \bigcup_{i=1}^n f(S_1, \ldots, P_i, \ldots, S_n){\downarrow_{\mathcal{C}}}$.
Let $q \in P^{\neq\varepsilon}$. By definition, $q \in s[\bullet]_p{\downarrow_{\mathcal{C}}}$ for some $p = i{\cdot}p_i$ with $p_i \in \mathsf{REDEX}(s_i)$. So $q \in f(S_1, \ldots, s_i[\bullet]_{p_i}{\downarrow_{\mathcal{C}}}, \ldots, S_n){\downarrow_{\mathcal{C}}}$. Then $\exists q_i \in s_i[\bullet]_{p_i}$ such that
$q \in f(S_1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}}$. So $f(S_1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}} \cap P^{\neq\varepsilon} \neq \varnothing$. This implies by definition that $q_i \in P_i$ so that $q \in f(S_1, \ldots, P_i, \ldots, S_n){\downarrow_{\mathcal{C}}}$ which ends the proof of the claim.
Take $P' = P^{\neq\varepsilon}$ and $P'' = P^{=\varepsilon}$. We have $P = P' \cup P''$. From the previous claim we get $P' \subseteq \bigcup_{i=1}^n f(S_1, \ldots, P_i, \ldots, S_n){\downarrow_{\mathcal{C}}}$. By assumption, $\forall i, \forall q_i \in P_i$, $f(S_1, \ldots, \{q_i\}, \ldots, S_n) \cap P' \neq \varnothing$. $P'' = \bullet{\downarrow_{\mathcal{C}}}$ when s is a redex and $= \varnothing$ otherwise. So all the condition are fulfilled to have a rule $f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_n : B_n : P_n\rangle) \to \langle S : B : P\rangle$. It follows that $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : P\rangle$. □

Corollary 5.3.4. *Let $s \in \mathcal{T}(\mathcal{F})$. $\forall p \in \mathsf{REDEX}(s), s[\bullet]_p{\downarrow_{\mathcal{C}}} \in (\to^*_{\mathcal{R}})[\mathsf{NF}]$ iff $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : P\rangle$ for some $P \subseteq \bigcup_{p\in\mathsf{REDEX}(s)} s[\bullet]_p{\downarrow_{\mathcal{C}}}$ such that $P \cap Q^f \neq \varnothing$.*

Proposition 5.3.5. *Let $s \in \mathcal{T}(\mathcal{F})$. The term $s \in L(\mathcal{E}(\mathcal{R}))$ iff s is reducible and $\forall p \in \mathsf{REDEX}(s), s[\bullet]_p{\downarrow_{\mathcal{C}}} \in (\to^*_{\mathcal{R}})[\mathsf{NF}]$.*

Proof. By Lemma 5.3.1, Corollary 5.3.4 and the observation that s is reducible if and only if $[r] \in s{\downarrow_{\mathcal{B}}}$. □

Theorem 5.3.6. *Let $\mathcal{R}$ be a left-linear system. We have $\mathcal{R} \in \mathsf{CBN}_{\mathsf{lg}}$ if and only if $L(\mathcal{E}(\mathcal{R}_{\mathsf{lg}})) = \varnothing$.*

Proof. Note that $\mathcal{R}_{\mathsf{lg}}$ is a linear-growing esystem. By definition of $\mathsf{CBN}_{\mathsf{lg}}$, $\mathcal{R} \notin \mathsf{CBN}_{\mathsf{lg}}$ if and only if there exists a reducible term s in $\mathcal{T}(\mathcal{F})$ without $\mathcal{R}_{\mathsf{lg}}$-needed redexes. The latter is equivalent to $\forall p \in \mathsf{REDEX}(s)$, $s[\bullet]_p \in (\to^*_{\mathcal{R}_{\mathsf{lg}}})[\mathsf{NF}]$. According to Corollary 5.3.4 this is equivalent to $s \in L(\mathcal{E}(\mathcal{R}_{\mathsf{lg}}))$. Hence $\mathcal{R} \in \mathsf{CBN}_{\mathsf{lg}}$ if and only if $L(\mathcal{E}(\mathcal{R}_{\mathsf{lg}})) = \varnothing$. □

Decidability of membership to $\mathsf{CBN}_{\mathsf{lg}}$ follows.

For our example, we will not attempt to present the corresponding automaton $\mathcal{E}(\mathcal{R}_{\mathsf{lg}})$ which has 25 states and 506 rules (but which can be obtained using `Autowrite`, see Chapter 15). Rather, we show a computation of $\mathcal{E}(\mathcal{R}_{\mathsf{lg}})$ on the term $t = \mathsf{f}(\Delta, \Delta, \Delta)$ with $\Delta = \mathsf{g}(\mathsf{a})$.

$$\mathsf{g}(\mathsf{a}) \to \mathsf{g}(\langle [\mathsf{a}][x] : [\mathsf{a}][x] : \varnothing \rangle) \to \langle [X][\mathsf{a}][\mathsf{b}][x] : [x][r][l] : [x] \rangle$$

$\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to$
$\mathsf{f}(\langle [X][\mathsf{a}][\mathsf{b}][x] : [x][r][l] : [x] \rangle, \langle [X][\mathsf{a}][\mathsf{b}][x] : [x][r][l] : [x] \rangle, \langle [X][\mathsf{a}][\mathsf{b}][x] : [x][r][l] : [x] \rangle)$
$\to \langle [X][\mathsf{a}][\mathsf{b}][x] : [x][r] : [\mathsf{a}][\mathsf{b}] \rangle$

We obtain a final state; this means that t has no $\mathcal{R}_{\mathsf{lg}}$-needed redex which is indeed the case and that $\mathcal{R} \notin \mathsf{CBN}_{\mathsf{lg}}$.

5.3.2 (left-linear)-Growing case

The construction being almost the same as in the linear-growing case, we will only outline the differences. First we use the deterministic automaton $\mathcal{D}$ instead of the non-deterministic one $\mathcal{C}$. The set $Q_{\mathcal{E}}$ of states consists of all triples $\langle S : B : P \rangle$ where $S \in Q_{\mathcal{D}} \subseteq 2^Q$, $P \subseteq Q_{\mathcal{D}}$, $B \subseteq Q_{\mathcal{B}}$. The final states are such that $[r] \in B$ and $P \subseteq Q^f_{\mathcal{D}}$. Intuitively, the component P is a subset of $Q_{\mathcal{D}}$ containing for every redex, the state reachable with $\mathcal{D}$ when the redex is replaced by $\bullet$. This will be made precise in Lemma 5.3.7 below. The only change in the description of $\Gamma_{\mathcal{E}}$ (except that $\mathcal{D}$ is used instead of $\mathcal{C}$) is that

$$P' = \{f(S_1, \ldots, K_i, \ldots, S_n)\downarrow_{\mathcal{D}}, i \in \{1, \ldots, n\}, K_i \in P_i\}$$

Note that the resulting automaton is deterministic.

Similar proofs as in Subsection 5.3.1 (but simpler due to determinism) yield the following lemma.

Lemma 5.3.7. *Let $s \in \mathcal{T}(\mathcal{F})$. $s\downarrow_{\mathcal{E}} = \langle S : B : P \rangle$ iff $S = s\downarrow_{\mathcal{D}}$, $B = s\downarrow_{\mathcal{B}}$ and $P = \{s[\bullet]_p\downarrow_{\mathcal{D}}, p \in \mathsf{REDEX}(s)\}$.*

We obtain the same proposition as Proposition 5.3.5.

Proposition 5.3.8. *Let $s \in \mathcal{T}(\mathcal{F})$. $s \in L(\mathcal{E}(\mathcal{R}))$ if and only if s is reducible and $\forall p \in \mathsf{REDEX}(s)$, $s[\bullet]_p \in (\to^*_{\mathcal{R}})[\mathsf{NF}]$.*

Proof. By Lemma 5.3.1, Lemma 5.3.7 and the observation that s is reducible if and only if $[r] \in s\downarrow_{\mathcal{B}}$. □

Theorem 5.3.9. *Let $\mathcal{R}$ be a left-linear system. We have $\mathcal{R} \in \mathsf{CBN}_{\mathsf{g}}$ if and only if $L(\mathcal{E}(\mathcal{R}_{\mathsf{g}})) = \varnothing$.*

The proof is the same as for Theorem 5.3.6 but using Proposition 5.3.8 instead of Proposition 5.3.5. Decidability of membership to $\mathsf{CBN}_{\mathsf{g}}$ follows.

In our example ($\mathcal{R}_{\mathsf{g}} = \mathcal{R}$), the corresponding reduced automaton $\mathcal{E}(\mathcal{R})$ has 15

states and 3392 rules. It has no final state which means that $\mathcal{R} \in \mathsf{CBN}_{\mathsf{g}}$. We show the computation of $\mathcal{E}(\mathcal{R})$ on the term t.

$$\begin{array}{l}\mathsf{a} \to \langle\{[\mathsf{a}][x]\} : [x][\mathsf{a}] : \varnothing\rangle \\ \mathsf{g}(\mathsf{a}) \to \langle\{[X][x]\} : [x][r][l] : \{[x]\}\rangle \\ \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to \langle\{[X][x]\} : [x][r] : \{[x]\}\rangle\end{array}$$

In the linear-growing case, although we do not have a proof that $\mathcal{D}_T(\mathcal{R})$ coincides with the determinized version of $\mathcal{C}_T(\mathcal{R})$, we were able to check with `Autowrite`(see Chapter 15) that the deterministic automaton $\mathcal{D}_T(\mathcal{R})$ has the same number of states and rules than the determinize version of the non-deterministic automaton $\mathcal{C}_T(\mathcal{R})$.

5.4 Complexity Analysis

In this section, we analyze the complexity of the decision procedures of the previous section. Given a term t, we denote its size (i.e., its total number of symbols) by $|t|$. Given a(n) (e)system $\mathcal{R}$, we denote its size (the sum of the sizes of the left and right-handsides) by $|\mathcal{R}|$ and its number of rules by $\sharp\mathcal{R}$. Let m be the maximum arity of a function symbol in $\mathcal{F}$.

Given an automaton $\mathcal{A} = (\mathcal{F}, Q_{\mathcal{A}}, Q_{\mathcal{A}}^f, \Gamma_{\mathcal{A}})$, its number of transition rules $\sharp\Gamma_{\mathcal{A}}$ is less than $|Q_{\mathcal{A}}|^{m+1}$ so less than $|Q_{\mathcal{A}}|^{|\mathcal{R}|}$.

It is well-known that the minimal number of states of an automaton $\mathcal{A}$ recognizing $T = \mathsf{NF}$ is less than $2^{|\mathcal{R}|}$. $|Q_{\mathcal{B}}|$ is neglectable with regard to $Q_{\mathcal{A}}$. So $|Q| \leq |Q_{\mathcal{A}}| + |Q_{\mathcal{B}}|$ is in $\mathcal{O}(2^{|\mathcal{R}|})$.

Next we analyze the complexity of the saturation processes of Section 5.2. (A similar analysis is reported in [Jac96a, chapitre IV] for the linear-growing case.)

5.4.1 Linear-growing case

Lemma 5.4.1. *Let $\mathcal{R}$ be a linear-growing esystem. The new rules obtained by saturation of $\mathcal{C}_T(\mathcal{R})$ can be computed in $\mathcal{O}(|\mathcal{R}|^2\,|Q|^{4|\mathcal{R}|})$ time.*

Proof. Let Ξ be the set of transition rules that may potentially appear in the automaton $\mathcal{C}_T(\mathcal{R})$: $\Xi = \{f(q_1, \ldots, q_n) \to q \mid f \in \mathcal{F} \text{ and } q_1, \ldots, q_n, q \in Q\}$. Let K be the number of rules in Ξ. We have $K \leq |Q|^{m+1}$. The saturation process may be described by the following algorithm:

$\Xi^0 := \Xi \setminus {\Gamma_{\mathcal{C}}}^0$;
$k := 1$;
`while`
$\quad \exists\, f(q_1, \ldots, q_n) \to q \in \Xi^{k-1}$
$\quad \exists\, f(l_1, \ldots, l_n) \to r \in \mathcal{R}$ such that $r\theta \to^*_{{\Gamma_{\mathcal{C}}}^{k-1}} q$ and, for all $1 \leqslant i \leqslant n$, $q_i = l_i\theta$ if $l_i \in \mathcal{V}\mathsf{ar}(r)$ and $q_i = [l_i]$ otherwise
$\quad \exists\, \theta \colon \mathcal{V}\mathsf{ar}(r) \to Q$
`do`
$\quad \Xi^k := \Xi^{k-1} \setminus \{f(q_1, \ldots, q_n) \to q\}$;

$\Gamma_{\mathcal{C}}{}^{k} := \Gamma_{\mathcal{C}}{}^{k-1} \cup \{f(q_1, \ldots, q_n) \to q\}$;
$k := k + 1$

Let us estimate the time to evaluate the condition of the while-loop. There are $K - \sharp\Gamma_{\mathcal{C}}{}^{k-1}$ choices for $f(q_1, \ldots, q_n) \to q$, $\sharp\mathcal{R}$ choices for $f(l_1, \ldots, l_n) \to r$, and $|Q|^{\mathcal{V}\mathsf{ar}(r)}$ choices for θ. For every choice we have to test whether $r\theta \to^*_{\Gamma_{\mathcal{C}}{}^{k-1}} q$ is true. (The other requirements are neglectable.) This can be done in $\mathcal{O}(|r| \cdot \sharp\Gamma_{\mathcal{C}}{}^{k-1}) \cdot |Q|$ time by simulating a computation of the determinized version of $\Gamma_{\mathcal{C}}{}^{k-1}$: at every position non-variable position u of r, we may consider every transition $f(q_1, \ldots, q_n)$ in Γ_{k-1}, and for each considered transition, we must check for every i, $1 \leq i \leq n$, whether q_i is a member of the set of states computed so far for the position ui.

So one iteration of the while-loop takes

$$\mathcal{O}((K - \sharp\Gamma_{\mathcal{C}}{}^{k-1}) \cdot \sharp\mathcal{R} \cdot |Q|^{\mathcal{V}\mathsf{ar}(r)} \cdot |r| \cdot \sharp\Gamma_{\mathcal{C}}{}^{k-1} \cdot |Q|)$$

time. To obtain the time complexity of the algorithm we have to multiply this by the maximum number of iterations, which is $K - \sharp\Gamma_{\mathcal{C}}{}^{0}$. Removing the negative terms and estimating $\sharp\Gamma_{\mathcal{C}}{}^{k-1}$ by K and $\mathcal{V}\mathsf{ar}(r)$ by $|r|$ yields

$$\mathcal{O}(K^3 \cdot \sharp\mathcal{R} \cdot |Q|^{|r|+1} \cdot |r|)$$

Estimating $\sharp\mathcal{R}$ and $|r| + 1$ by $|\mathcal{R}|$ and K by $|Q|^{|\mathcal{R}|}$ yields the complexity class $\mathcal{O}(|\mathcal{R}|^2 \, |Q|^{4|\mathcal{R}|})$ in the statement of the lemma. □

As $|Q|$ is in $\mathcal{O}(2^{|\mathcal{R}|})$ and from Lemma 5.4.1 the time to compute the saturation rules is in $\mathcal{O}(2^{\mathcal{O}(|\mathcal{R}|^2)})$. For $\mathcal{R}_{\mathsf{s}}$ and $\mathcal{R}_{\mathsf{nv}}$ we get a polynomial time complexity, but the space and time complexity of the automaton $\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$ is still exponential in $|\mathcal{R}|$ due to the normal form automaton.

Lemma 5.4.2. *The automaton $\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$ can be computed in $2^{\mathcal{O}(|\mathcal{R}|^2)}$.*

Proof. The time to compute $\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$ is the sum of the times to compute the automaton $\mathcal{A}_{\mathsf{NF}}$ $2^{\mathcal{O}(|\mathcal{R}|^2)}$, the automaton $\mathcal{B}(\mathcal{R})$ (neglectable) and the saturation rules $2^{\mathcal{O}(|\mathcal{R}|^2)}$. This yields an $2^{\mathcal{O}(|\mathcal{R}|^2)}$ time complexity. □

Finally, let us consider the construction of $\mathcal{E}(\mathcal{R})$.

Lemma 5.4.3. *$\sharp\Gamma_{\mathcal{E}}$ is in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$.*

Proof. $|Q_{\mathcal{E}}| \leq 2^{|Q|}$ so $|Q_{\mathcal{E}}|$ is in $\mathcal{O}(2^{2^{|\mathcal{R}|}})$. $\sharp\Gamma_{\mathcal{E}} \leq |Q_{\mathcal{E}}|^{|\mathcal{R}|}$ so $\sharp\Gamma_{\mathcal{E}}$ is in $\mathcal{O}(2^{|Q|^{|\mathcal{R}|}})$ where $|Q|$ is in $\mathcal{O}(2^{|\mathcal{R}|})$. We conclude that $\sharp\Gamma_{\mathcal{E}}$ is in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$. □

Lemma 5.4.4. *Let $\mathcal{R}$ be a linear-growing esystem. Using the automata $\mathcal{C}_T(\mathcal{R})$ and $\mathcal{B}(\mathcal{R})$, the rules of $\Gamma_{\mathcal{E}}$ can be computed in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$ time.*

Proof. Consider a rule $f(\langle S_1 : B_1 : P_1\rangle, \ldots, \langle S_n : B_n : P_n\rangle) \to \langle S : B : P\rangle$. To check whether it is a rule of $\Gamma_{\mathcal{E}}$ we have to check whether $f(S_1, \ldots, S_n) = S{\downarrow_{\mathcal{C}}}$ [takes time $\mathcal{O}(|Q|^n \cdot \sharp\Gamma_{\mathcal{C}})$], $f(B_1, \ldots, B_n) = S{\downarrow_{\mathcal{B}}}$ [neglectable] and that P satisfies the conditions of the construction. We need to compute $\bigcup_{i=1}^{n} f(S_1, \ldots, P_i, \ldots, S_n){\downarrow_{\mathcal{C}}}$ [takes time $\mathcal{O}(|Q|^n \cdot \sharp\Gamma_{\mathcal{C}})$] and to check that $P \subseteq S' = \bigcup_{i=1}^{n} f(S_1, \ldots, P_i, \ldots, S_n){\downarrow_{\mathcal{C}}}$ [takes time $\mathcal{O}(|Q|^2)$]. If $[l] \in B$, we have to check that $\bullet{\downarrow_{\mathcal{C}}} = \{[x]\} \subseteq P$ [neglectable].

We are left to check that $\forall i \in \{1, \ldots, n\}$ and $q_i \in P_i$, we have

$$f(S_1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}} \cap P \neq \varnothing.$$

We claim this can be done in $|Q|^{\mathcal{O}(|\mathcal{R}|)}$.

There are at most $n\,|Q|$ of these checks and each one takes the time to compute $f(S_1, \ldots, \{q_i\}, \ldots, S_n){\downarrow_{\mathcal{C}}}$ $\mathcal{O}(|Q|^n \cdot \sharp\Gamma_{\mathcal{C}})$ plus the time to compute the intersection $\mathcal{O}(|Q|^2)$. So we get $\mathcal{O}(n\,|Q|\,(|Q|^n\,\sharp\Gamma_{\mathcal{C}} + |Q|^2))$ (estimating $\sharp\Gamma_{\mathcal{C}}$ by $|Q|^{|\mathcal{R}|}$ as at the end of Lemma 5.4.1). Now, estimating n by $|\mathcal{R}|$ we obtain the claim.

To get the total time we multiply by the number of rules of $\Gamma_{\mathcal{E}}$ [$2^{2^{\mathcal{O}(|\mathcal{R}|)}}$ by Lemma 5.4.3] which gives $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$. □

Lemma 5.4.5. *Given a linear-growing esystem $\mathcal{R}$, $\mathcal{E}(\mathcal{R})$ can be computed in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$.*

Proof. The time to build $\mathcal{E}$ is the time to build $\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$ plus the time to compute $\Gamma_{\mathcal{E}}$. The former is neglectable with respect to the latter, which can be done in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$ time by Lemma 5.4.4. □

As emptiness can be decided in polynomial time with respect to the size of the automaton, we conclude with the following theorem.

Theorem 5.4.6. *It can be decided in double exponential time whether a linear-growing system belongs to* $\mathsf{CBN}_{\mathsf{lg}}$.

Although the saturation process for $\mathcal{R}_{\mathsf{s}}$ and $\mathcal{R}_{\mathsf{nv}}$ is much simpler, our construction does not give better complexity results for deciding membership to $\mathsf{CBN}_{\mathsf{s}}$ or $\mathsf{CBN}_{\mathsf{nv}}$. Nevertheless, for $\mathsf{CBN}_{\mathsf{s}}$ (which almost coincides with the class of strongly sequential systems, see Chapter 9), a smaller complexity bound is known: Comon [Com00] showed that it can be decided in exponential time whether a left-linear system is strongly sequential. He uses an automaton for ω-reduction which plays the same role as our $\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$ automaton. Since there is no satisfactory notion of ω-reduction corresponding to the approximation mapping nv, it remains to be seen whether the result of Theorem 5.4.7 can be improved; a possible track could be to analyze whether the gtt-construction of page 33 gives a better complexity.

5.4.2 (left-linear) Growing case

A similar analysis yields a construction of $\mathcal{D}_{\mathcal{R}}(\mathsf{NF})$ in $2^{2^{\mathcal{O}(|\mathcal{R}|)}}$ and a construction of $\mathcal{E}(\mathcal{R})$ in $2^{2^{2^{\mathcal{O}(|\mathcal{R}|)}}}$. The extra exponential does not come from C components

of the states (which have the same size as in the linear case) but from the P components which are subsets of $Q_{\mathcal{D}}$ (compared to subsets of Q in the linear case).

Theorem 5.4.7. *It can be decided in triple exponential time whether a left-linear growing system belongs to* $\mathsf{CBN}_{\mathtt{g}}$.

Unfortunately, this result does not give a better upperbound than the one that could be obtained by expressing membership to $\mathsf{CBN}_{\mathtt{g}}$ by a weak second-order monadic formula similarly as done for $\mathsf{CBN\text{-}RS}_{\mathtt{g}}$ in the proof of Proposition 6.2.2. However, the construction which has been implemented in `Autowrite` [Dur02] shows that in practice the complexity is often significantly less than the worst case.

5.4.3 Summary of the complexity results

The following arrays recapitulate the complexity results used or obtained in this section.

	Size	Time
$\mathcal{A}_{\mathsf{NF}(\mathcal{R})}$	$\mathcal{O}(2^{\lvert\mathcal{R}\rvert})$	$\mathcal{O}(2^{\lvert\mathcal{R}\rvert})$
$\mathcal{B}(\mathcal{R})$	$\mathcal{O}(\mathcal{R})$	$\mathcal{O}(\mathcal{R})$

	Linear-growing		Growing	
	Size	Time	Size	Time
$\mathcal{C}_{\mathsf{NF}}(\mathcal{R})$	$2^{\mathcal{O}(\lvert\mathcal{R}\rvert^2)}$	$2^{\mathcal{O}(\lvert\mathcal{R}\rvert^2)}$		
$\mathcal{D}_{\mathsf{NF}}(\mathcal{R})$			$2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}$	$2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}$
$\mathcal{E}(\mathcal{R})$	$2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}$	$2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}$	$2^{2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}}$	$2^{2^{2^{\mathcal{O}(\lvert\mathcal{R}\rvert)}}}$

Chapter 6

Computations to root-stable forms

In this chapter, we consider call-by-need computations to root-stable forms. In [Mid97] it is shown that *root-neededness* is more fundamental than neededness when it comes to infinitary normalization. However, root-stability is undecidable and, unlike neededness, root-neededness of a redex is not determined by its position. This considerably complicates the quest for a computable call-by-need strategy to root-stable forms. Preliminary results were given in [DM97].

The following example is entirely taken from [Mid97].

Example 6.0.8. *By definition every redex in a term that has no normal form is needed. Hence the theory of needed reduction is not useful for terms that have an infinite normal form. Consider for instance the following rewrite rules implementing the Sieve of Erathostenes for generating the infinite list of all prime numbers:*

$$\begin{array}{lcl}
\mathsf{primes} & \to & \mathsf{sieve}(\mathsf{from}(2)) \\
\mathsf{from}(x) & \to & x : \mathsf{from}(x+1) \\
\mathsf{sieve}(x:y) & \to & x : \mathsf{sieve}(\mathsf{filter}(x,y)) \\
\mathsf{filter}(x, y:z) & \to & \mathsf{if}(x|y, \mathsf{filter}(x,z), y : \mathsf{filter}(x,z)) \\
\mathsf{if}(\mathsf{true}, x, y) & \to & x \\
\mathsf{if}(\mathsf{false}, x, y) & \to & y
\end{array}$$

*Assuming, that the addition $x+y$ and $x|y$ (is x a divisor of y?) are built-in operations which return respectively an integer and a boolean value (*true *or* false*), the evaluation of* primes *starts as follows:*

$$\begin{array}{lcl}
\mathsf{primes} & \to & \mathsf{sieve}(\mathsf{from}(2)) \\
 & \to & \mathsf{sieve}(2 : \mathsf{from}(3))
\end{array}$$

At this point, there are two ways to proceed. Either we contract $\mathsf{sieve}(2 : \mathsf{from}(3))$ *using the third rule resulting in* $2 : \mathsf{sieve}(\mathsf{filter}(2, \mathsf{from}(3)))$*, or we contract the*

redex $\mathsf{from}(3)$ *using the secong rule, resulting in* $\mathsf{sieve}(2 : 3 : \mathsf{from}(4))$. *Because the term* $\mathsf{sieve}(2 : \mathsf{from}(3))$ *doesn't have a (finite) normal form, both redexes are trivially needed. Consequently, the theory of needed computation cannot distinghuish between the meaningless computation*

$$\begin{array}{rll} \mathsf{primes} & \to^* & \mathsf{sieve}(2 : \mathsf{from}(3)) \\ & \to & \mathsf{sieve}(2 : 3 : \mathsf{from}(4)) \\ & \to & \mathsf{sieve}(2 : 3 : 4 : \mathsf{from}(5)) \\ & \to & \ldots \end{array}$$

from the infinite computation

$$\begin{array}{rll} \mathsf{primes} & \to^* & \mathsf{sieve}(2 : \mathsf{from}(3)) \\ & \to & 2 : \mathsf{sieve}(\mathsf{filter}(2, \mathsf{from}(3))) \\ & \to^* & 2 : 3 : \mathsf{sieve}(\ldots)) \\ & \to & \ldots \end{array}$$

whose limit is the infinite list $2 : 3 : 5 : 7 : \ldots$ *of primes numbers.*

The concept of needed redex for computing normal forms extends naturally to the concept of root-needed redex for computing root-stable forms.

Definition 6.0.9. *Given a term rewriting system and a non-root-stable term* t*, a redex in* t *is* root-needed *if it is contracted in every rewrite sequence from* t *to root-stable form.*

6.1 Decidable approximations of root-neededness

In the remainder of this chapter we generalize the results concerning call-by-need computations to normal form to call-by-need computations to root-stable forms. The first problem we face is to find a characterization of root-needed redex that doesn't depend on the notion of descendant. This is less trivial than it seems, because the obvious adaptation

> redex Δ in term $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is root-needed if and only if there is no term $t \in \mathsf{RS}_{\mathcal{R}_\bullet}$ such that $C[\bullet] \to^*_{\mathcal{R}} t$

of Lemma 3.2.1 doesn't work.

Example 6.1.1. *For instance, redex* a *in the term* $\mathsf{f}(\mathsf{a})$ *is root-needed with respect to the system* $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{f}(\mathsf{b}) \to \mathsf{c}\}$ *but* $\mathsf{f}(\bullet)$ *is root-stable with respect to* $\mathcal{R}_\bullet$*. On the other hand, redex* $\mathsf{f}(\mathsf{b})$ *in the term* $\mathsf{f}(\mathsf{f}(\mathsf{b}))$ *is not root-needed as* $\mathsf{f}(\mathsf{f}(\mathsf{b}))$ *is root-stable.*

This shows that root-needed redexes, unlike needed redexes, are not uniform: root-neededness of a redex doesn't depend only on its position in a term. Hence when trying to determine whether a redex Δ is root-needed we cannot simply replace it by $\bullet$. Instead, we mark the root symbol of Δ.

Definition 6.1.2. *Let $\mathcal{F}$ be a signature. Let $\mathcal{F}_\circ = \mathcal{F} \cup \{f_\circ \mid f \in \mathcal{F}\}$ with every $f_\circ$ having the same arity as f. Let $(\mathcal{R}, \mathcal{F})$ be a system. Let $\Delta \in \mathcal{T}(\mathcal{F})$ be a redex. We write Δ° for the term that is obtained from Δ by marking its root symbol, i.e., if $\Delta = f(t_1, \ldots, t_n)$ then $\Delta^\circ = f_\circ(t_1, \ldots, t_n)$. We call Δ° a* marked redex. *The mapping from $\mathcal{T}(\mathcal{F}_\circ)$ to $\mathcal{T}(\mathcal{F})$ that simply erases all marks is denoted by* erase, *so $\mathsf{erase}(f(t_1, \ldots, t_n)) = \mathsf{erase}(f_\circ(t_1, \ldots, t_n)) = f(\mathsf{erase}(t_1), \ldots, \mathsf{erase}(t_n))$.*

The marking of redexes serves a double purpose. On one hand, it tells us which redex we want to test for root-neededness. On the other hand, marked redexes are not redexes—simply because there are no marks in the (left-hand sides of the) rewrite rules—so if we rewrite a term that contains a marked redex, the marked redex is never contracted.

Lemma 6.1.3. *Let $\mathcal{R}$ be an orthogonal system over a signature $\mathcal{F}$. Redex Δ in term $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is root-needed if and only if there is no term t such that $C[\Delta^\circ] \to_\mathcal{R}^* t$ and $\mathsf{erase}(t) \in \mathsf{RS}_\mathcal{R}$.*

Proof.

$\Rightarrow$ Suppose there is a term t such that $C[\Delta^\circ] \to_\mathcal{R}^* t$ and $\mathsf{erase}(t) \in \mathsf{RS}_\mathcal{R}$. Erasing all marks yields a sequence $A\colon C[\Delta] = \mathsf{erase}(C[\Delta^\circ]) \to_\mathcal{R}^* \mathsf{erase}(t)$ with $\Delta \perp A$. Hence Δ is not root-needed.

$\Leftarrow$ Suppose Δ is not root-needed.
Then there exists a rewrite sequence $A\colon C[\Delta] \to_\mathcal{R}^* t$ with t root-stable and $\Delta \perp A$. Marking every descendant of Δ in A yields a sequence $C[\Delta^\circ] \to_\mathcal{R}^* t'$ with $\mathsf{erase}(t') = t \in \mathsf{RS}_\mathcal{R}$.

□

Let us check that this characterization of root-stable redexes behaves adequaletly with Example 6.1.1.

- $\mathsf{f}(\mathsf{a}_\circ)$ cannot be rewritten and $\mathsf{erase}(\mathsf{f}(\mathsf{a}_\circ)) = \mathsf{f}(\mathsf{a}) \notin \mathsf{RS}_\mathcal{R}$ so redex a is root-needed.

- $\mathsf{f}(\mathsf{f}_\circ(\mathsf{b}))$ cannot be rewritten and $\mathsf{erase}(\mathsf{f}(\mathsf{f}_\circ(\mathsf{b}))) = \mathsf{f}(\mathsf{f}(\mathsf{b})) \in \mathsf{RS}_\mathcal{R}$ so redex $\mathsf{f}(\mathsf{b})$ is not root-needed.

Definition 6.1.4. *Given a system $(\mathcal{R}, \mathcal{F})$, $\mathcal{R}_\circ$ denotes the system $\mathcal{R} \cup \{l^\circ \to r \mid l \to r \in \mathcal{R}\}$ over the signature $\mathcal{F}_\circ$.*

Actually, we can do without the relation erase because for orthogonal systems $\mathcal{R}$ it is not difficult to prove that $\{t \in \mathcal{T}(\mathcal{F}_\circ) \mid \mathsf{erase}(t) \in \mathsf{RS}_\mathcal{R}\}$ coincides with $\mathsf{RS}_{\mathcal{R}_\circ}$. Hence redex Δ in term $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is root-needed if and only if there is no term $t \in \mathsf{RS}_{\mathcal{R}_\circ}$ such that $C[\Delta^\circ] \to_\mathcal{R}^* t$.

Besides the problem that root-stability is in general not computable, we face the problem of the non-computability of $\to_\mathcal{R}^*$ as in the case of computations to normal form. So in order to arrive at decidable approximations of root-neededness, we approximate $\mathcal{R}$ by two esystems $\mathcal{S}_1$ and $\mathcal{S}_2$ such that it is decidable whether a term has an $\mathcal{S}_1$-reduct in $\mathsf{RS}_{(\mathcal{S}_2)_\circ}$.

Definition 6.1.5. *Let $(\mathcal{S}_1, \mathcal{F})$ and $(\mathcal{S}_2, \mathcal{F})$ be left-linear esystems. We say that redex Δ in $C[\Delta] \in \mathcal{T}(\mathcal{F})$ is* $(\mathcal{S}_1, \mathcal{S}_2)$-root-needed *if there is no term $t \in \mathsf{RS}_{(\mathcal{S}_2)_\circ}$ such that $C[\Delta^\circ] \to^*_{\mathcal{S}_1} t$. We abbreviate $(\mathcal{S}, \mathcal{S})$-root-needed to $\mathcal{S}$-root-needed.*

Our $\mathcal{R}_\mathsf{s}$-root-needed redexes coincide with the *strongly root-needed* redexes of Kennaway [Ken95].

Lemma 6.1.6. *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be left-linear systems over the same signature with approximations $\mathcal{S}_1$ and $\mathcal{S}_2$. Every $(\mathcal{S}_1, \mathcal{R}_2)$-root-needed redex is $(\mathcal{R}_1, \mathcal{S}_2)$-root-needed.*

Proof. First we show that $\mathsf{RS}_{(\mathcal{S}_2)_\circ} \subseteq \mathsf{RS}_{(\mathcal{R}_2)_\circ}$. If $t \notin \mathsf{RS}_{(\mathcal{R}_2)_\circ}$ then there exists an $(\mathcal{R}_2)_\circ$-redex Δ such that $t \to^*_{(\mathcal{R}_2)_\circ} \Delta$. Because $\mathcal{S}_2$ approximates $\mathcal{R}_2$ we have $\to^*_{\mathcal{R}_2} \subseteq \to^*_{\mathcal{S}_2}$ and $\mathsf{NF}_{\mathcal{R}_2} = \mathsf{NF}_{\mathcal{S}_2}$. The latter implies $\mathsf{NF}_{(\mathcal{R}_2)_\circ} = \mathsf{NF}_{(\mathcal{S}_2)_\circ}$ and thus $(\mathcal{R}_2)_\circ$ and $(\mathcal{S}_2)_\circ$ have the same redexes. Hence $t \to^*_{(\mathcal{S}_2)_\circ} \Delta$ with Δ an $(\mathcal{S}_2)_\circ$-redex and therefore $t \notin \mathsf{RS}_{(\mathcal{S}_2)_\circ}$. Combining the inclusion $\mathsf{RS}_{(\mathcal{S}_2)_\circ} \subseteq \mathsf{RS}_{(\mathcal{R}_2)_\circ}$ with $\to^*_{\mathcal{R}_1} \subseteq \to^*_{\mathcal{S}_1}$ immediately yields the desired result. □

It should be noted that, for an approximation $\mathcal{S}$ of $\mathcal{R}$, $\mathcal{S}$-root-needed redexes need not be $\mathcal{R}$-root-needed. Consider for instance the orthogonal system $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{f}(\mathsf{c}) \to \mathsf{c}\}$ and its strong approximation $\mathcal{R} = \{\mathsf{a} \to x, \mathsf{f}(\mathsf{c}) \to x\}$.

The term $\mathsf{f}(\mathsf{a})$ is root-stable, hence its redex a is not root-needed. Nevertheless, redex a is $\mathcal{R}_\mathsf{s}$-root-needed because the only term t such that $\mathsf{f}(\mathsf{a}_\circ) \to^*_{\mathcal{R}_\mathsf{s}} t$ is $\mathsf{f}(\mathsf{a}_\circ)$ itself and $\mathsf{f}(\mathsf{a}_\circ) \notin \mathsf{RS}_{(\mathcal{R}_\mathsf{s})_\circ}$ because we have $\mathsf{f}(\mathsf{a}_\circ) \to \mathsf{f}(\mathsf{c})$ in the system $(\mathcal{R}_\mathsf{s})_\circ = \{\mathsf{a} \to x, \mathsf{a}_\circ \to x, \mathsf{f}(\mathsf{c}) \to x, \mathsf{f}_\circ(\mathsf{c}) \to x\}$. So not every strongly root-needed redex is root-needed, contradicting Theorem 16 in [Ken95].

Definition 6.1.7. *Let $(\mathcal{R}, \mathcal{F})$ and $(\mathcal{S}, \mathcal{F})$ be left-linear esystems. Let $\mathsf{M}^\circ_\mathcal{R} = \{C[\Delta^\circ] \mid C[\Delta] \in \mathcal{T}(\mathcal{F})$ and Δ is an $\mathcal{R}$-redex$\}$, the set of all terms that contain exactly one marked redex.*
The set of all terms $C[\Delta^\circ] \in \mathsf{M}^\circ_\mathcal{R}$ such that there is no term $t \in \mathsf{RS}_{\mathcal{S}_\circ}$ with $C[\Delta^\circ] \to^*_\mathcal{R} t$ is denoted by $(\mathcal{R}, \mathcal{S})$-ROOT-NEEDED.

Theorem 6.1.8. *Let $(\mathcal{R}, \mathcal{F})$ and $(\mathcal{S}, \mathcal{F})$ be left-linear esystems. If $(\to^*_\mathcal{R})[\mathsf{RS}_{\mathcal{S}_\circ}]$ is recognizable then $(\mathcal{R}, \mathcal{S})$-ROOT-NEEDED is recognizable.*

Proof. We have $(\mathcal{R}, \mathcal{S})$-ROOT-NEEDED $= (\to^*_\mathcal{R})[\mathsf{RS}_{\mathcal{S}_\circ}]^c \cap \mathsf{M}^\circ_\mathcal{R}$. We show that $\mathsf{M}^\circ_\mathcal{R}$ is recognizable. Let $\mathcal{A}$ be a term automaton that accepts $\mathsf{REDEX}_\mathcal{R} \cap \mathcal{T}(\mathcal{F})$ such that $\mathcal{A}$ has a unique final state [!], a state $[x]$ in which all terms in $\mathcal{T}(\mathcal{F})$ are accepted, and for every left-hand side $f(l_1, \ldots, l_n)$ of a rewrite rule in $\mathcal{R}$ a single transition rule of the form $f(q_1, \ldots, q_n) \to$ [!]. We transform $\mathcal{A}$ into a term automaton $\mathcal{B}$ by changing the latter transition rules to $\mathsf{f}_\circ(q_1, \ldots, q_n) \to$ [!] and adding all transition rules of the form $f([x], \ldots, [!], \ldots, [x]) \to$ [!] with $f \in \mathcal{F}$. It is not difficult to see that $\mathcal{B}$ accepts $\mathsf{M}^\circ_\mathcal{R}$. Hence the desired result follows from Lemma 2.4.1. □

For the example system $\mathcal{R}$ on page 33 a term automaton that accepts $\mathsf{M}^\circ_\mathcal{R}$ is presented in Table 6.1. In the remainder of this subsection we show that the first premise of Theorem 6.1.8 is satisfied for the four approximations defined in chapter 3.

a	$\to$	$[x]$	a	$\to$	$[\mathsf{a}]$
b	$\to$	$[x]$	b	$\to$	$[\mathsf{b}]$
$\mathsf{f}([x],[x])$	$\to$	$[x]$	$\mathsf{g}([x])$	$\to$	$[\mathsf{g}(x)]$
$\mathsf{g}([x])$	$\to$	$[x]$	$\mathsf{f}([x],[\mathsf{b}])$	$\to$	$[\mathsf{f}(x,\mathsf{b})]$
$\mathsf{h}([x])$	$\to$	$[x]$			
			$\mathsf{g}([!])$	$\to$	$[!]$
$\mathsf{f}_\circ([\mathsf{g}(x)],[a])$	$\to$	$[!]$	$\mathsf{h}([!])$	$\to$	$[!]$
$\mathsf{h}_\circ([\mathsf{a}])$	$\to$	$[!]$	$\mathsf{f}([!],[x])$	$\to$	$[!]$
$\mathsf{h}_\circ([\mathsf{f}(x,\mathsf{b})])$	$\to$	$[!]$	$\mathsf{f}([x],[!])$	$\to$	$[!]$

Table 6.1: A term automaton that accepts $\mathsf{M}^\circ_{\mathcal{R}}$.

Lemma 6.1.9. *Let $\mathcal{R}$ be a left-linear system and α be a recognizability preserving approximation mapping. The set $\mathsf{RS}_{(\mathcal{R}_\alpha)}$ is recognizable.*

Proof. We have $\mathsf{RS}_{(\mathcal{R}_\alpha)} = (\to^*_{(\mathcal{R}_\alpha)})[\mathsf{REDEX}_{(\mathcal{R}_\alpha)}]^c$. Because $(\mathcal{R}_\alpha)$ is left-linear, $\mathsf{REDEX}_{(\mathcal{R}_\alpha)}$ is recognizable according to Lemma 2.4.2. As α is regularity preserving, we get that $(\to^*_{(\mathcal{R}_\alpha)})[\mathsf{REDEX}_{(\mathcal{R}_\alpha)}]^c$ is recognizable. □

Lemma 6.1.10. *Let $\mathcal{R}$ be a left-linear system and α,β be two recognizability preserving approximation mappings. The set $(\to^*_{\mathcal{R}_\alpha})[\mathsf{RS}_{(\mathcal{R}_\beta)_\circ}]$ is recognizable.*

Proof. As β is recognizability preserving, lemma 6.1.9 yields that $\mathsf{RS}_{(\mathcal{R}_\circ)_\beta}$ is recognizable. As α is recognizability preserving, we get that $(\to^*_{\mathcal{R}_\alpha})[\mathsf{RS}_{(\mathcal{R}_\circ)_\beta}]$ is recognizable. It is easy to see that $(\mathcal{R}_\circ)_\beta = (\mathcal{R}_\beta)_\circ$. We conclude that $(\to^*_{\mathcal{R}_\alpha})[\mathsf{RS}_{(\mathcal{R}_\beta)_\circ}]$ is recognizable. □

Corollary 6.1.11. *Let $(\mathcal{R},\mathcal{F})$ be a left-linear system and α,β be two recognizability preserving approximation mappings. It is decidable whether a redex in a term in $\mathcal{T}(\mathcal{F})$ is $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed.*

Corollary 6.1.12. *Let $(\mathcal{R},\mathcal{F})$ be a left-linear system. It is decidable whether a redex in a term in $\mathcal{T}(\mathcal{F})$ is $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed for $\alpha,\beta \in \{\mathsf{s},\mathsf{nv},\mathsf{lg},\mathsf{g}\}$.*

6.2 Call-by-need computations to root-stable forms

Definition 6.2.1. *Let α and β be approximation mappings. The class of systems $(\mathcal{R},\mathcal{F})$ such that every non-$\mathcal{R}_\beta$-root-stable term in $\mathcal{T}(\mathcal{F})$ has an $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed redex is denoted by $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$.*

It can be shown that $\mathsf{CBN\text{-}RS}_{\mathsf{s},\mathsf{s}}$ coincides with the class of strongly root-sequential systems introduced by Kennaway [Ken95].

The following Theorem is the counterpart of Theorem 3.4.4. The proof, however, is more difficult because the parallel closure of $\{(\Delta,\Delta^\circ) \mid \Delta \in \mathcal{T}(\mathcal{F}) \text{ is a redex}\}$ is not gtt-recognizable since the size of redexes is unbounded and in GTTs one can only transfer a finite amount of information between the two sides. We overcome this problem by resorting to weak second-order monadic logic.

Proposition 6.2.2. *Let $\mathcal{R}$ and $\mathcal{S}$ be left-linear esystems such that $(\mathcal{R},\mathcal{S})$-*ROOT-NEEDED *is recognizable. The set of terms that have an $(\mathcal{R},\mathcal{S})$-root-needed redex is recognizable.*

Proof. Let $\phi(\vec{X})$ be the definition of $(\mathcal{R},\mathcal{S})$-ROOT-NEEDED in WS$k$S, whose existence is guaranteed by Theorem 2.4.4. The set of terms that have an $(\mathcal{R},\mathcal{S})$-root-needed redex is defined by the following WSkS formula:

$$\overbrace{\bigwedge_{f\in\mathcal{F}_\circ\setminus\mathcal{F}} X_f=\varnothing}^{(1)} \;\wedge\; \exists\vec{Y}\left[\overbrace{X=Y \;\wedge\; \bigwedge_{f\in\mathcal{F}} X_f=Y_f\cup Y_{f_\circ}}^{(2)} \;\wedge\; \overbrace{\phi(\vec{Y})}^{(3)}\right].$$

Part (1) ensures that the term t encoded by $\vec{X}$ contains no marks, part (2) ensures that $\mathsf{erase}(t')=t$ for the term t' encoded by $\vec{Y}$, and part (3) ensures that $t'\in(\mathcal{R},\mathcal{S})$-ROOT-NEEDED. Hence the result follows from Theorem 2.4.4. □

Lemma 6.2.3. *Let $(\mathcal{R},\mathcal{F})$ be a left-linear system and α,β be two recognizability preserving approximation mappings. The set of terms that have an $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed redex is recognizable.*

Proof. From Lemma 6.1.10, $(\to^*_{\mathcal{R}_\alpha})[\mathsf{RS}_{(\mathcal{R}_\beta)_\circ}]$. From Lemma 6.1.8, $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-ROOT-NEEDED is recognizable. The results follows from Proposition 6.2.2. □

Theorem 6.2.4. *Let $\mathcal{R}$ be a left-linear system and α, β be two recognizability preserving approximation mappings. It is decidable whether $\mathcal{R}\in$ CBN-RS$_{\alpha,\beta}$.*

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R}$. The system $\mathcal{R}$ belongs to CBN-RS$_{\alpha,\beta}$ if and only if the set

$$A=(\mathcal{T}(\mathcal{F})\setminus\mathsf{RS}_{\mathcal{R}_\beta})\setminus\{t\in\mathcal{T}(\mathcal{F})\mid t \text{ has an } (\mathcal{R}_\alpha,\mathcal{R}_\beta)\text{-root-needed redex}\}$$

is empty.

From Lemma 6.1.9, $\mathsf{RS}_{\mathcal{R}_\beta}$ is recognizable. From Lemma 6.2.3), $\{t\in\mathcal{T}(\mathcal{F})\mid t$ has an $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed redex$\}$ is recognizable. $\mathcal{T}(\mathcal{F})$ is known to be recognizable. So A is a boolean combination of recognizable sets of terms. From Lemma 2.4.1(1), A is recognizable. Emptiness of A is decidable by the second part of Lemma 2.4.1(2). □

In the remainder of this section we show that for the approximation mappings s and nv we don't need weak second-order monadic logic to conclude the decidability of membership to CBN-RS$_{\alpha,\beta}$ for *orthogonal* systems. The reason is that $(\mathcal{R}_\alpha,\mathcal{R}_\beta)$-root-needed redexes for $\alpha,\beta\in\{\mathrm{s},\mathrm{nv}\}$ and orthogonal $\mathcal{R}$ satisfy the partial uniformity result expressed in Proposition 6.2.6 below.

Definition 6.2.5. *Let $\mathcal{R}$ be a system over a signature $\mathcal{F}$. Two redexes $\Delta_1,\Delta_2\in\mathcal{T}(\mathcal{F})$ are called* pattern equal*, denoted by $\Delta_1\approx\Delta_2$, if they have the same redex pattern, i.e., they are redexes with respect to the same rewrite rule. The relation* $\mathsf{mark}^\circ_{\mathcal{R}}$ *on $\mathcal{T}(\mathcal{F}_\circ)$ is defined as the parallel closure of*

$\{(\Delta_1,\Delta_2^\circ)\mid\Delta_1,\Delta_2\in\mathcal{T}(\mathcal{F})$ *are pattern equal redexes*$\}$.

Proposition 6.2.6. *Let $\mathcal{R}$ be an orthogonal system. If redex Δ_1 in $C[\Delta_1]$ is $(\mathcal{R}_\alpha, \mathcal{R}_\beta)$-root-needed with $\alpha, \beta \in \{\mathrm{s}, \mathrm{nv}\}$ then so is redex Δ_2 in $C[\Delta_2]$, for any $\Delta_2 \approx \Delta_1$.*

Orthogonality is essential in Proposition 6.2.6. Consider for instance the pattern equal redexes $\mathsf{f}(\mathsf{g}(\mathsf{a}))$ and $\mathsf{f}(\mathsf{g}(\mathsf{b}))$ with respect to the left-linear system $\mathcal{R}$:

$$\begin{array}{rcl} \mathsf{f}(\mathsf{g}(x)) & \to & \mathsf{f}(\mathsf{a}) \\ \mathsf{g}(\mathsf{b}) & \to & \mathsf{a} \end{array}$$

Redex $\mathsf{f}(\mathsf{g}(\mathsf{a}))$ is $(\mathcal{R}_\alpha, \mathcal{R}_\beta)$-root-needed for all $\alpha, \beta \in \{\mathrm{s}, \mathrm{nv}\}$ because $\mathsf{f}_\circ(\mathsf{g}(\mathsf{a}))$ is an $\mathcal{R}_\alpha$-normal form that is not $(\mathcal{R}_\beta)_\circ$-root-stable. However, redex $\mathsf{f}(\mathsf{g}(\mathsf{b}))$ is not $(\mathcal{R}_\alpha, \mathcal{R}_\beta)$-root-needed since $\mathsf{f}_\circ(\mathsf{g}(\mathsf{b})) \to_{\mathcal{R}_\alpha} \mathsf{f}_\circ(\mathsf{a})$ and $\mathsf{f}_\circ(\mathsf{a})$ is $(\mathcal{R}_\beta)_\circ$-root-stable.

Lemma 6.2.7. *Let $\mathcal{R}$ be a left-linear system. The relation $\mathsf{mark}^\circ_\mathcal{R}$ is gtt-recognizable.*

Proof. Let $\mathcal{F}$ be the signature of $\mathcal{R} = \{l_i \to r_i \mid 1 \leqslant i \leqslant n\}$. Define the GTT $\mathcal{G}$ as $(\mathcal{A}, \mathcal{B})$ where $\mathcal{A}$ is a term automaton without ϵ-rules that accepts in state $[i]$ all instances in $\mathcal{T}(\mathcal{F})$ of l_i and $\mathcal{B}$ is obtained from $\mathcal{A}$ by changing every transition rule of the form $f(q_1, \ldots, q_n) \to [i]$ to $f_\circ(q_1, \ldots, q_n) \to [i]$, followed by a renaming of all states different from $[1], \ldots, [n]$. It is not difficult to see that $\mathcal{G}$ accepts $\mathsf{mark}^\circ_\mathcal{R}$. □

a	$\to$	$[x]$	a	$\to$	$\underline{[x]}$
b	$\to$	$[x]$	b	$\to$	$\underline{[x]}$
$\mathsf{f}([x],[x])$	$\to$	$[x]$	$\mathsf{f}(\underline{[x]},[x])$	$\to$	$\underline{[x]}$
$\mathsf{g}([x])$	$\to$	$[x]$	$\mathsf{g}(\underline{[x]})$	$\to$	$\underline{[x]}$
$\mathsf{h}[x])$	$\to$	$[x]$	$\mathsf{h}\underline{[x]})$	$\to$	$\underline{[x]}$
a	$\to$	$[\mathsf{a}]$	a	$\to$	$\underline{[\mathsf{a}]}$
b	$\to$	$[\mathsf{b}]$	b	$\to$	$\underline{[\mathsf{b}]}$
$\mathsf{g}([x])$	$\to$	$[\mathsf{g}(x)]$	$\mathsf{g}(\underline{[x]})$	$\to$	$\underline{[\mathsf{g}(x)]}$
$\mathsf{f}([x],[\mathsf{b}])$	$\to$	$[\mathsf{f}(x,\mathsf{b})]$	$\mathsf{f}(\underline{[x]},\underline{[\mathsf{b}]})$	$\to$	$\underline{[\mathsf{f}(x,\mathsf{b})]}$
$\mathsf{f}([\mathsf{g}(x)],[\mathsf{a}])$	$\to$	$[1]$	$\mathsf{f}_\circ(\underline{[\mathsf{g}(x)]},\underline{[\mathsf{a}]})$	$\to$	$[1]$
$\mathsf{h}[\mathsf{a}])$	$\to$	$[2]$	$\mathsf{h}_\circ(\underline{[\mathsf{a}]})$	$\to$	$[2]$
$\mathsf{h}[\mathsf{f}(x,\mathsf{b})])$	$\to$	$[3]$	$\mathsf{h}_\circ(\underline{[\mathsf{f}(x,\mathsf{b})]})$	$\to$	$[3]$

Table 6.2: A GTT that accepts $\mathsf{mark}^\circ_\mathcal{R}$.

For the example system $\mathcal{R}$ on page 33 a GTT that accepts $\mathsf{mark}^\circ_\mathcal{R}$ is presented in Table 6.2. Let $\mathcal{R}$ be an orthogonal system over a signature $\mathcal{F}$. Using the preceding results we give a proof of the recognizability of $\{t \in \mathcal{T}(\mathcal{F}) \mid t$ has an $(\mathcal{R}_\alpha, \mathcal{R}_\beta)$-root-needed redex$\}$ for $\alpha, \beta \in \{\mathrm{s}, \mathrm{nv}\}$ which does not rely on

weak second-order monadic logic. Because of Proposition 6.2.6 the above set coincides with $\mathsf{mark}^{\circ}_{\mathcal{R}}[(\mathcal{R}_\alpha, \mathcal{R}_\beta)\text{-}\mathsf{ROOT\text{-}NEEDED}] \cap \mathcal{T}(\mathcal{F})$ and this latter set is recognizable due to Lemmata 6.2.7, 2.4.3(3), and 2.4.1.

Chapter 7

Complexity of CBN-RS classes

The aim of this chapter is to analyze the complexity of deciding membership to $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$ classes. The results of this chapter are new and unpublished. For the linear-growing approximation, we obtain a double exponential upperbound, which is a significant improvement over the non-elementary upperbound of the complexity of the decision procedure presented in Chapter 6 and [DM97].

In this section we assume that $\mathcal{R}$ and $\mathcal{S}$ are orthogonal growing esystems over the same signature $\mathcal{F}$ and that $\mathcal{G} = \mathcal{F} \cup \{f^\circ \mid f \in \mathcal{F}\}$.

7.1 Construction of $\mathcal{A}_{\mathsf{RS}_{\mathcal{S}_\circ}}$

Our first goal is to construct a term automaton recognizing the set $\mathsf{RS}_{\mathcal{S}_\circ}$ of root-stable ground terms with respect to the system $\mathcal{S}_\circ = \mathcal{S} \cup \{l^\circ \to r \mid l \to r \in \mathcal{S}\}$.

Consider the automaton $\mathcal{B}(\mathcal{S}_\circ)$ as defined in Section 5.1.1. To this automaton we add a single final state q_f and transition rules $f([l_1], \ldots, [l_n]) \to q_f$ and $f^\circ([l_1], \ldots, [l_n]) \to q_f$ for every left-hand side $f(l_1, \ldots, l_n)$ of a rewrite rule in $\mathcal{S}$. One easily verifies that the resulting automaton, which we denote by $\mathcal{A}_{\mathsf{REDEX}_{\mathcal{S}_\circ}}$, accepts the set of ground redexes of $\mathcal{S}_\circ$. Applying the construction in Section 5.2.2 to $\mathcal{A}_{\mathsf{REDEX}_{\mathcal{S}_\circ}}$ and $\mathcal{B}(\mathcal{S}_\circ)$ results in deterministic automaton $\mathcal{D}_{\mathsf{REDEX}_{\mathcal{S}_\circ}}(\mathcal{S}_\circ)$ that accepts all ground terms in $\mathcal{T}(\mathcal{G})$ that rewrite in $\mathcal{S}_\circ$ to a term in $\mathsf{REDEX}_{\mathcal{S}_\circ}$, in other words, all non-$\mathcal{S}_\circ$-root-stable terms. This deterministic automaton may be completed, then complemented easily (by changing the set of final states into its complement set in the set of states) to obtain the desired term automaton $\mathcal{A}_{\mathsf{RS}_{\mathcal{S}_\circ}}$. Note that the final states are those that do not contain q_f (the unique final state of $\mathcal{B}(\mathcal{S}_\circ)$.

The previous construction works for the most general (left-linear) growing case. For the linear-growing case, we could have obtained the desired automaton by a subset construction of the non-deterministic automaton $\mathcal{C}_{\mathsf{REDEX}_{\mathcal{S}_\circ}}(\mathcal{S}_\circ)$

(Section 5.2.1), but the size of the final automaton would have been similar to the one obtained from the $\mathcal{D}_{\mathsf{REDEX}_{\mathcal{S}_\circ}}(\mathcal{S}_\circ)$ automaton.

7.2 Automaton $(\to^*_{\mathcal{R}})[\mathsf{RS}_{\mathcal{S}_\circ}]$

Next we apply the constructions in Section 5.2 with $T = \mathsf{RS}_{\mathcal{S}_\circ}$ and $\mathcal{R}$.

7.2.1 Linear-growing case

The construction of Section 5.2.1 yields a non-deterministic automaton $\mathcal{C}_{\mathsf{RS}_{\mathcal{S}_\circ}}(\mathcal{R})$ that accepts all ground terms that rewrite in $\mathcal{R}$ to a term in $\mathsf{RS}_{\mathcal{S}_\circ}$.

7.2.2 (Left-linear) Growing case

The construction of Section 5.2.2 yields a deterministic automaton $\mathcal{D}_{\mathsf{RS}_{\mathcal{S}_\circ}}(\mathcal{R})$ that accepts all ground terms that rewrite in $\mathcal{R}$ to a term in $\mathsf{RS}_{\mathcal{S}_\circ}$.

7.3 Call-by-need computation to root-stable forms

Based on an automaton recognizing $(\to^*_{\mathcal{R}})[\mathsf{RS}_{\mathcal{S}_\circ}]$ (either the non-deterministic $\mathcal{C}_{\mathsf{RS}_{\mathcal{S}_\circ}}(\mathcal{R})$ or the deterministic automaton $\mathcal{D}_{\mathsf{RS}_{\mathcal{S}_\circ}}(\mathcal{R})$), we construct an automaton $\mathcal{E}(\mathcal{R},\mathcal{S})$ that accepts all non-$\mathcal{S}$-root-stable terms in $\mathcal{T}(\mathcal{F})$ that do not have an $(\mathcal{R},\mathcal{S})$-root-needed redex.

7.3.1 Linear-growing case

The construction of $\mathcal{E}(\mathcal{R},\mathcal{S})$ is related to the construction of $\mathcal{E}(\mathcal{R})$ in Section 5.3.1 but there are two main differences.

First the states have now an additional component H to record the behaviour of the automaton $\mathcal{H} = \mathcal{C}_{\mathsf{REDEX}(\mathcal{S})}(\mathcal{S})$ in order to detect whether the recognized term is $\mathcal{S}$-root-stable. The final states are such that $H \cap Q^f_{\mathcal{H}} \neq \emptyset$ (which implies that $[r] \in B$) and $P \subseteq Q^f$.

Second, instead of $P'' = \bullet{\downarrow_{\mathcal{C}}} = \{[x]\}$ (in the case where $[l] \in B$), we define P'' as a non-empty subset of $f^\circ(S_1,\ldots,S_n){\downarrow_{\mathcal{C}}}$; this corresponds to the marking of the root symbol of a redex by $\circ$.

The proofs of the following statements are extensions of the corresponding ones in Section 5.3.

Lemma 7.3.1. *Let $s \in \mathcal{T}(\mathcal{F})$. If $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : H : P\rangle$ then $S = s{\downarrow_{\mathcal{C}}}$, $B = s{\downarrow_{\mathcal{B}}}$ and $H = s{\downarrow_{\mathcal{H}}}$.*

Proof. Straightforward. □

Lemma 7.3.2. *Let $s \in \mathcal{T}(\mathcal{F})$ such that $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : H : P\rangle$. Let $p \in \mathsf{REDEX}(s)$. We have $s[(s/p)^\circ]_p{\downarrow_{\mathcal{C}}} \cap P \neq \varnothing$.*

Proof. Similar to the proof of Lemma 5.3.2 by induction on the depth of p. □

Lemma 7.3.3. *Let $s \in \mathcal{T}(\mathcal{F})$. If $P \subseteq \bigcup_{p\in\mathsf{REDEX}(s)} s[(s/p)^\circ]_p{\downarrow_\mathcal{C}}$ and $\forall p \in \mathsf{REDEX}(s)$, $P \cap s[(s/p)^\circ]_p{\downarrow_\mathcal{C}} \neq \varnothing$ then $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : H : P\rangle$ with $S = s{\downarrow_\mathcal{C}}$, $B = s{\downarrow_\mathcal{B}}$ and $H = s{\downarrow_\mathcal{H}}$.*

Proof. Similar to the proof of Lemma 5.3.3 by induction on the structure of s and taking $P' = P \cap s^\circ{\downarrow_\mathcal{C}}$ in the case where s is a redex. □

Corollary 7.3.4. *Let $s \in \mathcal{T}(\mathcal{F})$. $\forall p \in \mathsf{REDEX}(s)$, $s[(s/p)^\circ]_p \in (\to^+_\mathcal{R})[\mathsf{RS}_{\mathcal{S}_\circ}]$ iff $s \to^+_{\Gamma_\mathcal{E}} \langle S : B : H : P\rangle$ for some $P \subseteq \bigcup_{p\in\mathsf{REDEX}(s)} s[(s/p)^\circ]_p{\downarrow_\mathcal{C}}$ such that $P \cap s[(s/p)^\circ]_p{\downarrow_\mathcal{C}} \cap Q^f \neq \varnothing$.*

Proposition 7.3.5. *Let $s \in \mathcal{T}(\mathcal{F})$. The term $s \in L(\mathcal{E}(\mathcal{R},\mathcal{S}))$ iff s is non $\mathcal{S}$-root-stable and $\forall p \in \mathsf{REDEX}(s)$, $s[(s/p)^\circ]_p \in (\to^+_\mathcal{R})[\mathsf{RS}_{\mathcal{S}_\circ}]$.*

Proof. By Lemma 7.3.1, Corollary 7.3.4 and the observation that s is non $\mathcal{S}$-root-stable if and only if $s{\downarrow_\mathcal{H}} \cap Q^f_\mathcal{H} \neq \varnothing$. □

Theorem 7.3.6. *Let $\mathcal{R}$ be a left-linear system and $\alpha, \beta \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}\}$. We have $\mathcal{R} \in \mathsf{CBN\text{-}RS}_{\alpha,\beta}$ iff $L(\mathcal{E}(\mathcal{R}_\alpha, \mathcal{R}_\beta)) = \varnothing$.*

Proof. By definition of $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$, $\mathcal{R} \notin \mathsf{CBN\text{-}RS}_{\alpha,\beta}$ if and only if there exists a non $\mathcal{R}_\beta$-root-stable term in $\mathcal{T}(\mathcal{F})$ without $(\mathcal{R}_\alpha, \mathcal{R}_\beta)$-root-needed redex. The latter is equivalent to $\forall p \in \mathsf{REDEX}(s)$, $s[(s/p)^\circ]_p \in (\to^+_\mathcal{R})[\mathsf{RS}_{(\mathcal{R}_\beta)_\circ}]$. According to Corollary 7.3.4, this is equivalent to $s \in L(\mathcal{E}(\mathcal{R}_\alpha, \mathcal{R}_\beta))$. Hence $\mathcal{R} \in \mathsf{CBN\text{-}RS}_{\alpha,\beta}$ iff $L(\mathcal{E}(\mathcal{R}_\alpha, \mathcal{R}_\beta)) = \varnothing$. □

Decidability of membership to $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$ follows.

7.3.2 (left-linear) Growing case

With the same modifications as in Section 7.3.1 but applied to the automaton of Section 5.3.2, we obtain from the deterministic $\mathcal{D}_{\mathsf{RS}_{\mathcal{S}_\circ}}(\mathcal{R})$, a deterministic automaton $\mathcal{E}(\mathcal{R},\mathcal{S})$ which recognizes all non-$\mathcal{S}$-root-stable terms of $\mathcal{T}(\mathcal{F})$ with no $(\mathcal{R},\mathcal{S})$-root-needed redex. Decidability of membership to the class $\mathsf{CBN\text{-}RS}_\mathsf{g}$ follows.

Theorem 7.3.7. *Let $\mathcal{R}$ be a left-linear system and $\alpha, \beta \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$. We have $\mathcal{R} \in \mathsf{CBN\text{-}RS}_{\alpha,\beta}$ iff $L(\mathcal{E}(\mathcal{R}_\alpha, \mathcal{R}_\beta)) = \varnothing$.*

7.4 Complexity

The following complexity results are obtained by a similar analysis to the one in Section 5.4. Note that the nested saturation process does not give rise to an extra exponential since saturation increases only the time but not the space complexity by an exponential. Also the additional component $\mathcal{H}$ having a lower complexity than the S component is not a problem.

7.4.1 Linear-growing case

Theorem 7.4.1. *It can be decided in double exponential time whether a left-linear system belongs to* $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$ *for all* $\alpha, \beta \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}\}$.

7.4.2 (left-linear) Growing case

Theorem 7.4.2. *It can be decided in triple exponential time whether a left-linear system belongs to* $\mathsf{CBN\text{-}RS}_{\alpha,\beta}$ *for all* $\alpha, \beta \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$.

Part II

Strong sequentiality

This part recalls the framework for sequentiality and strong sequentiality given in [HL91] (see Chapter 8). This remainder is necessary for Part III which deals exclusively with subclasses of the strongly sequential class. In Chapter 9 we compare our call-by-need approach with the sequentiality-based approach.

Chapter 8

Strong sequentiality

8.1 Strongly Sequential Systems

In this section we recall the definition of strong sequentiality given by Huet and Lévy in their landmark paper of 1979 now published in [HL91].

The notion of sequentiality for a monotonic predicate was first introduced by [KP78].

Definition 8.1.1. *Let P be a monotonic predicate on $\mathcal{T}(\mathcal{F}_\Omega)$. Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$ and $u \in \mathcal{P}\mathsf{os}_\Omega(M)$. u is an* index *of M with respect to P if $N/u \neq \Omega$ for all terms $N \succeq M$ such that $P(N)$. The predicate P is called* sequential *if every Ω-normal form has an index.*

Definition 8.1.2. *Let $(\mathcal{R}, \mathcal{F})$ be a system. The predicate* nf *is defined on $\mathcal{T}(\mathcal{F}_\Omega, \mathcal{V})$ as follows:* $\mathsf{nf}(M)$ *if $M \to^*_{\mathcal{R}} N$ for some normal form $N \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. We say that $\mathcal{R}$ is* sequential *if* nf *is a sequential predicate.*

It is easily checked that nf is a monotonic predicate. The explanation for not restricting the above definitions to ground terms will be given after Example 9.2.2.

Huet and Lévy remarked that sequentiality is undecidable and that indexes (plural of index) with respect to nf are not computable in general. They identified a decidable subclass, the class of *strongly sequential* systems, in which every Ω-normal form admits at least one computable index. This subclass, as well as several later extensions, is defined below using the concept of approximation mapping defined in Section 3.2.

Definition 8.1.3. *Let $(\mathcal{R}, \mathcal{F})$ be a system and α be an approximation mapping. The predicate nf_α is defined on $\mathcal{T}(\mathcal{F}_\Omega, \mathcal{V})$ as follows: $\mathsf{nf}_\alpha(M)$ if and only if $M \to^*_\alpha N$ for some normal form $N \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. We say that $\mathcal{R}$ is* α-sequential *if nf_α is a sequential predicate.*

The class of s-sequential systems coincides with the class of strongly sequential systems (SS) of Huet and Lévy (s being the arbitrary approximation mapping as defined in Section 3.3). The class of nv-sequential systems coincides with

the class of *NVNF-sequential* systems of Nagaya *et al.* [NST95], which is an extension of the class of *NV-sequential* systems of Oyamaguchi [Oya93]. The latter class is defined using the nv approximation mapping but with a different predicate $\mathrm{term}_{\mathsf{nv}}$: $\mathrm{term}_{\mathsf{nv}}(M)$ if and only if $M \to^*_{\mathsf{nv}} N$ for some *term* $N \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. The class of lg-sequential systems coincides with the class of linear-growing sequential systems of Jacquemard [Jac96b] and the class of g-sequential with the class of growing sequential systems of Nagaya and Toyama [NT02].

In Chapter 9, we will compare the classes defined in Definition 8.1.3 with our CBN_α classes.

We now give more definitions and properties (most of them due to Huet and Lévy) that will be useful in Part III which deals with subclasses of the s-sequential class (SS).

The set of indexes of an Ω-term M is denoted by $\mathcal{I}(M)$. Intuitively, a position $u \in \mathcal{I}(M)$ cannot disappear via s-reduction if the Ω-term M is not refined at u. From now on, an index with respect to $\mathsf{nf_s}$ will be simply called an *index*. It is easy to decide whether an Ω-position of an Ω-term is an index [HL91]. However deciding whether a system is strongly sequential is not a trivial matter; the first proof was given by [HL91]; other proofs can be found in [KM91], [Com00]. The first proof avoiding the concept of index was given in [DM97] and is the one given in Chapter 3. In [Com00], Comon showed that the problem is in EXPTIME. In [KM91], Klop and Middeldorp conjecture that deciding strong sequentiality is NP-Complete. This is still an open problem (listed as number 9, in the RTA list of (open) problems in rewriting, see Chapter 13).

Definition 8.1.4. *The reduction relation* $\longrightarrow_\Omega$ *(*Ω-reduction*) is defined as follows: Let* $M \in \mathcal{T}(\mathcal{F}_\Omega)$ *and* $u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M)$*,* $M \longrightarrow_\Omega M[\Omega]_u$ *if* M/u *is redex compatible.*

All examples in this section will be related to the system $\mathcal{R}_1$ of Example A.0.1.

Example. $\mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega))) \to_\Omega \mathsf{f}(\mathsf{a}, \mathsf{f}(\Omega, \mathsf{g}(\mathsf{b}, \Omega))) \to_\Omega \mathsf{f}(\mathsf{a}, \mathsf{f}(\Omega, \Omega)) \to_\Omega \mathsf{f}(\mathsf{a}, \Omega)$.

Proposition 8.1.5. *[KM91]* Ω*-reduction is confluent and terminating.*

The next proposition relates Ω-reduction to arbitrary reduction.

Proposition 8.1.6. *[KM91] Let* $M, N \in \mathcal{T}(\mathcal{F}_\Omega)$
If $M \to^*_\Omega N$ *then* $M' \to^*_{\mathsf{s}} N$ *for some* $M' \succeq M$*.*
If $M \to^*_{\mathsf{s}} N$ *then* $M \to^*_\Omega N'$ *for some* $N' \preceq N$*.*

The next two lemmata relate Ω-reduction to arbitrary reduction in the case of reductions to Ω.

Lemma 8.1.7. *Let* $M \in \mathcal{T}(\mathcal{F}_\Omega)$*. If* $M \to^+_{\mathsf{s}} \Omega$ *then* $\forall N \in \mathcal{T}(\mathcal{F}_\Omega)$*,* $M \to^*_\Omega N$*.*

Proof. The sequence of rewrite steps $M \to^+_{\mathsf{s}} \Omega$ is necessarily of the form $M \to^*_{\mathsf{s}} \Delta \to_{\mathsf{s}} \Omega$ for some redex Δ. From the definition of $\to_{\mathsf{s}}$, it follows that $M \to^*_{\mathsf{s}} \Delta \to_{\mathsf{s}} N$, $\forall N \in \mathcal{T}(\mathcal{F}_\Omega)$. □

Lemma 8.1.8. *Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$. If $M \rightarrow_\Omega^+ \Omega$ then there exists $M' \succeq M$ such that $\forall N \in \mathcal{T}(\mathcal{F}_\Omega)$, $M' \rightarrow_{\mathsf{s}}^* N$.*

Proof. By induction on the length of $M \rightarrow_\Omega^+ \Omega$. If the length is 1 then $M \preceq \Delta$ for some redex Δ. We have $\Delta \rightarrow_{\mathsf{s}} N$, $\forall N \in \mathcal{T}(\mathcal{F}_\Omega)$. Otherwise, we may write $M = C[M_1, \ldots, M_n]_{u_1,\ldots,u_n} \rightarrow_\Omega^* C[\Omega, \ldots, \Omega]_{u_1,\ldots,u_n} \rightarrow_\Omega \Omega$. By definition of Ω-reduction $C[\Omega, \ldots, \Omega]_{u_1,\ldots,u_n} \preceq \Delta$ for some redex Δ. We have $\forall i$, $1 \leq i \leq n$, $M_i \rightarrow_\Omega^+ \Omega$. $\forall i, 1 \leq i \leq n$, let $N_i = \Delta/u_i$. The induction hypothesis yields $\forall i$, $1 \leq i \leq n$, $M_i \rightarrow_{\mathsf{s}}^* N_i$. It follows that $\forall N \in \mathcal{T}(\mathcal{F}_\Omega)$, $M = C[M_1, \ldots, M_n]_{u_1,\ldots,u_n} \rightarrow_{\mathsf{s}}^* C[N_1, \ldots, N_n]_{u_1,\ldots,u_n} = \Delta \rightarrow_{\mathsf{s}} N$. for any $N \in \mathcal{T}(\mathcal{F}_\Omega)$. □

Definition 8.1.9. *[HL91] The* direct approximant $\omega(M)$ *of an Ω-term M is the normal form of M with respect to Ω-reduction. $\omega(M)$ is well defined according to Proposition 8.1.5.*

Example. $\omega(\mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega)))) = \mathsf{f}(\mathsf{a}, \Omega)$.

We recall some simple but very useful properties related to Ω-reduction.

Proposition 8.1.10. *[HL91] Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$.*
1) $\omega(M) \preceq M$.
2) Let $u \in \mathcal{P}\mathsf{os}(M)$. $\omega(M) = \omega(M[\omega(M/u)]_u)$.
3) Let $N \in \mathcal{T}(\mathcal{F}_\Omega)$ such that $N \preceq M$. $\omega(N) \preceq \omega(M)$.
4) $\omega(\omega(M)) = \omega(M)$.
5) If M is redex compatible then $\omega(M) = \Omega$.

Lemma 8.1.11. *Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$. $\omega(M)$ can be computed in $O(|\mathsf{LHS}_\Omega| + |t|^3)$*

Proof. Let $M = M_0 \longrightarrow_\Omega M_1 \ldots \longrightarrow_\Omega M_k = \omega(M)$ be any sequence of Ω-reductions from M to $\omega(M)$. $\forall i, 0 < i \leq k$, $M_{i-1} \prec M_i$, so $\forall i, 0 < i \leq k$, $|M_{i-1}| < |M_i|$, so $k \leq |M|$. In $O(|\mathsf{LHS}_\Omega|)$, we can build a pattern matching automaton to search for a redex compatible subterm in a term. At each step i of Ω-reduction, using the automaton, the search of a redex compatible subterm in M_i can be done in $O(|M_i|^2)$. We conclude that the global computation of $\omega(M)$ can be done in $O(|\mathsf{LHS}_\Omega| + |M|^3)$. □

The concept of *soft term* comes from [KM91]. Soft terms are also called *potential* redexes in O'Donnell's terminology [O'D85]. Soft terms "melt" through Ω-reduction.

Definition 8.1.12. *An Ω-term M is called* soft *if $\omega(M) = \Omega$.*

Example. *The Ω-term $\mathsf{f}(\Omega, \mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega)))$ is soft.*
The Ω-term $\mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega)))$ is not soft.

Definition 8.1.13. *An Ω-term M such that $\omega(M) \neq \Omega$ is called* strongly root-stable*. Similarly, an Ω-position u of an Ω-term M is* strongly root-stable *if M/u is strongly root-stable.*

Strongly root-stable Ω-terms are also called *strong head normal forms* in [Str88].

The following lemma and corollary relate strong root-stability and $\mathcal{R}_s$-root-stability.

Lemma 8.1.14. *Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$. M is not strongly root-stable iff there exists $M' \succeq M$ such that M' is not $\mathcal{R}_s$-root-stable.*

Proof. We first get rid of the trivial case where $M = \Omega$ and any redex is greater than or equal to M and not $\mathcal{R}_s$-root-stable. Now we assume that $M \neq \Omega$.

$\Longrightarrow$ By definition of stronlgy root-stability we have $M \to_\Omega^* \Omega$. Proposition 8.1.6 (first item) yields some $M' \succeq M$ such that $M' \to_s^* \Omega$ which gives $M' \to_s^+ \Omega$ as $M \neq \Omega$. Lemma 8.1.8 yields that M' can arbitrarily reduce to any term, in particular to a redex so M' is not $\mathcal{R}_s$-root-stable.

$\Longleftarrow$ By definition of $\mathcal{R}_s$-root-stability, M' arbitrarily reduces to a redex which arbitrarily reduces to Ω. So $M' \to_s^* \Omega$. Proposition 8.1.6 (second item) yields $M' \to_\Omega^* \Omega$so $\omega(M') = \Omega$. As $M \preceq M'$, the third item of Proposition 8.1.10 yields $\omega(M) = \Omega$ which means by definition that M is not strongly root-stable.

□

Corollary 8.1.15. *Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$. M is strongly root-stable iff $\forall M' \succeq M$, M' is $\mathcal{R}_s$-root-stable.*

Definition 8.1.16. *[HL91] Let $M = f(M_1, \ldots, M_n) \in \mathcal{T}(\mathcal{F}_\Omega)$. The* internal direct approximant *of M is defined by $\overline{\omega}(M) = f(\omega(M_1), \ldots, \omega(M_n))$.*

Example. $\overline{\omega}(\mathsf{f}(\Omega, \mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega)))) = \mathsf{f}(\Omega, \Omega)$.

Let $\bullet$ be a fresh constant symbol. The next lemma shows that with Ω-reduction it is simple to decide whether an Ω-position of an Ω-term is an index.

Lemma 8.1.17. *[HL91] Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$ and $u \in \mathcal{P}os_\Omega(M)$. $u \in \mathcal{I}(M)$ iff $\omega(M[\bullet]_u) \neq \omega(M)$.*

Example. $\omega(\mathsf{f}(\mathsf{g}(\bullet, \Omega), \Omega)) = \Omega$ *so* $1.1 \notin \mathcal{I}(\mathsf{f}(\mathsf{g}(\Omega, \Omega), \Omega))$.
$\omega(\mathsf{f}(\mathsf{g}(\Omega, \Omega), \bullet)) = \mathsf{f}(\Omega, \bullet)$ *so* $2 \in \mathcal{I}(\mathsf{f}(\mathsf{g}(\Omega, \Omega), \Omega))$.

The following lemma from [HL91] holds for orthogonal systems.

Lemma 8.1.18. *[HL91] Let $\mathcal{R}$ be an orthogonal system. If $uv \in \mathcal{I}(M)$, then $v \in \mathcal{I}(M/u)$.*

The set $\mathsf{Dir}(M, S)$ of *directions* from M to a set of Ω-terms S is defined in [HL91]. We will use the characterization given by the following lemma as an alternative definition.

Lemma 8.1.19. *[HL91] Let S be a subset of $\mathcal{T}(\mathcal{F}_\Omega)$. Let $M \in \mathcal{T}(\mathcal{F}_\Omega)$ and $u \in \mathcal{P}os_\Omega(M)$. $u \in \mathsf{Dir}(M, S)$ iff $\forall N \in S$ such that $N \uparrow M$, one has $u \in \mathcal{P}os_{\overline{\Omega}}(N)$.*

$\mathsf{Dir}(M, \mathsf{LHS}_\Omega)$ is abbreviated $\mathsf{Dir}(M)$.

Example. $\mathsf{Dir}(\mathsf{f}(\mathsf{g}(\Omega, \mathsf{a}), \Omega)) = \{2\}$.

8.2 matching DAGs

We now give Huet & Lévy's construction of a *matching DAG*. This construction is directly taken from [HL91]. We need the concept of matching DAG in order to compare it with the concept of index tree in Chapter 10.

HYPOTHESIS: Huet and Lévy assume the existence of a function Q mapping every preredex M into a nonempty subset of its directions, $\varnothing \neq Q(M) \subseteq \mathsf{Dir}(M)$ and such that $\forall u \in Q(M)$, for all preredexes N, if $M[N]_u$ is a preredex then

$\exists v, uv \in Q(M[N]_u)$ (*Q1*).

$\forall v, uv \in Q(M[N]_u) \Rightarrow v \in Q(N)$ (*Q2*).

Intuitively, a position in $Q(M)$ is a direction for matching redexes and is also a direction for internal matches (by condition $Q2$). Furthermore, it is possible to find some (strongly needed) redex without any dangling partial matches (condition $Q1$).

CONSTRUCTION: To construct a *matching DAG* associated with Q and satisfying $Q1$ and $Q2$, let $\mathsf{LHS}_\Omega = \{L_1, L_2, \ldots, L_n\}$. Consider $Occ = \bigcup\{\mathcal{P}\mathsf{os}_{\overline{\Omega}}(L_i) | i \leq n\}$ and a set $Succ = \{S_1, S_2, \ldots, S_n\}$ of success tokens disjoint from Occ. The nodes of the graph are all

a) pairs (M, v) such that M is a preredex, $v \in Q(M)$, and $\forall u, M/u \neq \Omega$ and $M/u \uparrow L$ for some $L \in \mathsf{LHS}_\Omega \Rightarrow u < v$,

or

b) pairs (L_i, S_i), with $1 \leq i \leq n$ (the success nodes).

Let (M, v) be a node, f a function symbol in $\mathcal{F}$, and $M' = \mathsf{ext}(M, v, f)$. If $(M', *)$ is a node where $*$ denotes any element of $Occ \cup Succ$, an arc is drawn: $(M, v) \xrightarrow{f} (M', *)$. It is clear that such graphs are directed and acyclic.

A node $(M, *)$ is *accessible* iff either it is the origin node (Ω, ε) or there is some accessible node (M_1, v_1) and $f \in \mathcal{F}$ such that $(M_1, v_1) \xrightarrow{f} (M', *)$.

To end the construction, inaccessible nodes are removed. Then the terminal nodes are exactly all the success nodes and in all nonterminal nodes (M, v), M is a preredex. The *failure function* is defined for every nonterminal (accessible) node (M, v) different from the initial node (Ω, ε) in the following way:

$Fail((M, v)) = (M/u, v/u)$ where u is the smallest non-null prefix of v such that $M/u \uparrow L$ for some $L \in \mathsf{LHS}_\Omega$.

As $Fail((M, v))$ can be proven to be accessible, $Fail$ can also be characterized by:

$Fail((M, v)) = (M/u, v/u)$ where u is the minimun non-null prefix of v such that M/u is a preredex.

With the following theorem, Huet and Lévy prove that Strong Sequentiality is a decidable property of orthogonal systems.

Theorem 8.2.1. (Huet & Lévy, 1979) *Let $\mathcal{R}$ be an orthogonal system. $\mathcal{R}$ is strongly sequential iff there exists a matching DAG for LHS_Ω.*

A matching DAG for $\mathcal{R}_1$ of Example A.0.1 is shown in Figure 8.1. Note that as matching DAGs are essentialy the same as index trees (see Chapter 10), we

adopt the same legend for index trees and matching DAGs (see p 99).

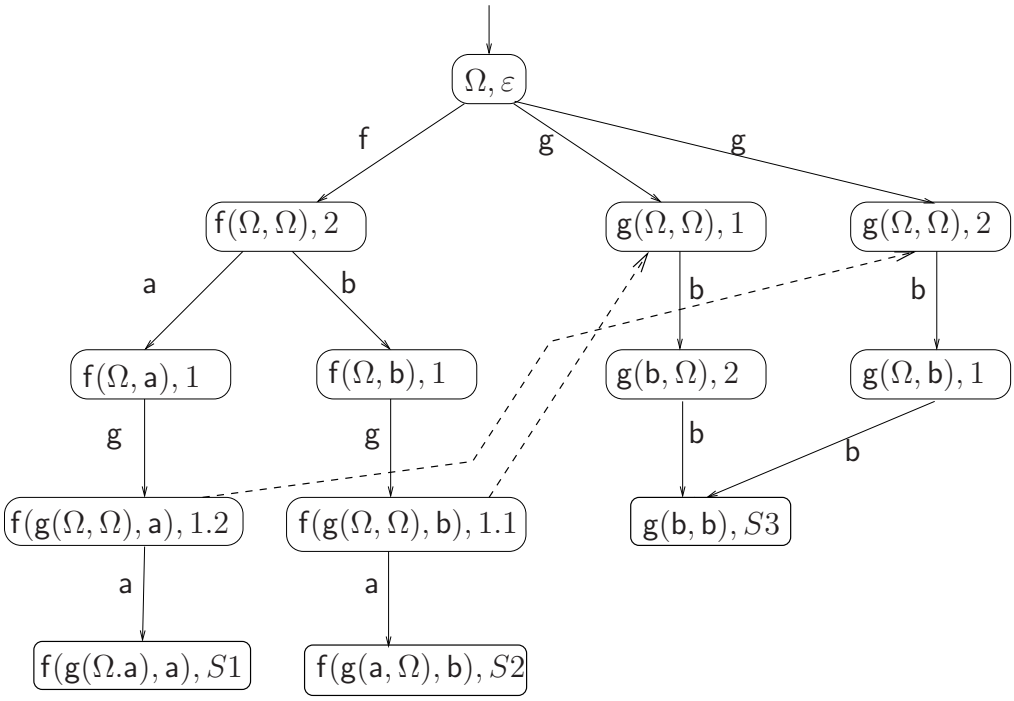


Figure 8.1: A matching DAG for $\mathcal{R}_1$ of Example A.0.1

The next proposition first proved by Huet and Lévy shows that deciding Strong Sequentiality becomes easier with the restriction to constructor systems.

Proposition 8.2.2. [HL91] [KM91] [Klo92] *A constructor system is strongly sequential iff every strict preredex has an index.*

In a matching DAG for a constructor system every failure transition goes to the initial node. Such a matching DAG can be constructed in linear time in a single top-down pass. If the system is not strongly sequential a preredex without index will be found during the construction. Consequently, it can be decided in linear time whether a constructor system is strongly sequential.

8.3 Simple Systems

The notion of *sequential set* of Ω-terms is defined in [HL91]. We will use the characterization given by the following lemma as an alternative definition.

Lemma 8.3.1. *Let S be a set of Ω-terms. S is* sequential *iff $\forall M \in \mathcal{T}(\mathcal{F}_\Omega)$ such that $M \uparrow S$ but $M \not\geq S$, $\mathsf{Dir}(M) \neq \varnothing$.*

Definition 8.3.2. *[HL91] A system is* simple *if every subset of* $\mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^*$ *is a sequential set.*

By SP we denote the set of simple systems. The class of simple systems will be compared with the class of forward-branching systems in Chapter 12.

Chapter 9

CBN versus Sequentiality

In this chapter, we compare the α-sequential classes defined in Definition 8.1.3 with our CBN_α classes of Section 3.4.

9.1 α-sequentiality versus CBN_α

The following lemma connects nf_α-indexes with α-needed redexes.

Lemma 9.1.1. *Let $(\mathcal{R}, \mathcal{F})$ be a left-linear system and let α be an approximation mapping. If a position p in a term $t \in \mathcal{T}(\mathcal{F}_\Omega)$ is an nf_α-index then redex Δ in the term $s[\Delta]_p$ is α-needed, for all terms $s \geqslant t$ and redexes Δ.*

Proof. Suppose Δ is not an α-needed redex in the term $s[\Delta]_p$. Then there exists a normal form $u \in \mathcal{T}(\mathcal{F})$ such that $s[\bullet]_p \to_\alpha^* u$. Since $\mathcal{R}_\alpha$ is left-linear and $\bullet$ does not appear in its rewrite rules, we obtain $s[\Omega]_p \to_\alpha^* u$ from $s[\bullet]_p \to_\alpha^* u$ by replacing all positions of $\bullet$ by Ω. It follows that $\mathsf{nf}_\alpha(s[\Omega]_p)$ holds. We have $s[\Omega]_p \geqslant t$ as $t|_p = \Omega$. Hence p is not an nf_α-index position. □

Corollary 9.1.2. *Let α be an approximation mapping. Every left-linear α-sequential system belongs to CBN_α.*

Proof. Let $\mathcal{R}$ be a left-linear α-sequential system. We show that every reducible term s has an α-needed redex. Let t be the Ω-normal form obtained from s by replacing all outermost redexes by Ω. Because $\mathcal{R}$ is α-sequential, t has an nf_α-index, say at Ω-position p. We obviously have $s \geqslant t$. According to the previous lemma the redex at position p in s is α-needed. We conclude that $\mathcal{R} \in \mathsf{CBN}_\alpha$. □

The reverse directions do not hold in general. For the strong approximation this is kind of surprising since redexes carry the same information as Ω because the former can reduce to any term.

Example 9.1.3. *Consider the system $\mathcal{R} = \mathcal{R}_5$ of Example A.0.5 over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules.*
As $\mathsf{NF}(\mathcal{R}) = \varnothing$, $\mathcal{R}$ trivially belongs to $\mathsf{CBN_s}$. However, $\mathcal{R}$ is not strongly sequential since the Ω-normal form $\mathsf{f}(\Omega,\Omega,\Omega)$ does not have an $\mathsf{nf_s}$-index:

$$\mathsf{f}(\Omega,\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a})) \to_{\mathsf{s}} x \qquad \mathsf{f}(\mathsf{h}(\mathsf{a}),\Omega,\mathsf{g}(\mathsf{a})) \to_{\mathsf{s}} x \qquad \mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a}),\Omega) \to_{\mathsf{s}} x$$

9.2 s-sequentiality (SS) versus $\mathsf{CBN_s}$

The following lemma states that for orthogonal systems the discrepancy between strong sequentiality and $\mathsf{CBN_s}$ can only occur if there are no ground normal forms.

Lemma 9.2.1. *Let $(\mathcal{R},\mathcal{F})$ be an orthogonal system such that $\mathsf{NF}(\mathcal{R},\mathcal{F}) \neq \varnothing$. If $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN_s}$ then $(\mathcal{R},\mathcal{F})$ is strongly sequential.*

Proof. Suppose that $(\mathcal{R},\mathcal{F})$ is not strongly sequential. So there exists an Ω-normal form $t \in \mathcal{T}(\mathcal{F}_\Omega,\mathcal{V})$ without $\mathsf{nf_s}$-indexes. Let $u \in \mathcal{T}(\mathcal{F})$ be the term obtained from t by replacing all positions of Ω by a ground redex. (Since the empty system is trivially strongly sequential, $\mathcal{R}$ contains at least one rule.) We claim that u has no s-needed redexes. Let P be the set of Ω-positions in t, which coincides with the set of redex positions in u because of orthogonality. Let $p \in P$. We show that the redex in u at position p is not s-needed. Since p is not an $\mathsf{nf_s}$-index position in t, we have $\mathsf{nf_s}(s)$ for some term $s \in \mathcal{T}(\mathcal{F}_\Omega,\mathcal{V})$ with $s \geqslant t$ and $s/p = \Omega$. Without loss of generality we assume that p is the only Ω-position in s. There exists a rewrite sequence $A\colon s \to_{\mathsf{s}}^* s'$ with $s' \in \mathcal{T}(\mathcal{F},\mathcal{V})$ a normal form. Since there is no Ω in s', A must contain a rewrite step at a position $q < p$. Let $s_1 \to_{\mathsf{s}} s_2$ be the first such step. By simply replacing every position of Ω by a variable, we may assume that the remainder $s_2 \to_{\mathsf{s}}^* s'$ of A does not contain any positions of Ω. We will now transform A into a sequence $B\colon u[\bullet]_p \to_{\mathsf{s}}^* u'$ with $u' \in \mathsf{NF}(\mathcal{R}_\bullet)$, which implies that redex $u|_p$ is not s-needed. By replacing every variable in A by some constant we obtain the sequence $\hat{A}\colon \hat{s} \to_{\mathsf{s}}^* \hat{s}_1 \to_{\mathsf{s}} \hat{s}_2 \to_{\mathsf{s}}^* \hat{s}'$, where $\hat{s}'$ need not be in normal form. Next we replace all positions of Ω in $\hat{s} \to_{\mathsf{s}}^* \hat{s}_1$ by $\bullet$, yielding $\hat{u} \to_{\mathsf{s}}^* \hat{u}_1$. Because redexes s-rewrite to all possible terms and $\hat{u}/p = \bullet$, we clearly have $u[\bullet]_p \to_{\mathsf{s}}^* \hat{u}$. Note that $\hat{u}_1$ contains a single position of $\bullet$, at position p, and a redex at position q. We obtain $\hat{u}_1 \to_{\mathsf{s}} \hat{s}_2$ by contracting this redex. Combining the various parts yields $u[\bullet]_p \to_{\mathsf{s}}^* \hat{s}'$. If we can s-rewrite $\hat{s}'$ to a ground normal form then we obtain the desired rewrite sequence B. It is easy to see that repeatedly replacing redexes by any ground normal form, whose existence is guaranteed by the assumption $\mathsf{NF}(\mathcal{R}) \neq \varnothing$, will terminate in a ground normal form. □

The following example shows that Lemma 9.2.1 need not be true for left-linear systems.

Example 9.2.2. *Consider the left-linear system* $\mathcal{R}$

$$\begin{array}{lll} \mathsf{g}(\mathsf{f}(x,\mathsf{a})) \to \mathsf{a} & \mathsf{f}(\mathsf{g}(x),\mathsf{g}(y)) \to \mathsf{a} & \mathsf{f}(\mathsf{g}(x),\mathsf{f}(y,z)) \to \mathsf{a} \\ \mathsf{g}(\mathsf{f}(\mathsf{a},x)) \to \mathsf{a} & \mathsf{f}(\mathsf{f}(x,y),\mathsf{f}(z,u)) \to \mathsf{a} & \mathsf{f}(\mathsf{f}(x,y),\mathsf{g}(z)) \to \mathsf{a} \end{array}$$

The Ω*-normal form* $\mathsf{g}(\mathsf{f}(\Omega,\Omega))$ *has no* $\mathsf{nf_s}$*-indexes:*

$$\mathsf{g}(\mathsf{f}(\Omega,\mathsf{a})) \to_{\mathsf{s}} \mathsf{a} \qquad\qquad \mathsf{g}(\mathsf{f}(\mathsf{a},\Omega)) \to_{\mathsf{s}} \mathsf{a}$$

and hence $\mathcal{R}$ *is not strongly sequential. Membership in* $\mathsf{CBN_s}$ *is not hard to prove.*[1]

The reader may wonder why definitions 8.1.2 and 8.1.3 in section 8.1 are not restricted to ground terms. The reason is that the standard decision procedure for $\mathsf{nf_s}$-indexes requires the existence of variables. To see this, let us recall the details of this procedure [HL91, KM91].

A term $t \in \mathcal{T}(\mathcal{F}_\Omega,\mathcal{V})$ is redex-compatible if $t \leqslant u$ for some redex u. The relation $\to_\Omega$ is defined as follows: $C[t] \to_\Omega C[\Omega]$ for every context C and redex-compatible term $t \neq \Omega$. The relation $\to_\Omega$ is confluent and terminating, and hence every term t admits a unique normal form with respect to $\to_\Omega$, which is denoted by $\omega(t)$. Now, an Ω-position p in t is an $\mathsf{nf_s}$-index if and only if $p \in \mathcal{P}\mathsf{os}(\omega(t[\bullet]_p))$. The proof of this equivalence (see [KM91, Lemma 4.8]) relies on the existence of variables.

Returning to Example 9.1.3, we have

$$\omega(\mathsf{f}(\bullet,\Omega,\Omega)) = \omega(\mathsf{f}(\Omega,\bullet,\Omega)) = \omega(\mathsf{f}(\Omega,\Omega,\bullet)) = \Omega,$$

confirming that the term $\mathsf{f}(\Omega,\Omega,\Omega)$ indeed lacks $\mathsf{nf_s}$-indexes. If we would restrict the above sequentiality definitions to ground terms, then all Ω-positions would become $\mathsf{nf_s}$-indexes; because of the rewrite rule $\mathsf{a} \to \mathsf{a}$ there are no ground normal forms without Ω and hence $\mathsf{nf_s}(t)$ fails as soon as t contains a position of Ω.[2]

After this digression we return to the comparison between CBN_α and α-sequentiality.

9.3 α-sequentiality versus CBN_α (continued)

It is easy to show that $\mathsf{CBN_{nv}}$ properly includes the class of nv-sequential systems (and hence also the class of NV-sequential systems introduced by Oyamaguchi [Oya93]).

Example 9.3.1. *Consider the system* $\mathcal{R}_a$ *defined in the proof of Lemma 3.4.6 (p36). The following rewrite steps show that the* Ω*-normal form* $\mathsf{f}(\Omega,\Omega,\Omega)$ *does not have an index with respect to* $\mathsf{nf_{nv}}$*:*

$$\mathsf{f}(\Omega,\mathsf{a},\mathsf{b}) \to_{\mathsf{nv}} \mathsf{c} \qquad \mathsf{f}(\mathsf{b},\Omega,\mathsf{a}) \to_{\mathsf{nv}} \mathsf{b} \qquad \mathsf{f}(\mathsf{a},\mathsf{b},\Omega) \to_{\mathsf{nv}} \mathsf{a}$$

[1] Membership can also be verified by the `Autowrite` tool; see Chapter 15.

[2] It follows that the suggestion made in the footnote [2] in [Com00] to simulate variables by enriching the signature is mandatory rather than optional.

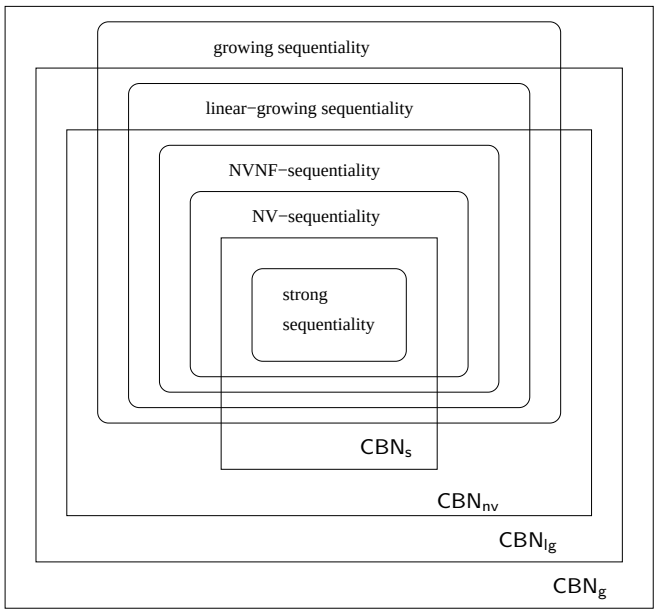


Figure 9.1: Comparison.

Since $\mathcal{R}_a \in \mathsf{CBN}_{\mathsf{nv}}$*, it follows that the class of* nv*-sequential systems is a proper subclass of* $\mathsf{CBN}_{\mathsf{nv}}$*.*

It is interesting to note that the same example illustrates that Huet and Lévy's sequentiality concept does *not* capture the class of (orthogonal) systems that admit a (computable or otherwise) call-by-need strategy. Since $\mathcal{R}_a$ is right-ground, we have $\to_{\mathsf{nv}} = \to_{\mathcal{R}_a}$ and thus $\mathsf{nf}_{\mathsf{nv}} = \mathsf{nf}$. Hence $\mathcal{R}_a$ is not sequential. Because $\mathcal{R}_a$ is orthogonal and belongs to $\mathsf{CBN}_{\mathsf{nv}}$, it obviously admits a computable call-by-need strategy.

Figure 9.3 summarizes the findings of this section. Concerning the placement of $\mathsf{CBN}_{\mathsf{s}}$, the system $\mathcal{R}$ in Example 9.1.3 is not nv-sequential. To show that $\mathsf{CBN}_{\mathsf{s}}$ contains systems that are not g-sequential, we need to slightly modify the example.

Example 9.3.2. *Consider the system* $\mathcal{R}$

$$\begin{aligned} \mathsf{f}(x,\mathsf{g}(y),\mathsf{h}(z)) &\to \mathsf{i}(\mathsf{g}(x)) & \mathsf{i}(\mathsf{g}(x)) &\to x\\ \mathsf{f}(\mathsf{h}(z),x,\mathsf{g}(y)) &\to \mathsf{i}(\mathsf{g}(x)) & \mathsf{a} &\to \mathsf{a}\\ \mathsf{f}(\mathsf{g}(y),\mathsf{h}(z),x) &\to \mathsf{i}(\mathsf{g}(x)) \end{aligned}$$

over the signature $\mathcal{F}$ *consisting of all symbols appearing in the rewrite rules. We have* $\mathcal{R} \in \mathsf{CBN}_{\mathsf{s}}$ *because* $\mathsf{NF}(\mathcal{R}) = \varnothing$*. The system* $\mathcal{R}$ *is not* g*-sequential since*

the Ω*-normal form* $\mathsf{f}(\Omega,\Omega,\Omega)$ *does not have an* nf_g*-index:*

$$\mathsf{f}(\Omega,\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a})) \to_\mathsf{g} \mathsf{i}(\mathsf{g}(\Omega)) \to_\mathsf{g} x$$
$$\mathsf{f}(\mathsf{h}(\mathsf{a}),\Omega,\mathsf{g}(\mathsf{a})) \to_\mathsf{g} \mathsf{i}(\mathsf{g}(\Omega)) \to_\mathsf{g} x$$
$$\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{h}(\mathsf{a}),\Omega) \to_\mathsf{g} \mathsf{i}(\mathsf{g}(\Omega)) \to_\mathsf{g} x$$

Part III

Below strong sequentiality

In this part we will discuss only subclasses of the strongly sequential class (SS). The reason for studying such classes was that they seem to be the only hope for perfoming efficiently sequences of rewriting steps (see our discussion in Chapter 14).

From [HL91], we know that deciding strong sequentiality for constructor systems is linear. A natural idea was to look for classes of strongly sequential systems that could be transformed into constructor strongly sequential systems. We will see in Section 12.1, that the forward-branching class is the greatest known such class. But this result came years after the definition of the forward-branching was given. In the meantime, we had defined the class of constructor equivalent systems discussed in Chapter 11 and many interesting results concerning the forward-branching class were obtained.

In this part we deal with orthogonal systems only.

Chapter 10

Forward-branching systems

Forward-branching systems where first defined by Strandh in [Str88, Str89]. A system is *forward-branching* if it admits a *forward-branching index tree*. This is why we recall below some of Strandh's terminology and his definition of *index tree*. We show that although defined in a completely different way, an index tree and a matching DAG (see Section 8.2) are quite similar.

The forward-branching class is a proper extension of the class of simple systems defined by [HL91]. We give a simple algebraic characterization of forward-branching systems from which it easy to deduce that the forward-branching class is the same as the class of *transitive* systems defined by [TSvEP93b].

Most of the work presented in this chapter has been published in [Dur94a] but Section 10.4 has been improved before being included in the present document.

10.1 Definition of an index tree

We now give some of Strandh's terminology and the definition of index tree. Strandh's terminology is mostly inspired by O'Donnell's [O'D85]. In all this section definition and lemmata are related to some system $\mathcal{R}$. The examples used to illustrate the definitions presented in this section are based on the system $\mathcal{R} = \mathcal{R}_1$ of example A.0.1.

Definition 10.1.1. *An Ω-term M is* firm *if $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$ such that $\forall v \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M)$, either v is strongly root-stable or $v < u$. We call such a position u a* firm extension position *of M. By $\mathsf{ep}(M)$ we denote the set of firm extension positions of a firm Ω-term M.*

Example. $\mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{a}, \Omega))$ *is a firm Ω-term for $\mathcal{R}_1$:*
1.1 *(resp.* 1.2*) is a firm extension position.*
2.2 *is not a firm extension position because* 1 *is not strongly root-stable.*

Example. $\mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{g}(\mathsf{b}, \Omega))$ *is not a firm Ω-term for $\mathcal{R}_1$: none of the positions* 1.1*,* 1.2 *or* 2.2 *is a firm extension position.*

Definition 10.1.2. *An* index point *is a pair* $[M, u]$ *where* M *is a firm preredex and* $u \in \mathsf{ep}(M) \cap \mathcal{I}(M)$.

The index point $[\Omega, \varepsilon]$ is called the *initial index point*.

Example. $[\mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{a}), 1.2]$ *is an index point for* $\mathcal{R}_1$*:* 1.2 *is both a firm extension position and an index.*
No index point can be obtained with the firm preredex $\mathsf{f}(\mathsf{g}(\Omega, \Omega), \Omega)$ *because* 2 *is not a firm extension position and* 1.1 *and* 1.2 *are not indexes.*

Definition 10.1.3. *Given a non initial index point* $s = [M, w]$*, an index point* $t = [N, v]$ *is a* failure point *of* $[M, w]$ *if* $\exists u \in \mathcal{P}\mathsf{os}^+(M)$ *such that* $w = uv$ *and* $N = M/u$*. A failure point* t *of* s *is the* immediate failure point *of* s *iff every failure point of* s *other than* t *is a failure point of* t.

The following lemma shows that the failure points of a non initial index point $[M, w]$ are exactly the pairs $[M/u, w/u]$ where $\varepsilon < u < w$ and M/u is a preredex (there is no need to check whether they are index points).

Lemma 10.1.4. *Let* $[M, w]$ *be a non initial index point. If* $\exists u$*,* $\varepsilon < u \leq w$*, such that* M/u *is a preredex then* $[M/u, w/u]$ *is a failure point of* $[M, w]$.

Proof. As $w \in \mathcal{I}(M)$ and from Lemma 8.1.18, we get $w/u \in \mathcal{I}(M/u)$. As w is a firm extension position of M it is clear that w/u is a firm extension position of M/u. So, $[M/u, w/u]$ is an index point. □

Example. $[\mathsf{g}(\Omega, \Omega), 2]$ *is the immediate failure point of* $[\mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{a}), 1.2]$.
$[\Omega, \varepsilon]$ *is a failure point of* $[\mathsf{f}(\mathsf{g}(\Omega, \Omega), \mathsf{a}), 1.2]$.
$[\Omega, \varepsilon]$ *is the immediate failure point of* $[\mathsf{g}(\Omega, \Omega), 2]$.

Definition 10.1.5. *An* index tree I*, for a set of redex schemes* LHS_Ω*, is a* finite state automaton *which, in addition to the usual transfer function, also has a* failure function *as explained below. The set of final states is* LHS_Ω*. Nonfinal states are index points. The initial state is* $[\Omega, \varepsilon]$*. The transfer function of* I*, written* $\delta(s, f)$*, gives a new state of the automaton, given a state* s *and a symbol* f*. The transfer function is constructed so that* $\delta([M, u], f) = [M', u']$ *(or* $\delta([M, u], f) = M'$ *if* $M' \in \mathsf{LHS}_\Omega$*) only if* $M' = \mathsf{ext}(M, u, f)$*. As mentioned above, we define, in addition to the transfer function, a* failure function ϕ*, designed so that* $\phi(s) = t$ *if and only if* t *is the immediate failure point of* s*. For the initial state the failure function is undefined. For a final state both the transition function and the failure function are undefined.*

Remark: In an index tree some states may not be reachable from the initial state $[\Omega, \varepsilon]$ via transfer transitions only.

In the figures representing index trees, we show failure transitions only if they lead to a state which is not the initial state; the legend given in Figure 10.1 is valid for all the figures representing index trees (and also matching DAGs in sections 8.2 and 13.2.
The system $\mathcal{R}_1$ of Example A.0.1 admits an index tree. Such index tree is given in Figure 10.2.

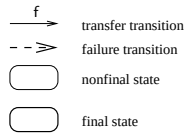


Figure 10.1: Legend for the representation of index trees

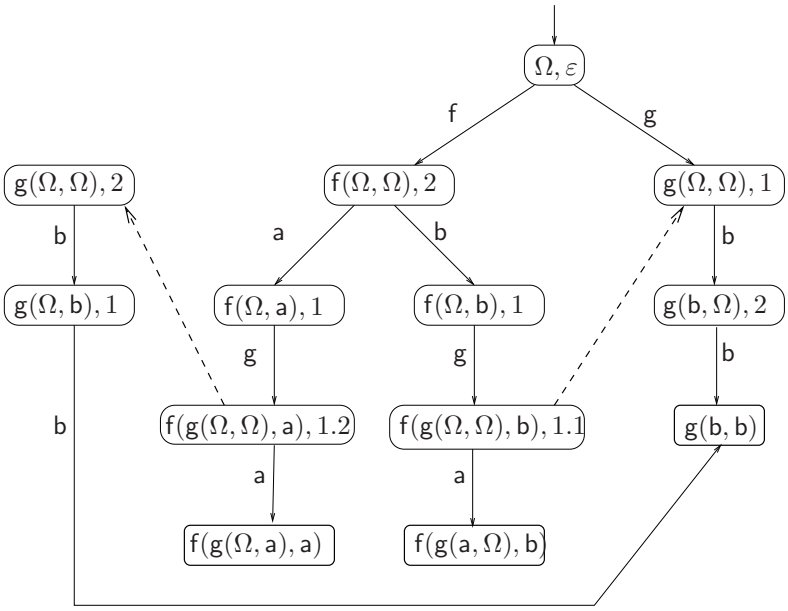


Figure 10.2: An index tree for $\mathcal{R}_1$ of Example A.0.1

10.2 Equivalence between index trees and matching DAGs

In this section we informally check that index trees are equivalent to matching DAGs. This section is not as important as the next but is needed to relate clearly Strandh's work to Huet & Lévy's work.

In his thesis Strandh just claimed that every strongly sequential system admits an index tree without giving a proof. The definition of an index tree differs from the construction of a matching DAG and there was a need to relate both notions.

We just check that a matching DAG is equivalent to an index tree. Then we have

$\mathcal{R}$ admits an index tree $\Leftrightarrow$ $\mathcal{R}$ admits a matching DAG $\overset{Th\ 8.2.1}{\Longleftrightarrow}$ $\mathcal{R} \in \mathsf{SS}$.

The following lemma is useful to relate matching DAGs and index trees.

Lemma 10.2.1. *[HL91] Let $[M, v]$ be an accessible node in a matching DAG.*

Then, $v \in \mathcal{I}(M)$.

It is clear that an index tree and a matching DAG have a similar structure: nonterminal accessible nodes correspond to nonfinal states and success nodes to final states. The main difference is that an index tree is deterministic while a matching DAG is not. Requirement a) for the construction of a matching DAG and Lemma 10.2.1 ensures that nonterminal nodes are index points. From Lemma 10.1.4, the failure function ϕ is identical to the $Fail$ function.

To obtain an index tree from a matching DAG:
for each nonterminal node $[M, v]$, for each function symbol f labeling more than one arc issued from $[M, v]$, we remove all arcs labeled f except one.

To obtain a matching DAG from an index tree:
for each pair of states $([M, v], [M', v'])$ (resp. each pair $([M, v], M')$ if M' is a final state) such that $M' = \mathsf{ext}(M, v, f)$ and such that there is no transition from $[M, v]$ to $[M', v']$ (resp. from $[M, v]$ to M') we add an arc $[M, v] \xrightarrow{f} [M', v']$ (resp. an arc $[M, v] \xrightarrow{f} M'$).

Example 10.2.2. *An equivalent matching DAG for the system* $\mathcal{R}_1$ *of Example A.0.1 is obtained by adding a transition labeled* g *from* $[\Omega, \varepsilon]$ *to* $[\mathsf{g}(\Omega, \Omega), 2]$ *to the index tree shown in Figure 10.2 and success tokens in the final states. The resulting matching DAG is given in Figure 8.1.*

It seems that Strandh's main motivation for defining index trees was to define the forward-branching class that we are going to introduce now.

10.3 Forward-Branching systems (FB)

10.3.1 Definition of FB

Definition 10.3.1. *An index tree is said to be a* forward-branching index tree *if every state of the index tree can be reached via transfer transitions from the initial state.*

Note that a forward-branching index tree is the same as a deterministic matching DAG (with no two transitions issued from the same state labeled with the same symbol).

Definition 10.3.2. *An orthogonal system* $\mathcal{R}$ *is* forward-branching *if there exists a forward-branching index tree for* $\mathcal{R}$.

The class of forward-branching systems is denoted by FB.

10.3.2 Characterization of FB

This sections presents a nice characterization of forward-branching systems that could be used as an alternative definition for the class when one wants to avoid the tedious definition of index tree. We first give an auxiliary lemma.

Lemma 10.3.3. *Let* $M \in \mathsf{LHS}_\Omega^\prec$ *and* $u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}^+(M)$. $M/u \notin \mathsf{LHS}_\Omega$.

Proof. As $M \in \mathsf{LHS}_\Omega^\prec$, $\exists N \in \mathsf{LHS}_\Omega$ such that $M \prec N$. Suppose that $M/u \in \mathsf{LHS}_\Omega$. Let L_N be the left-hand side associated with N and $L_{M/u}$ be the left-hand side associated with M/u. L_N matches $L_{M/u}$ which contradicts the non-ambiguity hypothesis. □

Property 10.3.4. $\forall M \in \mathsf{LHS}_\Omega^\prec$, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, *s.t.* $\forall N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$ *s.t.* $M \prec N'$, $N'/u \neq \Omega$.

This property states that every preredex M contains an Ω-position which is not an Ω-position in any scheme or subscheme greater than M.

The class characterized by Property 10.3.4 is denoted by K. We now show that $K = \mathsf{FB}$.

Proposition 10.3.5. $K = \mathsf{FB}$.

Proof. In the next two subsections, we prove that $K \subseteq \mathsf{FB}$ and $\mathsf{FB} \subseteq K$. So, $\mathsf{FB} = K$. □

Before the formal proof, we give examples that illustrate Proposition 10.3.5.

Example 10.3.6. *Example of a forward-branching system.*
$\mathcal{R}_3$ of Example A.0.3 is forward-branching as a forward-branching index tree exists for $\mathcal{R}_3$. We have $\mathsf{LHS}_\Omega = \{\mathsf{f}(\mathsf{g}(\Omega,\mathsf{a}),\mathsf{a}), \mathsf{f}(\mathsf{g}(\Omega,\mathsf{a}),\mathsf{b}), \mathsf{g}(\mathsf{b},\mathsf{b})\}$ *and* $\mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega) = \mathsf{LHS}_\Omega \cup \{\mathsf{g}(\Omega,\mathsf{a})\}$.
A forward-branching index tree for $\mathcal{R}_3$ is given in Figure 10.3.
It is easy to check that $\mathcal{R}_3$ satisfies Property 10.3.4.

Example 10.3.7. *Example of a non-forward-branching system.*
Recall system $\mathcal{R}_1$ of Example A.0.1. We have $\mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega) = \mathsf{LHS}_\Omega \cup \{\mathsf{g}(\Omega,\mathsf{a}), \mathsf{g}(\mathsf{a},\Omega)\}$.
$\mathcal{R}_1$ does not satisfy Property 10.3.4: let $M = \mathsf{g}(\Omega,\Omega)$; $M \prec \mathsf{g}(\mathsf{b},\mathsf{b}) \in \mathsf{LHS}_\Omega$;
$\mathcal{P}\mathsf{os}_{\overline{\Omega}}(M) = \{\ 1,\ 2\ \}$;
$M \prec \mathsf{g}(\Omega,\mathsf{a}) \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$ *and* $\mathsf{g}(\Omega,\mathsf{a})/1 = \Omega$;
$M \prec \mathsf{g}(\mathsf{a},\Omega) \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$ *and* $\mathsf{g}(\mathsf{a},\Omega)/2 = \Omega$.
No forward-branching index tree exists for $\mathcal{R}_1$.

10.3.3 Correctness of the characterization

Lemma 10.3.8. *In a forward-branching index tree, two index points* $[M,u]$ *and* $[M,v]$ *where* $u \neq v$ *cannot exist.*

Proof. See in [Str89] Lemma 4.3. □

Lemma 10.3.9. *Let* M_k, M, N *be* Ω*-terms such that* $M_k \preceq M \prec N$. *Let* $M_{k+1} = \mathsf{ext}(M_k, u_k, f_k)$ *(with* $u_k \in \mathcal{P}\mathsf{os}_\Omega(M_k)$ *and* $f_k = \mathsf{root}(N/u_k)$*). If* $M_{k+1} \not\preceq M$, *then* $u_k \in \mathcal{P}\mathsf{os}_\Omega(M)$.

The next lemma will be the main lemma for the proof of $\mathsf{FB} \subseteq K$. First we add one more definition.

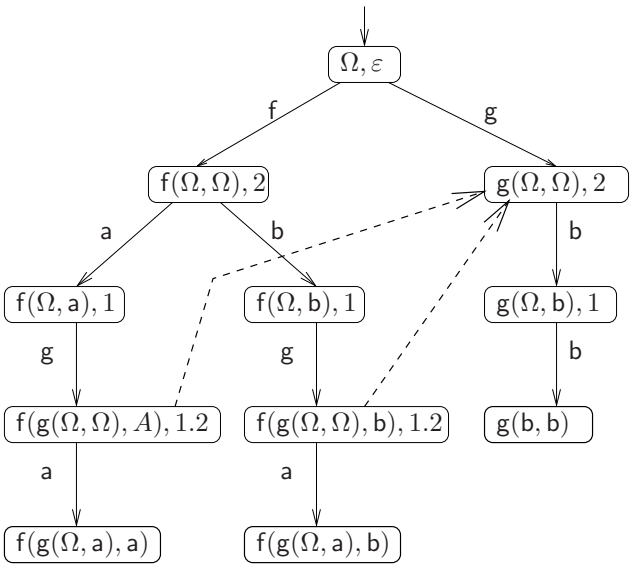


Figure 10.3: A forward-branching index tree for $\mathsf{LHS}_\Omega(\mathcal{R}_3)$

Definition 10.3.10. *A sequence of index points* $S = ([M_0, u_0], \ldots, [M_{n-1}, u_{n-1}])$ *is a* sequence of index points from M_0 to M_n *iff* $\forall i,\ 0 \leq i < n,\ M_{i+1} = \mathsf{ext}(M_i, u_i, f_i)$ *where* $f_i = \mathsf{root}(M_n/u_i)$.

Remark: A sequence of index points corresponds to a path in an index tree. An illustration of the following lemma is given in Figure 10.4.

Lemma 10.3.11. *(Failure points Lemma)*
In an index tree, let

$$([M_0, u_0], [M_1, u_1], \ldots, [M_k, u_k])$$

be a sequence of index points from $M_0 = \Omega$ *to* M_{k+1}. *In the same index tree, let*

$$([U_j, l_j], [U_{j+1}, l_{j+1}], \ldots, [U_{m-1}, l_{m-1}])$$

be a sequence of index points from U_j *to* U_m *such that* $M_k \prec U_m/l_j$. *Then* $\forall h$, $0 \leq h \leq k$, $[M_h, u_h]$ *is a failure point of* $[U_{j+h}, l_{j+h}]$ *and* $l_{j+h} = l_j u_h$.

Proof. This lemma is proven by induction on h.
if $h = 0$: $[M_0, u_0] = [\Omega, \varepsilon]$ is trivially a failure point of $[U_j, l_j]$ and $l_j = l_j\varepsilon$.
Induction step : $h < k$
By definition $M_{h+1} = \mathsf{ext}(M_h, u_h, f_h)]$ where $f_h = \mathsf{root}(M_{k+1}/u_h)$.
So, $\mathsf{root}(M_{h+1}/u_h) = f_h$. As $h < k$, we have $M_{h+1} \preceq M_k$. From $M_{h+1} \preceq M_k$ and the fact that $M_k \prec U_m/l_j$, we get $M_{h+1} \prec U_m/l_j$. From $\mathsf{root}(M_{h+1}/u_h) = f_h$ and $M_{h+1} \prec U_m/l_j$, we get $\mathsf{root}(U_m/l_j u_h) = f_h$. From the induction hypothesis, $[M_h, u_h]$ is a failure point of $[U_{j+h}, l_{j+h}]$ and $l_{j+h} = l_j u_h$, so $U_{j+h}/l_j = M_h$.

(1)
From $\mathsf{root}(U_m/l_j u_h) = f_h$ and $l_{j+h} = l_j u_h$, we get $U_{j+h+1} = \mathsf{ext}(U_{j+h}, l_j u_h, f_h)$ which gives $U_{j+h+1}/l_j = U_{j+h}/\mathsf{ext}(l_j, u_h, f_h)$. (2)
From (1) and (2), $U_{j+h+1}/l_j = \mathsf{ext}(M_h, u_h, f_h) = M_{h+1}$. Using Lemma 10.1.4, we obtain that $[M_{h+1}, l_{j+h+1}/l_j]$ is a failure point of $[U_{j+h+1}, l_{j+h+1}]$.
Using Lemma 10.3.8, we get $l_{j+h+1}/l_j = u_{h+1}$ which gives, $l_{j+h+1} = l_j u_{h+1}$. □

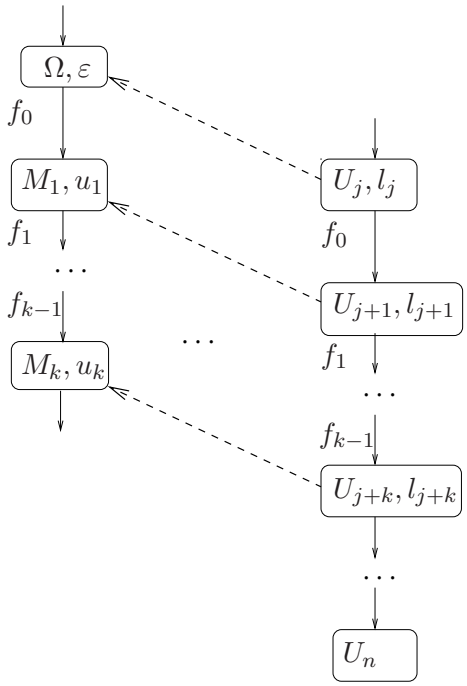


Figure 10.4: Illustration of the Failure points Lemma

Lemma 10.3.12. $\mathsf{FB} \subseteq K$.

Proof. Let $\mathcal{R}$ be a forward-branching system. Then, there exists a forward-branching index tree I for $\mathcal{R}$. Let us suppose that $\mathcal{R}$ does not satisfy Property 10.3.4. Then $\exists N \in \mathsf{LHS}_\Omega$, $\exists M \prec N$, $\forall u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\exists N' \in \mathsf{LHS}'_\Omega$ such that $M \prec N'$ and $N'/u = \Omega$. (1)
Let us consider in I,

$$([M_0, u_0], [M_1, u_1], \ldots, [M_{n-1}, u_{n-1}])$$

the sequence of index points from $M_0 = \Omega$ to $M_n = N$. Consider the first integer k, $0 \leq k < n$ such that $M_k \preceq M$ and $M_{k+1} \not\preceq M$. From Lemma 10.3.9, $u_k \in \mathcal{P}\mathsf{os}_\Omega(M)$. For $u = u_k$ in (1), $\exists N' \in \mathsf{LHS}'_\Omega$, such that $M \prec N'$ and $N'/u_k = \Omega$. As $M_k \preceq M$ and $M \prec N'$, we have $M_k \prec N'$.

CASE 1: $N' \in \mathsf{LHS}_\Omega$.
Then $\mathsf{nf_s}(N) = true$. From $N'/u_k = \Omega$, $M_k/u_k = \Omega$, $\mathsf{nf_s}(N)$ holds and $M_k \prec N'$, we get that $u_k \notin \mathcal{I}(M_k)$ which contradicts the hypothesis that $[M_k, u_k]$ is an index point.
CASE 2: $N' \notin \mathsf{LHS}_\Omega$.
By definition of LHS'_Ω, $\exists R \in \mathsf{LHS}_\Omega$, $\exists v \in \mathcal{P}\mathsf{os}(R)$, $v \neq \varepsilon$, such that $N' = R/v$. As $R \in \mathsf{LHS}_\Omega$, $\mathsf{nf_s}(R)$ holds. Let us consider in I,

$$([U_0, l_0], [U_1, l_1], \ldots [U_{m-1}, l_{m-1}])$$

the sequence of index points from $U_0 = \Omega$ to $U_m = R$. As $v \in \mathcal{P}\mathsf{os}(R)$ and $v \neq \varepsilon$, $\exists j$, $0 < j < m$ with $U_j/v = \Omega$ and $l_j = v$. As $M_k \prec N' = U_m/l_j$, we have $m - j > k$. So, we can apply Lemma 10.3.11and we get $\forall h$, $0 \leq h \leq k$, $[M_h, u_h]$ is a failure point of $[U_{j+h}, l_{j+h}]$ and $l_{j+h} = l_j u_h$. For $h = k$, we have $l_{j+k} = l_j u_k$. From $R/l_j u_k = N'/u_k$ and $N'/u_k = \Omega$, $R/l_j u_k = \Omega$. From $R/l_{j+k} = \Omega$, $U_{j+k}/l_{j+k} = \Omega$, $\mathsf{nf_s}(R)$ holds and $U_{j+k} \prec R$, $l_{j+k} \notin \mathcal{I}(U_{j+k})$ which contradicts the hypothesis that $[U_{j+k}, l_{j+k}]$ is an index point. □

Now we show the reverse implication *i.e* if a system satisfies the characterization, we can build an automaton which is a forward-branching index tree.

In the automaton, we denote by $\mathsf{nofinal}_n$ the set of nonfinal states reachable from the initial state $[\Omega, \varepsilon]$ via n transfer transitions. We call these states, *states of level n*. Let max be the maximum number of positions not labeled Ω in a scheme of LHS_Ω. $\bigcup_{n=0}^{\mathsf{max}-1} \mathsf{nofinal}_n$ is the set of nonfinal states reachable from the initial state $[\Omega, \varepsilon]$ via transfer transitions only.

The construction is done by induction on n. During the construction, we carry along three induction hypotheses:
- hypothesis *H0* corresponds to the forward-branching property of the automaton;
- hypothesis *H1* corresponds to the characterization; it is trivially true when the choice of u for a nonfinal state $[M, u]$ is done using Property 10.3.4 but it has to be proved when the choice of u depends on a choice previously made for a preredex which is also a subterm of M;
- hypothesis *H2* will be used to show that nonfinal states are index points.

Lemma 10.3.17 shows how to build the automaton. First we give the technical lemmata needed to verify that the automaton built in Lemma 10.3.17 is indeed a forward-branching index tree.

Lemma 10.3.13. *Let M_n be an Ω-term. Let u_n be a firm extension position of M_n. Let $M_{n+1} = M_n[f(\vec{\Omega})]_{u_n}$ with f a symbol. If u_{n+1}/w is a firm extension position of M_{n+1}/w, then u_{n+1} is a firm extension position of M_{n+1}.*

Proof. Consider $u \not< u_{n+1}$.
Case 1: $u_{n+1}/w < u$
As $u \not< u_{n+1}$, $u/w \not< u_{n+1}/w$. As u_{n+1}/w is a firm extension position of M_{n+1}/w, we get u_{n+1} is a firm extension position of M_{n+1}.
Case 2: $u_{n+1}/w \not< u$.

As M_n and M_{n+1} differ only at position u_n, we have $M_{n+1}/u = M_n/u \neq \Omega$. We also have $u \not< u_n$ (otherwise we would have $u < u_{n+1}$). So, u is a strongly stable position of M_{n+1}.

So all $u \not< u_{n+1}$ are strongly stable; this means that u_{n+1} is a firm extension position of M_{n+1}. □

The idea of the next lemma is that for a nonfinal state $[M, u]$, if the failure function goes to the root then all nonΩ-positions of M are strongly stable except the root. This means that any Ω-position of M is a firm extension position. Hypotheses *H1* and *H2* are the same as the ones in Lemma 10.3.17.

Lemma 10.3.14. *Let $\mathcal{R}$ be an orthogonal system satisfying Property 10.3.4. Consider that the automaton has been built up to level n, $0 < n < \mathsf{max}$, and that all nonfinal states $[M, u]$ of level $\leq n$ verify*

H1) $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$ such that $M \prec N$, $N/u \neq \Omega$,

H2) u is a firm extension position of M.

Let $[M_n, u_n] \in \mathsf{nofinal}_n$. Let $M_{n+1} = M_n[f(\vec{\Omega})]_{u_n}$ (with f a symbol).
If all nonfinal states t reachable from $[M_n, u_n]$ via failure transitions are such that $\delta(t, f)$ is undefined, then $\forall w \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M_{n+1})$ s.t. $w \neq \varepsilon$, w is a strongly stable position of M_{n+1}.

Proof. M_{n+1} and M_n differ only at position u_n ($M_{n+1}/u_n = f(\vec{\Omega})$ and $M_n/u_n = \Omega$). Using *H2* we get that $\forall w \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M_{n+1})$ such that $w \not\leq u_n$, w is strongly stable.

As $[\Omega, \varepsilon]$ is trivially a failure point of $[M_n, u_n]$ and $\delta([\Omega, \varepsilon], f)$ is undefined we know that f is a constructor symbol so u_n is strongly stable.

We now show that $\forall w \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M_{n+1})$ such that $w \neq \varepsilon$ and $w < u_n$, w is strongly stable. Suppose that this is not true and consider the greatest position $w \leq u_n$ such that $\omega(M_{n+1}/w)$ is a potential redex. As we consider the greatest position all nonΩ-positions of M_{n+1}/w (except the root) are strongly stable. As M_{n+1}/w is a potential redex, we get that $M_{n+1}/w \uparrow N$ for some $N \in \mathsf{LHS}_\Omega$. As $M_n < M_{n+1}$, we get also $M_n/w \uparrow N$. Using Lemma 10.3.15, this gives $M_n/w \prec N$. So $[M_n/w, u_n/w]$ is a failure point of $[M_n, u_n]$. By hypothesis *H1*, $N/u_n/w \neq \Omega$ so $\mathsf{root}(N/w) = f$. So $M_{n+1}/w \prec N$ and $\delta([M_n/w, u_n/w], f)$ should be defined, contradiction. □

The next Lemma shows that in the automaton the notion redex compatibility and the notion of preredex become equivalent.

Lemma 10.3.15. *Let $\mathcal{R}$ be an orthogonal system satisfying Property 10.3.4. Consider the automaton built with the construction described in Lemma 10.3.17. If $[M, u]$ is a nonfinal state, then $\forall w < u$, $\forall N \in \mathsf{LHS}_\Omega$ s.t. $M/w \uparrow N$, $M/w \prec N$.*

Proof. Let $w < u$ and $N \in \mathsf{LHS}_\Omega$ s.t. $M/w \uparrow N$. Suppose by contradiction that $M/w \not\prec N$. In the sequence of states $([M_0, u_0], [M_1, u_1], \ldots, [M_{k-1}, u_{k-1}])$ from $M_0 = \Omega$ to $M_k = M$ consider the first i such that $M_i/w \prec N$ and $M_{i+1}/w \not\prec N$.

$M_{i+1} = M_i[\mathsf{root}(M/u_i)(\overrightarrow{\Omega})]_{u_i}$. As $M_{i+1}/w \not\prec N$, we get $N/u_i \neq M_{i+1}/u_i$. Now, as $M/w \uparrow N$, we get $N/u_i = \Omega$ which contradicts *H1* applied to $[M_i, u_i]$. □

Lemma 10.3.16. *Let $\mathcal{R}$ be an orthogonal system satisfying Property 10.3.4. Consider the automaton built with the construction described in Lemma 10.3.17. If $[M, u]$ is a nonfinal state, then $u \in \mathcal{I}(M)$.*

Proof. As M is a preredex $\omega(M) = \Omega$. Suppose that $u \notin \mathcal{I}(M)$. Then, by Proposition 8.1.17 we have $\omega(M[\bullet]_u) = \omega(M) = \Omega$. As the symbol $\bullet$ does not appear in LHS_Ω, all nonfinal states t reachable from $[M, u]$ via failure transitions are such that $\delta(t, \bullet)$ is undefined. From Lemma 10.3.14, we get that $\forall w \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M)$ s.t. $w \neq \varepsilon$, w is a strongly stable position of $M[\bullet]_u$. So, the only possibility for having $\omega(M[\bullet]_u) = \Omega$ is to have $M[\bullet]_u \uparrow N$ with some redex scheme N. It follows that $M \uparrow N$. Using Lemma 10.3.15 and *H1*, we get $M \prec N$ (1). It follows that $u \in \mathcal{P}\mathsf{os}(N)$. As the symbol $\bullet$ cannot appear in N we must have $N/u = \Omega$ (2). (1) and (2) contradict hypothesis *H1*. □

The next lemma gives the construction of the automaton.

Lemma 10.3.17. *(Construction Lemma)*
Let $\mathcal{R}$ be an orthogonal system satisfying Property 10.3.4. We can build an automaton such that $\forall n$, $0 < n < \mathsf{max}$, $\forall [M, u] \in \mathsf{nofinal}_n$,

H0) $\phi([M, u]) \in \bigcup_{k=0}^{n-1} \mathsf{nofinal}_k$ (forward-branching property),
H1) $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$ such that $M \prec N$, $N/u \neq \Omega$,
H2) u is a firm extension position of M.

Proof. In the initialization, we create the initial state and the states of level 1. If all states of level $\leq n$ have been created, the induction step creates the states of level $n + 1$.

Initialization:

let $\mathsf{nofinal}_0 = \{[\Omega, \varepsilon]\}$; for all i, $0 < i < \mathsf{max}$, let $\mathsf{nofinal}_i = \emptyset$;
if $n = 1$: Construction of $\mathsf{nofinal}_1$
  for all $f \in \mathcal{F}$ s.t. $\exists N \in \mathsf{LHS}_\Omega$ with $\mathsf{root}(N) = f$,
    let $M_1 = f(\overrightarrow{\Omega})$;
    if $M_1 \in \mathsf{LHS}_\Omega$ then let $s_1 = M_1$
    else let $s_1 = [M_1, u_1]$ with u_1 **chosen** as follows;
      add s_1 to $\mathsf{nofinal}_1$;
  let $\delta([\Omega, \varepsilon], f) = s_1$;

Choice of u_1: We **choose** u_1 so that $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(LHSo)$ such that $M_1 \prec N$, $N/u_1 \neq \Omega$. This is possible because $\mathcal{R}$ satisfies Property 10.3.4. Then, *H1* is true for s_1.
Trivially, $\phi(s_1) = (\Omega, \varepsilon)$ so *H0* is true for s_1. *H2* is trivially true.

Induction step: Construction of $\mathsf{nofinal}_{n+1}$ $(n < \mathsf{max})$

for all $s_n = [M_n, u_n] \in \mathsf{nofinal}_n$,
 for all $f \in \mathcal{F}$ s.t. $\exists N \in \mathsf{LHS}_\Omega$ s.t. $M_n \prec N$ and $\mathsf{root}(N/u_n) = f$,
  let $M_{n+1} = M_n[f(\vec{\Omega})]_{u_n}$;
  if $M_{n+1} \in \mathsf{LHS}_\Omega$ then let $s_{n+1} = M_{n+1}$
  else
   let $s_{n+1} = [M_{n+1}, u_{n+1}]$ with u_{n+1} **chosen** as follows;
   add s_{n+1} to $\mathsf{nofinal}_{n+1}$;
let $\delta(s_n, f) = s_{n+1}$;

Choice of u_{n+1}. Let us consider the sequence of index points reachable from s_n following failure transitions only.

Case 1: none of these index points t is such that $\delta(t, f)$ is defined. This case is illustrated by Figure 10.5.
We **choose** u_{n+1} so that $\forall N \in \mathcal{S}\mathsf{ub}_\mathcal{D}(LHSo)$ such that $M_{n+1} \prec N$, $N/u_{n+1} \neq \Omega$, which is possible because $\mathcal{R}$ satisfies Property 10.3.4. Then, *H1* is true for s_{n+1}.
Trivially, $\phi([M_{n+1}, u_{n+1}]) = [\Omega, \varepsilon]$ and *H0* is true for s_{n+1}.
From Lemma 10.3.14, we get that the only potential redex position of M_{n+1} is ε.
This means that any Ω-position (in particular u_{n+1}) is a firm extension position of M_{n+1}, then *H2* is true. It also means that $\phi([M_{n+1}, u_{n+1}]) = [\Omega, \varepsilon]$, then *H0* is true.

Case 2: there exists a state $t = [U, v]$ reachable from s_n via failure transitions only such that $\delta(t, f)$ is defined. This case is illustrated by Figure 10.6.
Let $t' = \delta(t, f)$.
As t is a failure point of $s_n = [M_n, u_n]$, $\exists u$ such that $U = M_n/u$ and $u_n = uv$.
From the induction hypothesis *H0*, as we follow at least one failure transition,
 $t \in \bigcup_{k=0}^{n-1} \mathsf{nofinal}_k$.
The prefix appearing in t' is
 $M_n/u[f(\vec{\Omega})]_v = M_n[f(\vec{\Omega})]_{uv}/u = M_n[f(\vec{\Omega})]_{u_n}/u = M_{n+1}/u$.
From Lemma 10.3.3, M_{n+1}/u is not a redex scheme. So, t' is a nonfinal state of the form $[M_{n+1}/u, v']$.
As t' is reachable via one transfer transition from t, $t' \in \bigcup_{k=1}^{n} \mathsf{nofinal}_k$.
We choose $u_{n+1} = uv'$.
Then we have $\phi(s_{n+1}) = t'$, so *H0* is true for s_{n+1}.
Fact: *H1* is true for s_{n+1}.
 Proof: $\forall N \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$, as $M_{n+1} \prec N$, $M_{n+1}/u \prec N/u$. (1)
 As $[M_{n+1}/u, v'] \in \bigcup_{k=1}^{n} \mathsf{nofinal}_k$, $\mathsf{root}(M_{n+1}/u) \notin \mathcal{C}$. (2)
 From (1) and (2), $N/u \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$. As $[M_{n+1}/u, v'] \in \bigcup_{k=1}^{n} \mathsf{nofinal}_k$ and $N/u \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$ and $M_{n+1}/u \prec N/u$, the induction hypothesis *H1* applies and $N/u/v' \neq \Omega$. This implies that $N/uv' \neq \Omega$. So *H1* is true for s_{n+1}.
H2 is true in **case 2** by Lemma 10.3.13. □

To prove the next lemma, we show that the automaton built in Lemma 10.3.17

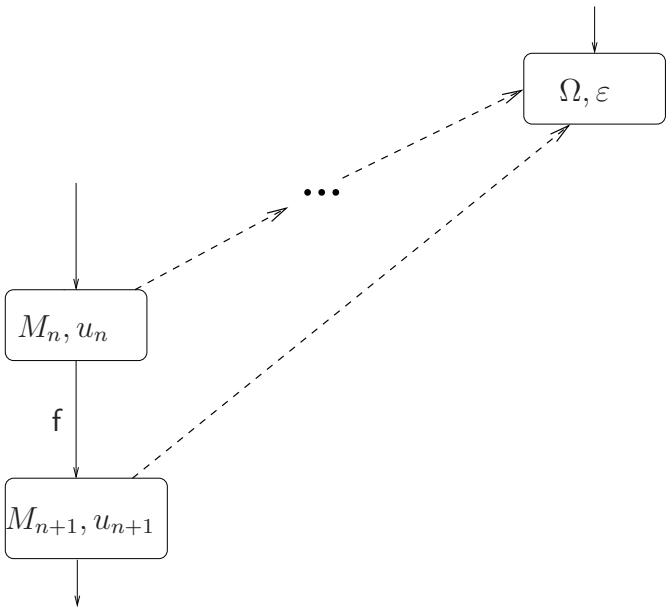


Figure 10.5: Illustration the proof of the Construction Lemma, case 1

is an index tree.

Lemma 10.3.18. $K \subseteq \mathsf{FB}$.

Proof. Let $\mathcal{R}$ be a system in K; by definition it satisfies Property 10.3.4. By Lemma 10.3.17 we can build an automaton for such that:

$\forall n,\ 0 < n < \mathsf{max},\ \forall [M,u] \in \mathsf{nofinal}_n,$

H0) $\phi([M,u]) \in \bigcup_{k=0}^{n-1} \mathsf{nofinal}_k$ (forward-branching property),

H1) $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$ such that $M \prec N$, $N/u \neq \Omega$.

H2) u is a firm extension position of M.

We now prove that nonfinal states of the automaton are index points:

Proof: Let $[M,u]$ be a nonfinal state. By *H2*, u is a firm extension position. Using Lemma 10.3.16, we get that $u \in \mathcal{I}(M)$. We conclude that $[M,u]$ is an index point.

So, the automaton is an index tree. *H0* ensures that the index tree is forward-branching. We conclude that $\mathcal{R} \in \mathsf{FB}$. □

We conclude that forward-branching systems are characterized by Property 10.3.4.

Theorem 10.3.19. *$\mathcal{R} \in \mathsf{FB}$ if and only if $\forall M \in \mathsf{LHS}_\Omega^{\prec}$, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, s.t. $\forall N' \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$ s.t. $M \prec N'$, $N'/u \neq \Omega$.*

Sometimes, when index trees are not relevant, it is convenient to use this characterization as an alternative definition for FB.

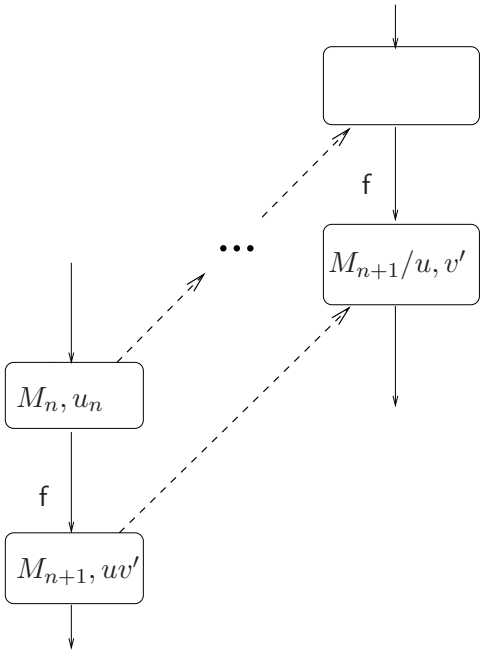


Figure 10.6: Illustration the proof of the Construction Lemma, case 2

10.4 An algorithm to build a forward-branching index tree

The constructive proof of Lemma 10.3.17 can be transformed into an algorithm.

10.4.1 General description of the algorithm

The **input** of the algorithm is a set of (pointers to) schemes $\mathsf{LHS}_\Omega = \{N_1, \ldots N_n\}$. The **output** is a (pointer to) the initial index point of a forward-branching index tree if the system is forward-branching. Otherwise the output is an error message ("not Strongly Sequential" or "not Forward-branching"). First, the algorithm computes the set of (pointers to) subschemes $\{N_{n+1}, \ldots, N_{n+n'}\}$. Then it builds the index tree breadth-first starting from the initial index point $s_0 = (\Omega, \varepsilon)$. Consequently, index points of level i are built before index points of level $i+1$. For all non-final index points $[M, u]$ and for all symbols f found at position u in the schemes greater than M, we refine M to obtain $M' = \mathsf{ext}(M, u, f)$. If M' is a scheme, we have reached a final index point. Otherwise, we have a non-final index point. We choose u' as described in the constructive proof of Lemma 10.3.17.

10.4.2 Variables and data structures

To avoid overloading the text of the algorithm, we first give a description of the variables used in the algorithm.

- i always represents a level in the index tree.
- j always corresponds to the number of a scheme (or subscheme) N_j.
- u always represents (a pointer to) a position in an Ω-term.
- o always represent (a pointer to) an Ω-position of the prefix M in a given index point.
- f always represents a symbol.
- k, $1 \leq k \leq \mathsf{arity}(f)$, represents the number of a child of a position labeled f.
- s (index point), cs (child index point), fs (failure index point) always represent (pointers to) index points.
- s_0 represents the initial index point of the index tree.

Representation of symbols: a symbol f is represented by a structure containing the following fields: $f.name$, the name of the symbol and $f.arity$, the arity the symbol.

Representation of terms: a term N is represented by a structure containing the following fields: $N.symbol$, the root symbol f and $N.subterms$, an array of size $p = f.arity$ containing the subterms $N/1, \ldots, N/p$.

Representation of the transfer and failure functions: $\phi(s)$ gives (a pointer to) the failure index point of s and $\delta(s, f)$ gives (a pointer to) the index point accessible from s with a transition labeled f. The representation of δ is distributed among the index points: each index point s contains the list of pairs (f, cs) such that $\delta(s, f) = cs$.

Representation of an index point: an index point $s = [M, u]$ is represented by a structure containing the following fields (the representation of an Ω-position is given further):

- $s.index$: will contain the chosen index position u in case of a non-final index point;
- $s.schemes$: linked list of the numbers of the schemes greater than M.
- $s.subschemes$: linked list of the numbers of the subschemes greater than M.
- $s.omegas$: linked list of the Ω-positions of M.
- $s.match$: in case of a final index point, number of the matching scheme; otherwise -1.
- $s.symbol$: symbol f of the incoming transition.
- $s.parent$: parent index point.

Note that we do not need to explicitly represent the prefix M; it is implicitly represented by the transitions from s_0 to s and the list of its Ω-positions.

Representation of a position u: an Ω-position u of a prefix M is represented by a structure containing two fields:

- $u.path$: the sequence of positive integers corresponding to the position,

- $u.pointers$: for each position u of M we maintain an array which contains at entry j a pointer to N_j/u for each scheme or subscheme N_j greater than M; this field enables us to replace operations of the type $f := \mathsf{root}(N_j/u)$ by a constant time operation: $f := \mathsf{root}(u.pointers[j])$.

Representation of an Ω-position o: an Ω-position o of a prefix M of an index point is represented by a structure containing two fields: $o.position$ is a pointer to a position and $o.possible$ is a boolean value indicating whether that Ω-position can be chosen as an index; this value depends on the presence of Ωs at that same position in schemes or subschemes greater than M.

10.4.3 Algorithm

The main algorithm is given by the forward-branching-index-tree procedure given in Figure 10.11. We first present the procedures and functions called by the forward-branching-index-tree procedure. When the time complexity of a function or a procedure is obvious we indicate it by a comment.

Computation of the failure function

This procedure is shown in Figure 10.7. Note that as the automaton is computed in a breadth-first manner, any state reachable via failure transitions from the parent state of s has already been computed.

```
procedure compute-failure-transition(s);    /* O(level(s)) */
begin
φ(s) := s_0; /* default value */
fs := φ(s.parent)
while fs is defined do
      ts := δ(s.symbol, fs);
      if ts is defined then
            φ(s) := ts;
            return;
      fs := φ(fs);
end
```

Figure 10.7: The compute-failure-transition algorithm

Choice of an index position u for a nonfinal state $s = [M, u]$

We have seen in the proof of Lemma 10.3.17 that there are two cases for the choice of an index position u of a nonfinal state $s = [M, u]$. They are materialized by two functions.

The first case is when the failure function gives the initial state ($\phi(s) = s_0$). In that case, we can choose any Ω-position such that no scheme or subscheme greater than M has an Ω at position u. As that information is saved in the field *v.possible* of all Ω-positions of *s.omegas*, we choose the first marked possible position of *s.omegas*. All this is done by the **function** any-index. In the worst case, we choose the last one of the list, so the time complexity is in $O(|s.omegas|)$.

function any-index(*s.omegas*):position; /* $O(|s.omegas|)$ */
extracts and returns from the list *s.omegas* the first "possible" position.

The second case is when $\phi(s) = (U, v) \neq s_0$. In that case, there is a w such that $U = M/w$ and we need to choose $u = wv$ in *s.omegas*. For this, we remark that if $fs.omegas = \{v_1, v_2, \ldots, v_k\}$, then necessarily $s.omegas = \{wv_1, wv_2, \ldots, wv_k, \ldots\}$. So v has the same rank r in *fs.omegas* as wv in *s.omegas*. So we follow both lists in parallel and when we find the index of s in *s.omegas* we have found the right index in *fs.omegas*. This is done by the function given-index. In the worst case, we pick up the last one of the list; this happens when $s.omegas = \{wv_1, wv_2, \ldots, wv_k\}$ and $r = k$. So the time complexity is the same as in the first case.

function given-index(*s*):position; /* $O(|\phi(s).omegas|)$ */
gets from *s.omegas* the position which is at the same rank than $\phi(s).index$ in $\phi(s).omegas$.

The match-scheme function

The match-scheme function is given in Figure 10.8. Applied to an index point and a scheme number j, it returns true if the implicit prefix of s matches N_j.

```
function match-scheme(s,j):boolean;    /* O(|s.omegas|) */
begin
for each u ∈ s.omegas do
    if root(u.pointers[j]) ≠ Ω then return false;
return true;
end
```

Figure 10.8: The match-scheme algorithm

The check-directions function

The check-directions function given in Figure 10.9 returns true if no j in *numbers* is such that $N_j/u = \Omega$ and returns false otherwise.

```
function check-directions(u, numbers):boolean;    /* O(|numbers|) */
begin
for each j ∈ numbers do
    if u.pointers[j] = Ω then return false;
return true;
end
```

Figure 10.9: The check-directions

The possible-omegas algorithm

The possible-omegas algorithm given in Figure 10.10 updates the *possible* flags of the Ω-positions according to a given set of (sub)schemes numbers.

```
function possible-omegas(omegas,numbers):integer;
/* O(|omegas| × |numbers|) */
begin
npossible := 0;
for each u ∈ s.omegas do
    if u.possible then
        if check-directions(u, numbers) then
            npossible := npossible + 1;
        else u.possible := false;
return npossible;
end
```

Figure 10.10: The possible-omegas algorithm

The forward-branching-index-tree algorithm

Finally, the forward-branching-index-tree procedure is presented in Figure 10.11. After some initializations, the main while loop takes care of an index point s of the *todo* list at a time. It checks whether s is a final index point; if not it computes its index and creates its children index points. The algorithm clearly terminates as each time we remove from the *todo* list an index point of level i and if nonfinal replace it by index points of level $i + 1$. So we will necessarily reach the final index points.

```
    procedure forward-branching-index-tree({N_1, ..., N_n});
    begin
    {N_{n+1}, ..., N_{n+n'}} := extract-subschemes({N_1, ..., N_n});
    u := make-position(ε);
    for i := 1 to n+n' do u.pointers[i] := N_j;
    s_0 := make-index point(); s_0.omegas := { make-omega(u) };
    s_0.schemes := {1, ..., n}; s_0.subschemes := {n+1, ..., n+n'};
    todo := {s_0};
    while todo ≠ ∅ do
        begin
        s := pop(todo);
I1:     for each j ∈ s.schemes do
            if match-scheme(s, j) and s.match > 0 then
                return "Overlapp between schemes"
            else s.match := j;
        if s.match > 0 then
            if length(s.subschemes) > 0 then
                return "Overlapp beween scheme and subschemes"
            else break;
        begin
I2:     compute-failure-transition(s);
I3:     if possible-omegas(s.omegas, s.schemes) = 0 then
            return "not Strongly Sequential";
I4:     if φ(s) = s_0 and possible-omegas(s.omegas, s.subschemes) = 0
        then
            return "not Forward-Branching";
I5:     if φ(s) = s_0 then s.index := any-index(s.omegas);
        else s.index := given-index(s);

        u := s.index;
        for each f ∈ {root(u.pointers[j]) | j ∈ s.schemes} do
            begin
            cs := make-index-point(); cs.parent := s; cs.symbol := f;
            cs.omegas := duplicate-omegas-except-index(s.omegas,u));
            /* cs.schemes and cs.subschemes contain the j s.t. root(N_j/u) = f. */
            cs.schemes := eject(s.schemes, u, f);
            cs.subschemes := eject(s.subschemes, u, f);
            for k := f.arity downto 1 do
                begin
                v := make-position(extend-position(u.position,k));
                for each j ∈ cs.schemes ∪ cs.subschemes do
                    v.pointers[j] := u.pointers[j].subterms[k];
                push(v, cs.omegas);
                end
            add-to-end(cs,todo)
        end
    return s_0;
    end
```

Figure 10.11: The forward-branching-index-tree algorithm

10.4.4 Time complexity of the algorithm

The *size* of an Ω-term N is the number of its positions: $size(N) = |\mathcal{P}\mathsf{os}(N)|$. The size of a set of Ω-terms $\{N_1, \ldots, N_n\}$ is the sum of the sizes of all Ω-terms.

$$size(\{N_1, \ldots, N_n\}) = \sum_{j=1}^{n} size(N_j)$$

We define also $size_{\overline{\Omega}}(N)$ as the number of nonΩ-positions of N: $size_{\overline{\Omega}}(N) = |\mathcal{P}\mathsf{os}_{\overline{\Omega}}(N)|$ and for a set $\{N_1, \ldots, N_n\}$,

$$size_{\overline{\Omega}}(\{N_1, \ldots, N_n\}) = \sum_{j=1}^{n} size_{\overline{\Omega}}(N_j)$$

If LHS_Ω is the input set of schemes, we will denote $size(\mathsf{LHS}_\Omega)$ by S and $size_{\overline{\Omega}}(\mathsf{LHS}_\Omega)$ by $\overline{S}$. We also denote by l_{max} the maximal number of non-Ω-positions in a scheme of LHS_Ω: $l_{\mathsf{max}} = \mathsf{max}(\{size_{\overline{\Omega}}(N_1), \ldots, size_{\overline{\Omega}}(N_n)\})$. l_{max} is also the maximal level of index points in the index tree.

The time complexity of the algorithm will be given in function of S.

The next lemma gives a few trivial relations. We recall that n' is the number of (proper) subschemes.

Lemma 10.4.1.
$n \leq n + n' \leq \overline{S} \leq S$.
$\sum_{j=1}^{n} size(N_j)^2 \leq (\sum_{j=1}^{n} size(N_j))^2$.
The number of index points is $\leq \overline{S} + 1$.
The number of final index points is n.
The number of nonfinal index points is $\leq \overline{S} + 1 - n \leq \overline{S}$.
$l_{\mathsf{max}} \leq \overline{S}$.

We now evaluate the time complexity of the algorithm.

We have numbered the instructions which do not contribute to the creation (allocation) and initialization of the data structure of the index tree. The total time complexity will be the time taken by the numbered instructions plus the time to build and initialize the data structure which is proportional to the size of the data structure (space complexity).

We start with the space complexity as the time taken by the numbered instructions depends on the size of the structure.

Space complexity

The maximal number of non initial index points is S, For every $N \in \mathsf{LHS}_\Omega$ and every non-Ω-position u in N, u is allocated at most once. Each position u uses an array $u.pointers$ of size $n + n'$ so the space used by the positions all the positions is at most proportional to $(n + n')S \leq \overline{S}S$. The total space to represent the transfer function δ is proportional to $\overline{S}$.

In an index point s, the fields that are not of fixed length are $s.schemes$, $s.subschemes$ and $s.omegas$. We shall sum up the sizes of these fields for all the index points.

Given a level l in the index tree, the lists $s.schemes$ (resp. $subschemes$) of all index points s_l of level l form a partition of $\{N_1, \ldots, N_n\}$ (resp. $\{N_{n+1}, \ldots, N_{n+n'}\}$). For all levels we obtain a maximum size occupied by the $s.schemes$ and $s.subschemes$ of all index points of

$$(n + n')l_{\mathsf{max}} \leq \overline{S}^2$$

For a scheme N, we consider the sequence of transitions and index points from the initial index point s_0 to the index point matching N:

$$(s_l = (M_l, u_l) \xrightarrow{F_{l+1}})_{l=0,\ldots,size_{\overline{\Omega}}(N)-1}$$

Only one such sequence leads to N. Before we go one with the space complexity we give two easy lemmata.

Lemma 10.4.2.

$$|s_l.omegas| = size(M_l) - l \leq size(N)$$

Lemma 10.4.3. $\displaystyle\sum_{l=1}^{size_{\overline{\Omega}}(N)} |s_l.omegas| \leq size(N)^2$

Proof. From Lemma 10.4.2, $|s_l.omegas| \leq size(N)$. So,

$$\sum_{l=1}^{size_{\overline{\Omega}}(N)} |s_l.omegas| \leq \sum_{l=1}^{size_{\overline{\Omega}}(N)} size(N) = size_{\overline{\Omega}}(N) * size(N) \leq size(N)^2$$

□

If we sum up the length of the $s.omegas$ lists for all the schemes we obtain a majorant of the length of all the $s.omegas$ for all index points s (as some index points concern several schemes). So we obtain an upper bound of

$$\sum_{j=1}^{n} size(N_j)^2 \leq S^2$$

Lemma 10.4.4. *The space complexity is in* $\mathcal{O}(S^2)$.

Proof. The space is proportional to the number of index points ($\mathcal{O}(S)$) plus the space to represent the lists *omegas*, *schemes* and *subschemes* ($\mathcal{O}(S^2)$). Then we obtain a total space in $\mathcal{O}(S^2)$. □

Time complexity

Lemma 10.4.5. *The global time of instruction* **I2** *is in* $O(\overline{S}^2)$.

Proof. For an index point s of level l, the time taken by the call to compute-failure-transition(s) is proportional to l. Let C be the global time. We have

$$\begin{aligned} C &\leq K \sum_{j=1}^{n} \sum_{i=1}^{size_{\overline{\Omega}}(N_j)-1} i = K \sum_{j=1}^{n} [size_{\overline{\Omega}}(N_j) - 1] * size_{\overline{\Omega}}(N_j)/2 \\ &\leq K \sum_{j=1}^{n} size_{\overline{\Omega}}(N_j)^2/2 \leq K\overline{S}^2/2 \end{aligned}$$

with K a constant. □

In fact, it is possible to show that the global time for computing failures transitions is in $O(\overline{S})$. As this does not change the global complexity of the forward-branching-index tree algorithm, we do not give the proof but just the intuitive idea.

In [AC75], Aho and Corasick give an efficient algorithm which finds occurrences of words belonging to a finite set of words $W = \{W_1, \ldots, W_n\}$ in a text. The algorithm uses a finite state automaton where states are prefixes of words of W. The initial state is the empty word. The final states are the words of W. The transfer function δ is defined for a nonfinal state M and a symbol F as follows:
if MA is a prefix of a word of W then $\delta(M, F) = MA$ otherwise $\delta(M, F)$ is undefined. The failure function ϕ is defined for each nonfinal state M which is not the initial state as $\phi(M)$ is the longest suffix which is prefix of a word of W (note that $\phi(M)$ can be the empty word).

Now we remark that our automaton has exactly the same structure (in terms of transfer and failure transitions) as an automaton which would recognizes the set of words $W = \{W_1, \ldots, W_n\}$ with each $W_j = F_1 \ldots F_{size_{\overline{\Omega}}(N_j)}$ where $(F_l)_{i=1,\ldots size_{\overline{\Omega}}(N_j)}$ are the symbols of N_j which label the transfer transitions going from (Ω, ε) to the index point matching N_j.

For Example A.0.3 for which a forward-branching automaton is given in Figure 10.3, the corresponding set of words would be $\{\mathsf{faga}, \mathsf{fbga}, \mathsf{gbb}\}$.

In [AC75], the authors show that the time for computing failure transitions requires a time proportional to the sum of the length of the words of W.

As we compute failure transitions exactly in the same way as [AC75], we can conclude that the global time for computing failure transitions is proportional to

$$\sum_{j=1}^{n} length(W_j) = \sum_{j=1}^{n} size_{\overline{\Omega}}(N_j) = \overline{S}.$$

Lemma 10.4.6. *The global time of instructions* **I1** *and* **I3** *is in* $\mathcal{O}(S^2)$.

Proof. The time taken by instructions **I1** and **I3** is proportional to the number of comparisions $o.position.pointers[j] = \Omega$ they perform for all $o \in s.omegas$ and $j \in s.schemes$. At each level l, the sets of the $o.position.pointers[j]$ examined in each index point form a partition of the set of $o.position.pointers[j]$ for all

j, $1 \leq j \leq n$ and $o.position \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(N_j)$. The total number of comparisons is $\leq Sl_{\mathsf{max}}$ which using Lemma 10.4.1 is $\leq S^2$. □

Lemma 10.4.7. *The global time of instruction* **I5** *is in* $\mathcal{O}(S^2)$.

Proof. Each nonfinal index point of level l induces one call to the instruction either any-index(s_l) or extract-index(s_l). Both take $\mathcal{O}(|s_l.omegas|)$. With the same arguments as for the space complexity, we obtain a global cost in $\mathcal{O}(S^2)$. □

We now evaluate the time taken by instruction **I4** more precisely the time taken by possible-omegas($s.omegas$, $s.subschemes$) as the check $\phi(s) = s_0$ is done in constant time so takes a global time in $\mathcal{O}(S)$.

Lemma 10.4.8. *The global time of instruction* **I4** *is in* $\mathcal{O}(S^2)$.

Proof. It is fundamental to note that there is a call to possible-omegas(N,s) **only if** $\phi(s) = s_0$. The time taken by possible-omegas($s.omegas$, $s.subschemes$) is proportional to the number of comparisons $o.position.pointers[j] = \Omega$ they perform for all $o \in s.omegas$ and $j \in s.subschemes$. But we do no more of these comparisons that the ones done for schemes in instruction **I3**: for each s such that $\phi(s) = (\Omega, \varepsilon)$, each time we do a comparison $v.pointers[j] = \Omega$ with $j \in s.subschemes$ we do a corresponding comparison $u.v.pointers[j'] = \Omega$ in an index point s' such that $\phi^{-1}(s') = \varnothing$, $j' \in s'.schemes$ and $N_j = N_{j'}/u$. So the number of comparisons with subschemes positions is less that the number of comparisons with the schemes. We conclude that the global time taken by instruction **I4** is in $\mathcal{O}(S^2)$. □

Theorem 10.4.9. *The time complexity of the forward-branching-index tree algorithm is quadratic.*

Proof. From Lemma 10.4.4 the time to allocate and initialize the data structure corresponding to the forward-branching index tree is in $\mathcal{O}(S^2)$. From Lemmata 10.4.5, 10.4.6, 10.4.7, 10.4.8, the time spent in examining the structure is also in $\mathcal{O}(S^2)$. □

10.5 Modularity of FB

From Theorem 10.3.19, it is easy to prove that the forward-branching property is a modular property of orthogonal systems even for constructor sharing combinations. This is an unpublished result.

Theorem 10.5.1. *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be orthogonal constructor-sharing systems. If* $\mathcal{R}_1, \mathcal{R}_2 \in \mathsf{FB}$ *then* $\mathcal{R}_1 \cup \mathcal{R}_2 \in \mathsf{FB}$.

Proof. We will show that $\mathcal{R}_1 \cup \mathcal{R}_2$ verfy the characterization. Let $M \in \mathsf{LHS}^{\prec}_{\Omega}(\mathcal{R}_1 \cup \mathcal{R}_2)$. As $\mathcal{R}_1$ and $\mathcal{R}_2$ do not share defined symbols we have necessarily either $\mathcal{R}_1 \in \mathsf{LHS}^{\prec}_{\Omega}(\mathcal{R}_1)$ or $\mathcal{R}_2 \in \mathsf{LHS}^{\prec}_{\Omega}(\mathcal{R}_2)$. W.l.o.g, we assume that $\mathcal{R}_1 \in \mathsf{LHS}^{\prec}_{\Omega}(\mathcal{R}_1)$.

From Theorem 10.3.19, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, s.t. $\forall N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega(\mathcal{R}_1))$ s.t. $M \prec N'$, $N'/u \neq \Omega$. Using again the hypothesis that $\mathcal{R}_1$ and $\mathcal{R}_2$ do not share defined symbols, we get that $\{N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega(\mathcal{R}_1))\mathtt{s.t.}M \prec N'\} = \{N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega(\mathcal{R}_1 \cup \mathcal{R}_2))\mathtt{s.t.}M \prec N'\}$. It follows that $\forall N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega(\mathcal{R}_1 \cup \mathcal{R}_2))$ s.t. $M \prec N'$, $N'/u \neq \Omega$. □

Chapter 11

Constructor Equivalent systems

11.1 Simulating SS with C

In [Tha85], Thatte demonstrated the possibility of simulating an orthogonal system with a left-linear *constructor system* obtained from the original system via a simple transformation. With examples, we point out that Thatte's transformation does not always preserve orthogonality and that if it does, it does not always preserve strong sequentiality. This leads us to define a new subclass of SS: the class of *constructor-equivalent* systems (CE) for which Thatte's transformation preserves strong sequentiality. We give a simple characterization of CE and show that it is a subclass of FB. Most of the work presented in this chapter is published in [DS93] and [DS94].

11.1.1 Thatte's Transformation

This subsection is almost directly taken from [Tha85].

Let $\mathcal{R}$ be an orthogonal system. With each $f \in \mathcal{F}_{\mathcal{D}}$ associate a new (constructor) symbol c_f. Let $\mathcal{F}_{\#} = \mathcal{F} \cup \{ c_f \mid f \in \mathcal{F}_{\mathcal{D}} \}$. Let $t'(T)$ denote the term T with every inner occurrence of $f \in \mathcal{F}_{\mathcal{D}}$ replaced by c_f and $t''(T)$ the term with all occurrences so replaced. $\mathcal{R}_{\#}$ is the smallest system which satisfies the following two assertions:
(1) If $L \to R \in \mathcal{R}$ then $t'(L) \to R \in \mathcal{R}_{\#}$.
(2) Whenever $U \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}) \setminus \mathsf{LHS}$, $t'(U) \to t''(U) \in \mathcal{R}_{\#}$.

We now give several simple lemmata (most of them involving the transformation t'). They will be used in Section 11.2.

Lemma 11.1.1. *Let T be an Ω-term. The following equalities hold:*
$\mathcal{P}\mathsf{os}(T) = \mathcal{P}\mathsf{os}(t'(T))$, $\mathcal{P}\mathsf{os}_{\Omega}(T) = \mathcal{P}\mathsf{os}_{\Omega}(t'(T))$, $\mathcal{P}\mathsf{os}_{\overline{\Omega}}(T) = \mathcal{P}\mathsf{os}_{\overline{\Omega}}(t'(T))$.

Lemma 11.1.2. *Let T be an Ω-term. Let $u \in \mathcal{P}\mathsf{os}(T)$. $t'(T[\bullet]_u) = t'(T)[\bullet]_u$.*

Lemma 11.1.3. *Let T and S be two Ω-terms such that $T < S$. $t'(T) < t'(S)$.*

Lemma 11.1.4. *If P is a strict preredex of $\mathcal{R}$ then $t'(P)$ is a strict preredex of $\mathcal{R}_{\#}$.*

Lemma 11.1.5. *If an Ω-term T matches L, then $t'(T)$ matches $t'(L)$.*

Lemma 11.1.6. *Let T be an Ω-term, L be a term. If T matches L and $u \in \mathcal{P}\mathsf{os}(L)$, then $u \in \mathcal{P}\mathsf{os}(T)$ and T/u matches L/u.*

Example 11.1.7. *Take system $\mathcal{R} = \mathcal{R}_4$ of Example A.0.4. We obtain*

$$\mathcal{R}_{\#} = \begin{cases} \mathsf{f}(c_{\mathsf{g}}(\mathsf{a},\mathsf{b},x),\mathsf{a}) & \to & x \\ \mathsf{f}(c_{\mathsf{g}}(\mathsf{a},x,\mathsf{a}),\mathsf{b}) & \to & \mathsf{g}(x,x,x) \\ \mathsf{g}(\mathsf{b},\mathsf{b},\mathsf{b}) & \to & \mathsf{b} \\ \mathsf{g}(\mathsf{a},\mathsf{b},x) & \to & c_{\mathsf{g}}(\mathsf{a},\mathsf{b},x) \\ \mathsf{g}(\mathsf{a},x,\mathsf{a}) & \to & c_{\mathsf{g}}(\mathsf{a},x,\mathsf{a}) \end{cases}$$

We note that $\mathcal{R}_{\#}$ is not orthogonal because it is ambiguous.

Example 11.1.8. *Consider*

$$\mathcal{R} = \begin{cases} \mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{b},x)) & \to & r1 \\ \mathsf{f}(\mathsf{g}(\mathsf{b},x,\mathsf{a})) & \to & r2 \\ \mathsf{h}(\mathsf{g}(x,\mathsf{a},\mathsf{b})) & \to & r3 \\ \mathsf{g}(\mathsf{b},\mathsf{b},\mathsf{b}) & \to & r4 \end{cases}$$

We obtain

$$\mathcal{R}_{\#} = \begin{cases} \mathsf{f}(c_{\mathsf{g}}(\mathsf{a},\mathsf{b},x)) & \to & r1 \\ \mathsf{f}(c_{\mathsf{g}}(\mathsf{b},x,\mathsf{a})) & \to & r2 \\ \mathsf{h}(c_{\mathsf{g}}(x,\mathsf{a},\mathsf{b})) & \to & r3 \\ \mathsf{g}(\mathsf{b},\mathsf{b},\mathsf{b}) & \to & r4 \\ \mathsf{g}(\mathsf{a},\mathsf{b},x) & \to & c_{\mathsf{g}}(\mathsf{a},\mathsf{b},x) \\ \mathsf{g}(\mathsf{b},x,\mathsf{a}) & \to & c_{\mathsf{g}}(\mathsf{b},x,\mathsf{a}) \\ \mathsf{g}(x,\mathsf{a},\mathsf{b}) & \to & c_{\mathsf{g}}(x,\mathsf{a},\mathsf{b}) \end{cases}$$

We note that $\mathcal{R}_{\#}$ is orthogonal but not strongly sequential as $\mathsf{g}(\Omega,\Omega,\Omega)$ does not have an index for $\mathsf{nf_s}$. However $\mathcal{R}$ is strongly sequential.

The following definition expresses the most comprehensive meaning of a function in a system. Let $\mathcal{P}(S)$ denote the *powerset* of S.

Definition 11.1.9. *Let $(\mathcal{R},\mathcal{F})$ be a system. The* meaning function $\mu_{\mathcal{R}}$ *maps each symbol $f \in \mathcal{F}$ to a function $\mu_{\mathcal{R}}(f) : (\mathcal{T}(\mathcal{F}))^k \to \mathcal{P}(\mathcal{T}(\mathcal{F}))$ such that $\mu_{\mathcal{R}}(f)(T_1,\ldots,T_k) = \{U \mid f(T_1,\ldots,T_k) \xrightarrow{*} U \text{ in } \mathcal{R}\ \}$.*

$\mathcal{R}_{\#}$ is expected to deal with terms in $\mathcal{T}(\mathcal{F}_{\#})$ which contains $\mathcal{T}(\mathcal{F})$ as a subset. The map $h : \mathcal{T}(\mathcal{F}_{\#}) \to \mathcal{T}(\mathcal{F})$ is defined as $h(E) = D$ where D is obtained from E by replacing every occurrence of c_f in E by f, for every $f \in \mathcal{F}_{\mathcal{D}}$. Clearly, $h(t'(T)) = h(t''(T)) = T$.

We now give Thatte's theorem. It shows the possibility of simulating an orthogonal system with a constructor system.

Theorem 11.1.10. *[Tha85] For all $f \in \mathcal{F}$, $\mu_{\mathcal{R}}(f) \subseteq h \times \mu_{\mathcal{R}_\#}(f)$ in the sense that, for each $(T1, \ldots, T_k) \in (\mathcal{T}(\mathcal{F}))^k$, $\mu_{\mathcal{R}}(f)(T1, \ldots, T_k) = h(\mu_{\mathcal{R}_\#}(f)(T1, \ldots, T_k))$ where k is the arity of f.*

11.1.2 Constructor-equivalent Systems (CE)

Remark 1: If a system $\mathcal{R}$ is not orthogonal then $\mathcal{R}_\#$ is not orthogonal.
Remark 2: Example 11.1.8 shows that given a strongly sequential system $\mathcal{R}$, $\mathcal{R}_\#$ is not necessarily strongly sequential.

Definition of constructor-equivalent systems (CE)

Thatte's transformation and the previous remarks incite us to define a new subclass of SS.

Definition 11.1.11. *Let $\mathcal{R}$ be an orthogonal system. $\mathcal{R}$ is* constructor-equivalent *if $\mathcal{R}_\#$ is strongly sequential.*

We denote the class of constructor-equivalent systems by CE.

Simple Characterization of CE

We shall prove that the following property is a characterization of constructor-equivalent systems.

Property 11.1.12. $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$, $\forall M \prec N$, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\forall N' \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega)$ *s.t.* $M \prec N'$, $N'/u \neq \Omega$.

We now give a few lemmata used to prove the characterization theorem.

Lemma 11.1.13. *The transformation t' (defined in subsection 11.1.1) is a bijection between $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega(\mathcal{R}))$ and $\mathsf{LHS}_\Omega(\mathcal{R}_\#)$.*

Lemma 11.1.14. *Let T_1 and T_2 be two Ω-terms. $T_1 \preceq T_2$ if and only if $t'(T_1) \preceq t'(T_2)$.*

Lemma 11.1.15. *Let $\mathcal{R}$ be an orthogonal system. The following are equivalent:*
i) $\forall N \in \mathsf{LHS}_\Omega(\mathcal{R}_\#)$, $\forall M \prec N$, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\forall N' \in \mathsf{LHS}_\Omega(\mathcal{R}_\#)$ *s.t.* $M \prec N'$, $N'/u \neq \Omega$
ii) $\forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega(\mathcal{R}))$, $\forall M \prec N$, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\forall N' \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega(\mathcal{R}))$ *s.t.* $M \prec N'$, $N'/u \neq \Omega$.

Proof. From Lemma 11.1.13, t' is a bijection between $\mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_\Omega(\mathcal{R}))$ and $\mathsf{LHS}_\Omega(\mathcal{R}_\#)$. From Lemma 11.1.14, t' preserves the partial order on Ω-terms. From Lemma 11.1.1, t' preserves Ω positions. Then it is clear that *i)* and *ii)* are equivalent. □

Theorem 11.1.16. Characterization Theorem.
An orthogonal system $\mathcal{R}$ is constructor-equivalent if and only if Property 11.1.12 holds.

Proof. Let $\mathcal{R}$ be an orthogonal system. $\mathcal{R} \in \mathsf{CE} \overset{def\ 11.1.11}{\Longleftrightarrow} \mathcal{R}_{\#} \in \mathsf{SS} \cap \mathsf{C}$
$\overset{prop\ 12.2.4}{\Longleftrightarrow} \mathcal{R}_{\#} \in \mathsf{FB} \cap \mathsf{C}$
$\overset{th\ 10.3.19}{\Longleftrightarrow} \forall N \in \mathsf{LHS}_{\Omega}(\mathcal{R}_{\#}),\ \forall M \prec N,\ \exists u \in \mathcal{P}\mathsf{os}_{\Omega}(M),\ \forall N' \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_{\Omega})_{\mathcal{R}_{\#}}$ s.t. $M \prec N'$, $N'/u \neq \Omega$ and $\mathcal{R}_{\#} \in \mathsf{C}$
$\overset{lemma\ 2.3.1}{\Longleftrightarrow} \forall N \in \mathsf{LHS}_{\Omega}(\mathcal{R}_{\#}),\ \forall M \prec N,\ \exists u \in \mathcal{P}\mathsf{os}_{\Omega}(M),\ \forall N' \in \mathsf{LHS}_{\Omega}(\mathcal{R}_{\#})$ s.t. $M \prec N'$, $N'/u \neq \Omega$
$\overset{lemma\ 11.1.15}{\Longleftrightarrow}$ Property 11.1.12 holds. □

Properties of constructor-equivalent Systems

Proposition 11.1.17. $\mathsf{CE} \subset \mathsf{FB}$.

Proof. It follows directly from the characterization of CE and from Theorem 10.3.19 that $\mathsf{CE} \subseteq \mathsf{FB}$. Example A.0.4 is forward-branching but not constructor-equivalent then we get $\mathsf{CE} \neq \mathsf{FB}$. So $\mathsf{CE} \subset \mathsf{FB}$. □

Proposition 11.1.18. $\mathsf{CE} \subset \mathsf{SS}$.

Proof. From Proposition 11.1.17 and Proposition 12.2.1, $\mathsf{CE} \subset \mathsf{FB} \subset \mathsf{SS}$. □

To make our work independent from [DS93], we have written a direct proof of the previous Proposition, (*i.e.* a proof which does not involves the forward-branching class). The proof which is neither short nor trivial; it is given in Section 11.2;

A proposition similar to Proposition 12.2.4 holds for CE:

Proposition 11.1.19. $\mathsf{CE} \cap \mathsf{C} = \mathsf{SS} \cap \mathsf{C}$.

Proof. By Proposition 11.1.18 we have $\mathsf{CE} \cap \mathsf{C} \subseteq \mathsf{SS} \cap \mathsf{C}$.
Let $\mathcal{R} \in \mathsf{SS} \cap \mathsf{C}$. As $\mathcal{R} \in \mathsf{C}$ we have $\mathcal{R}_{\#} = \mathcal{R}$. So, $\mathcal{R}_{\#} \in \mathsf{SS}$ and by definition of CE, $\mathcal{R} \in \mathsf{CE}$. So $SS \cap \mathsf{C} \subseteq \mathsf{CE} \cap \mathsf{C}$. We conclude that $\mathsf{CE} \cap \mathsf{C} = \mathsf{SS} \cap \mathsf{C}$. □

Constructor-equivalent and simple systems

Many people have noticed that Thatte's tranformation preserves strong sequentiality for simple systems (see Section 8.3) [HL91]). We just want to point out with an example that the simple systems form a strict subclass of CE.

Example 11.1.20. *Consider the following system $\mathcal{R} = \mathcal{R}_1$ of Example A.0.1: We have $\mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^* = \{\mathsf{f}(\mathsf{g}(\Omega,\mathsf{a}),\mathsf{a}), \mathsf{f}(\mathsf{g}(\mathsf{a},\Omega),\mathsf{b})\mathsf{g}(\Omega,\mathsf{a}), \mathsf{g}(\mathsf{a},\Omega),\mathsf{a},\mathsf{b}\}$ We know from Section 10.2 that $\mathcal{R}$ is strongly sequential; as it also constructor it is trivially in CE. But according to Huet & Lévy's definition, $\mathcal{R}$ is not a simple system because the set $\{\mathsf{g}(\Omega,\mathsf{a}),\mathsf{g}(\mathsf{a},\Omega)\} \subset \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*$ is not sequential because $M = \mathsf{g}(\Omega,\Omega) \uparrow \mathsf{g}(\Omega,\mathsf{a})$, $M \not\geq \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*$ but $\mathsf{Dir}(M, \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*) = \varnothing$: $\mathcal{P}\mathsf{os}_{\Omega}(M) = \{1,2\}$; $2 \notin \mathsf{Dir}(M, \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*)$ because $\mathsf{g}(\mathsf{a},\Omega)/2 = \Omega$ $1 \notin \mathsf{Dir}(M, \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*)$ because $\mathsf{g}(\Omega,\mathsf{a})/1 = \Omega$.*

11.2 CE ⊂ SS: a direct proof

In Section 11.1.2, we defined the class of constructor equivalent systems (CE) for which Thatte's transformation preserves strong sequentiality. We proved that CE ⊂ SS by showing that CE is a strict subset of the forward-branching class [Str89] which is itself a strict subset of SS. We now give a direct proof (*i.e.* a proof which does not involve the forward-branching class) of the inclusion CE ⊂ SS. It uses parts of the proof given in [KM91] for deciding strong sequentiality. This proof is also published in [DS94].

From now on, we will always deal with two systems: an **orthogonal** system $\mathcal{R}$ and the system $\mathcal{R}^{\#}$ obtained from $\mathcal{R}$ by Thatte's transformation as described in Section 11.1.1. When using notions related to $\mathcal{R}$ we omit the reference to $\mathcal{R}$; when using notions related to $\mathcal{R}^{\#}$, we explicitly specify $\mathcal{R}^{\#}$.

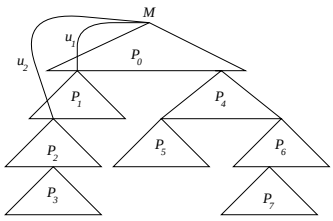


Figure 11.1: Decomposition of an Ω-term M

Definition 11.2.1. *An Ω-term M is* decomposable *if it is built with strict preredexes only.*
A set $D(M) = \{< u_i, P_i > |\ 0 \leq i < n\}$ is a decomposition of M into strict preredexes *if every P_i is a strict preredex and M is the n^{th} term of the sequence defined by*

$$\begin{cases} M_0 = \Omega, \\ M_{i+1} = M_i[P_i]_{u_i}. \end{cases}$$

In this case, we say that M is given by the decomposition $D(M)$.

Figure 11.1 shows a representation of a decomposition of an Ω-term M.

The following two propositions are essential for our proof of Section 11.2.

Proposition 11.2.2. *[KM91] Let $\mathcal{R}$ be an orthogonal system. If $\mathcal{R}$ is not strongly sequential then there exists a minimal free Ω-term M which is built with strict preredexes only.*

Proposition 11.2.3. *[KM91] If M is a minimal free Ω-term then $\mathcal{I}(M[\Omega]_u) = \{u\}$ for all $u \in \mathcal{P}os_{\overline{\Omega}}(M)$.*

Example 11.2.4. *Let $\mathcal{R}$ be the system of Example 11.1.8.*
Let $M = \mathsf{g}(\mathsf{f}(\mathsf{g}(\Omega,\Omega,\Omega)),\Omega,\mathsf{g}(\Omega,\Omega,\Omega))$.
$D_1 = \{< \varepsilon, \mathsf{g}(\Omega,\Omega,\Omega) >, < 1, \mathsf{f}(\mathsf{g}(\Omega,\Omega,\Omega)) >, < 3, \mathsf{g}(\Omega,\Omega,\Omega) >\}$ and
$D_2 = \{< \varepsilon, \mathsf{g}(\Omega,\Omega,\Omega) >, < 1, \mathsf{f}(\Omega) >, < 1.1, \mathsf{g}(\Omega,\Omega,\Omega) >, < 3, \mathsf{g}(\Omega,\Omega,\Omega) >\}$
are two decompositions of M for $\mathcal{R}$.

Lemma 11.2.5. *Let M be a decomposable Ω-term. There exists a unique decomposition of M into strict preredexes $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$ such that no strict subterm of a P_i is a strict preredex.*

Definition 11.2.6. *Let M be a* decomposable *Ω-term. Let $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$ be a decomposition of M into strict preredexes. $D(M)$ is the finest decomposition of M into strict preredexes if no strict subterm of a P_i is a strict preredex. Note that the finest decomposition is well-defined according to the previous lemma.*

Example 11.2.7. *D_2 given in Example 11.2.4 is the finest decomposition of M for $\mathcal{R}$.*

Definition 11.2.8. *Let M be a decomposable Ω-term given by the decomposition $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$. By $M^\#$, we denote the Ω-term of $\mathcal{T}_\Omega(\mathcal{F}^\#)$ given by the decomposition $D(M^\#) = \{< u_i, t'(P_i) >| \ 0 \leq i < n\}$. ($M^\#$ is obtained from M by replacing all strict preredexes P_i by $t'(P_i)$).*

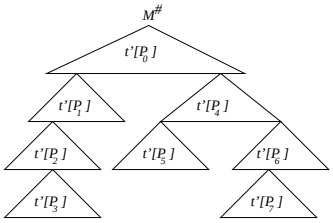


Figure 11.2: Decomposition of the Ω-term $M^\#$

Figure 11.2 shows the Ω-term $M^\#$ for the decomposition of M given in Figure 11.1.

Lemma 11.2.9. *Let M be a decomposable Ω-term given by the decomposition $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$. If $\mathcal{R}^\#$ is not ambiguous and M is in Ω-normal form, then $M^\#$ is in $\mathcal{R}^\#$-Ω-normal form.*

Proof. Suppose that $M^\#$ is not in $\mathcal{R}^\#$-Ω-normal form. Then it contains a $\mathcal{R}^\#$-redex. This redex appears necessarily at an position u_i $(0 \leq i < n)$ because no function symbol appears inside $t'(P_i)$ for all i.

Let $L^\#$ be the redex scheme of $\mathcal{R}^\#$ associated with the left-hand side which matches the $\mathcal{R}^\#$-redex $M^\#/u_i$.

As $\mathcal{R}^\#$ is a constructor system, $L^\#$ does not contain any function symbol except at the root. So, $L^\# \preceq t'(P_i)$. From Lemma 11.1.4, $t'(P_i)$ is a strict preredex of $\mathcal{R}^\#$. We conclude that $\mathcal{R}^\#$ is ambiguous. Contradiction. □

Lemma 11.2.10. *Let M be a decomposable Ω-term given by the decomposition $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$. Then, $\mathcal{P}os_\Omega(M) = \mathcal{P}os_\Omega(M^\#)$ and $\mathcal{P}os_{\overline{\Omega}}(M) = \mathcal{P}os_{\overline{\Omega}}(M^\#)$.*

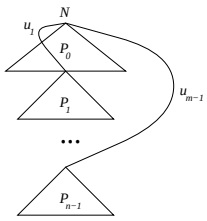


Figure 11.3: A tower of strict preredexes

Definition 11.2.11. *Let N be a decomposable Ω-term. N is a* tower of strict preredexes *if it has a decomposition into strict preredexes $D(N) = \{< u_i, P_i >| 0 \leq i < n\}$ such that $u_j < u_k$ for $j < k$.*

Example 11.2.12 and Figure 11.3 illustrate Definition 11.2.11.

Example 11.2.12. *Let $\mathcal{R}$ be the system of Example 11.1.8.*
Let $N = \mathsf{g}(\mathsf{f}(\mathsf{g}(\Omega, \Omega, \Omega)), \Omega, \Omega)$. N is a tower of preredexes given by the decomposition
$D(N) = \{< \varepsilon, \mathsf{g}(\Omega, \Omega, \Omega) >, < 1, \mathsf{f}(\Omega) >, < 1.1, \mathsf{g}(\Omega, \Omega, \Omega) >\}$

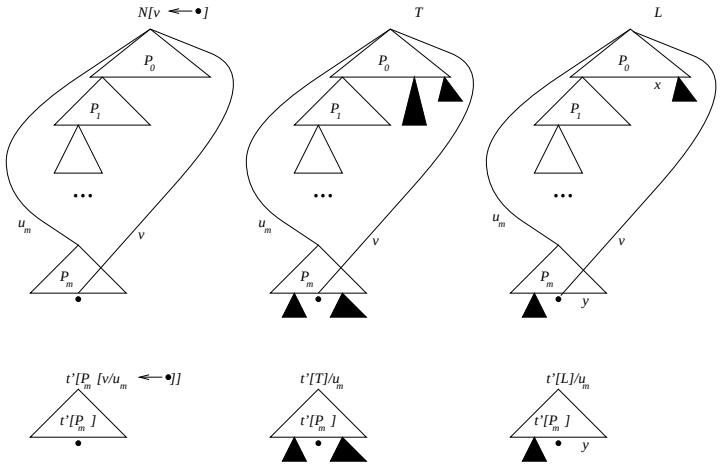


Figure 11.4: Illustration of Lemma 11.2.13

Lemma 11.2.13. *Let N be a tower of preredexes given by the decomposition $D(N) = \{< u_i, P_i >| 0 \leq i < m + 1\}$. Let $v \in \mathcal{P}\mathsf{os}_\Omega(N)$ such that $v > u_m$.*
Let T be an Ω-term and L a left-hand side such that $N[\bullet]_v < T$, T matches L and $u_m \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(L_\Omega)$, then, $t'(P_m[\bullet]_{v/u_m})$ is $\mathcal{R}^\#$-redex compatible.

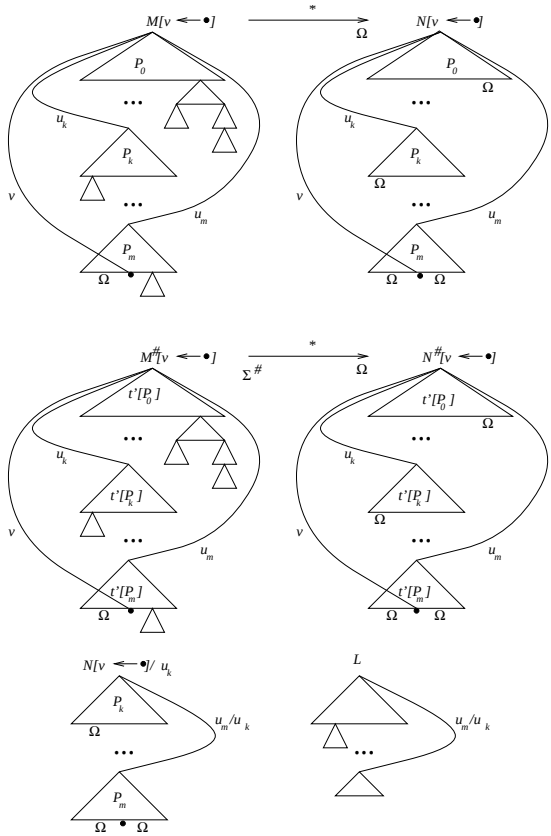


Figure 11.5: Illustration of Lemma 11.2.14

Lemma 11.2.13 is illustrated by Figure 11.4.

Proof. As T matches L and $u_m \in \mathcal{P}\mathsf{os}(L)$, using Lemma 11.1.6, we get T/u_m matches L/u_m. Now using Lemma 11.1.5, we get $t'(T/u_m]$ matches $t'[L/u_m)$. (1)
From $T > N[\bullet]_v$, we get $T/u_m > N[\bullet]_v/u_m$. (2)
$N[\bullet]_v/u_m = P_m[\bullet]_{v/u_m}$. (3)
From (2) and (3), we get $T/u_m > P_m[\bullet]_{v/u_m}$. (4)
From (4) and Lemma 11.1.3, we get $t'(T/u_m] > t'[P_m[\bullet]_{v/u_m})$. (5)
As $root(L_\Omega/u_m) \neq \Omega$, $root(L/u_m)$ is a function symbol so by Thatte's transformation
$t'(L/u_m)$ is a left-hand side of $\mathcal{R}^\#$. (6)
From (1), (5) and (6), $t'(P_m[\bullet]_{v/u_m})$ is $\mathcal{R}^\#$-redex compatible. □

Lemma 11.2.14. *Let M be a minimal free Ω-term given by its* **finest** *decomposition $D(M) = \{< u_i, P_i >| \ 0 \leq i < n\}$. Then, $\mathcal{I}_{\mathcal{R}^\#}(M^\#) = \varnothing$.*

Lemma 11.2.14 is illustrated by Figure 11.5.

Proof. Let $v \in \mathcal{P}\mathsf{os}_\Omega(M^\#)$. From Lemma 11.2.10, $v \in \mathcal{P}\mathsf{os}_\Omega(M)$. As M is a free Ω-term, we have $v \notin \mathcal{I}(M)$ and by Lemma 8.1.17, $v \notin \mathcal{P}\mathsf{os}(\omega(M[\bullet]_v))$.

In an Ω-reduction from $M[\bullet]_v$ to its Ω-normal form $\omega(M[\bullet]_v)$ reductions occur at $u_i s$ only (because we consider the finest decomposition). As every P_i is a preredex, every P_i such that $u_i \not< v$ Ω-reduces to Ω.
So $M[\bullet]_v \xrightarrow{*}_\Omega N[\bullet]_v$ with N being the tower given by the decomposition: $D(N) = \{< u_j, P_j > \in D(M) \mid u_j < v\}$.
In the same way, as from Lemma 11.1.4 each $t'(P_i)$ is a preredex of $\mathcal{R}^\#$, we have $M^\#[\bullet]_v \xrightarrow{*}_{\Omega,\mathcal{R}^\#} N^\#[\bullet]_v$ with $N^\#$ being the tower given by the decomposition: $D(N^\#) = \{< u_j, t'(P_j) > \in D(M^\#) \mid u_j < v\}$.
By confluence of Ω-reduction, $N[\bullet]_v \xrightarrow{*}_\Omega \omega(M[\bullet]_v)$.

But as $v \notin \mathcal{P}\mathsf{os}(\omega(M[\bullet]_v))$, this Ω-reduction contains at least one step. So, $\exists k$ such that $< u_k, P_k > \in D(N)$ and $N[\bullet]_v/u_k$ is redex compatible. This means that $N[\bullet]_v/u_k$ can be refined to a redex R. Let us consider L the left-hand side matching R and L_Ω the corresponding redex scheme. Let us consider m *s.t.* $< u_m, P_m > \in D(N)$ and *s.t.* $\forall j \neq m$ *s.t.* $< u_j, P_j > \in D(N)$, $u_j < u_m$.

FACT: $u_m/u_k \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(L_\Omega)$.

Proof. Suppose that $u_m/u_k \notin \mathcal{P}\mathsf{os}_{\overline{\Omega}}(L_\Omega)$. Then $N[\bullet]_{u_m}/u_k$ is redex compatible.
So, $\omega(N[\bullet]_{u_m}) = \omega(N[\Omega]_{u_k})$.
Now, $\omega(M[\bullet]_{u_m}) = \omega(N[\bullet]_{u_m}) = \omega(N[\Omega]_{u_k})$.
We have also $\omega(M[\Omega]_{u_m}) = \omega(N[\Omega]_{u_m}) = \omega(N[\Omega]_{u_k})$.
So we have $\omega(N[\bullet]_{u_m}) = \omega(M[\Omega]_{u_m})$ which means by Lemma 8.1.17 that $u_m \notin \mathcal{I}(M[\Omega]_{u_m})$. This contradicts Proposition 11.2.3 which says that if M is a minimal free term $\mathcal{I}(M[\Omega]_{u_m}) = \{u_m\}$. □

The hypothesis of Lemma 11.2.13 are satisfied so $t'(P_m[\bullet]_{v/u_m})$ is $\mathcal{R}^\#$-redex compatible.
But $t'(P_m[\bullet]_{v/u_m}) = t'(P_m)[\bullet]_{v/u_m} = N^\#[\bullet]_v/u_m$.
So, $N^\#[\bullet]_v/u_m$ is $\mathcal{R}^\#$-redex compatible and we have the following step of Ω-reduction: $N^\#[\bullet]_v \longrightarrow_{\Omega,\mathcal{R}^\#} N^\#[\Omega]_{u_m}$.
By confluence of Ω-reduction, $N^\#[\bullet]_v \xrightarrow{*}_{\Omega,\mathcal{R}^\#} \omega_{\mathcal{R}^\#}(M^\#[\bullet]_v)$. As $v \notin \mathcal{P}\mathsf{os}(N^\#[\Omega]_{u_m})$, we have also $v \notin \mathcal{P}\mathsf{os}(\omega_{\mathcal{R}^\#}(M^\#[\bullet]_v))$. So by Lemma 8.1.17, $v \notin \mathcal{I}_{\mathcal{R}^\#}(M^\#)$. □

Proposition 11.2.15. *Let $\mathcal{R}$ be an orthogonal system. $\mathcal{R}^\# \in \mathsf{SS} \Rightarrow \mathcal{R} \in \mathsf{SS}$.*

Proof. Suppose that $\mathcal{R} \notin \mathsf{SS}$. Then, from Proposition 11.2.2, there exists a minimal free Ω-term $M \in \mathcal{T}_\Omega(\mathcal{R})$ which is built with strict preredexes only. Let $D(M) = \{< u_i, P_i > \mid 0 \leq i < n\}$ be the finest decomposition of M into preredexes.
As $\mathcal{R}^\# \in \mathsf{SS}$, it is not ambiguous. From Lemma 11.2.9, $M^\#$ is in $\mathcal{R}^\#$-Ω-normal form. (1).
From Lemma 11.2.14, $\mathcal{I}_{\mathcal{R}^\#}(M^\#) = \varnothing$. (2).
From (1), (2) and the definition of SS, we conclude that $\mathcal{R}^\# \notin \mathsf{SS}$. Contradiction. □

Corollary 11.2.16. $\mathsf{CE} \subset \mathsf{SS}$.

11.3 Conclusion

Owing to Thatte's transformation, a constructor-equivalent system can be simulated by a strongly sequential constructor system. As constructor systems are easier to handle this last restriction ($\mathsf{C} \cap \mathsf{SS}$) has often been considered [BMS81]. However, the transformation doesn't preserve strong sequentiality in general. This has led us to define the class of constructor-equivalent systems for which strong sequentiality is preserved. We have given a simple characterization for this class.

At that point of the work, arose the question "is there a bigger class than CE whose systems can be transformed into an equivalent constructor strongly sequential system"?

The proposition "$\mathsf{CE} \subset \mathsf{FB}$" showed that considering forward-branching systems was even less restrictive and enhanced the interest of working on this class. Section 12.1 showed that this choice was right as in the end a transformation from FB to $\mathsf{C} \cap \mathsf{SS}$ was found.

Chapter 12

Back to forward-branching systems

12.1 Transformation from FB to SS ∩ C

We will describe here an algorithm to transform any forward-branching system into an equivalent constructor forward-branching system (so a constructor strongly sequential system according to Lemma 12.2.4). The correctness proofs and examples can be found in [SS96, SS97].

Although, we are not one of the authors of the published version of the algorithm [SS96, SS97], we have participated in its design as co-supervisor of Bruno Salinier's PHD thesis [Sal95]. To finalize that work, we have implemented the two versions of the algorithm in `Autowrite`.

The tranformation is based upon three procedures. The first one is the forward-branching-index-tree procedure described in Section 10.4.3.

The second shown in Figure 12.1 is the find-differentiating-term procedure which finds a differentiating term using the forward-branching index-tree built by the forward-branching-index-tree function.

The third procedure shown in Figure 12.2 is the transform-forward-branching procedure which recursively performs one step of the transformation using a differentiating term. The recursion stops when the system is constructor.

In the worst case the number of added rules is equal to the number of strict non-variable subterms in the left-hand sides. This we think cannot be improved. However, the size of rules can be improved. This first version of the transform-forward-branching algorithm duplicates a lot of subterms.

This is clearly shown by the following family of examples $(\mathcal{R}_n)_{n\geq 2}$ due to Rebelsky [Reb93]. These systems are based upon the signature $\mathcal{F} = \{\mathsf{a}^{(0)}, \mathsf{f}^{(1)}\}$. Each system $\mathcal{R}_n$ consists of a single rule:

$$\mathsf{f}(\mathsf{f}(\ldots\mathsf{f}(\mathsf{a}))) \quad \to \mathsf{a}$$

function find-differentiating-term(s_0:index-point):term;
begin
$s := s_0$; $M := \Omega$; $u := s.index$;
$f := \mathsf{root}(N)$ with N a constructor subscheme;
while $\delta(s, f)$ is defined **do**
begin
$s := \delta(s, f)$;
$M := M[f(\vec{\Omega})]_u$; $u := s.index$;
$f := \mathsf{root}(M/u)$;
end
return $M[f(\vec{\Omega})]_u$;
end

Figure 12.1: The find-differentiating-term algorithm

with n occurrences of f. For instance, for $n = 5$, we obtain

$$\mathcal{R}_5 = \{\ \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))))) \quad \to \mathsf{a}\ \}$$

The system produced by the first version of transform-forward-branching is

$$\begin{array}{rcl}
\mathsf{f}(\mathsf{a}) & \to & \mathsf{c}^1_{\mathsf{f}}(a) \\
\mathsf{f}(\mathsf{c}^1_{\mathsf{f}}(x)) & \to & \mathsf{c}^2_{\mathsf{f}}(\mathsf{c}^1_{\mathsf{f}}(x)) \\
\mathsf{f}(\mathsf{c}^2_{\mathsf{f}}(x)) & \to & \mathsf{c}^3_{\mathsf{f}}(\mathsf{c}^2_{\mathsf{f}}(x)) \\
\mathsf{f}(\mathsf{c}^3_{\mathsf{f}}(x)) & \to & \mathsf{c}^{\mathsf{f}}_4(\mathsf{c}^3_{\mathsf{f}}(x)) \\
\mathsf{f}(\mathsf{c}^{\mathsf{f}}_4(\mathsf{c}^3_{\mathsf{f}}(\mathsf{c}^2_{\mathsf{f}}(\mathsf{c}^1_{\mathsf{f}}(\mathsf{a}))))) & \to & \mathsf{a}
\end{array}$$

In the worst case, the size of the resulting system is quadratic with regard to the size of the initial system. The transform-forward-branching algorithm can be improved in order that the size of the resulting system increases only linearly with respect to the size of the input system. The main idea is that we do not need to know the details of the differenciating term $f(T_1, \ldots, T_k)$; we just need to know about its Ω-positions. So the symbols at non-Ω-positions are irrelevant and do not need to appear in the constructed rules.

The improved version of the algorithm is given Figure 12.3.

The system produced by the first version of transform-forward-branching is

$$\begin{array}{rcl}
\mathsf{f}(\mathsf{c}^3_{\mathsf{f}}) & \to & \mathsf{c}^{\mathsf{f}}_4 \\
\mathsf{f}(\mathsf{c}^2_{\mathsf{f}}) & \to & \mathsf{c}^3_{\mathsf{f}} \\
\mathsf{f}(\mathsf{c}^1_{\mathsf{f}}) & \to & \mathsf{c}^2_{\mathsf{f}} \\
\mathsf{f}(\mathsf{a}) & \to & \mathsf{c}^1_{\mathsf{f}} \\
\mathsf{f}(\mathsf{c}^{\mathsf{f}}_4) & \to & \mathsf{a}
\end{array}$$

We consider that this algorithm is a very interesting contribution to the domain: the forward-branching class is the biggest easily specified class defined so

```
procedure transform-forward-branching(R):system;
begin
s_0 := forward-branching-index-tree(LHS_Ω);
Let Φ := {s | φ^{-1}(s) ≠ ∅};
if Φ = {s_0} then
    return R; /* the system is already constructor */
f(T_1, ..., T_k) := find-differentiating-term(s_0);
Let c^f be a new symbol of arity k;
for each L ∈ LHS do
    for each u ∈ Pos_Ω̄(L) such that f(T_1, ..., T_k) ≤ L/u do
        begin
        L/u := f(L_1, ..., L_k);
        L := L[c^f(L_1, ..., L_k)]_u;
        end
R := R ∪ {f(T_1, ..., T_k) → c^f(T_1, ..., T_k)};
return transform-forward-branching(R);
end
```

Figure 12.2: The transform-forward-branching algorithm

far in which systems transform to constructor while preserving strong sequentiality. In addition the size of the system increases only linearly with regard to the original system.

The forward-branching class is the biggest defined subclass of the strongly sequential class for which a polynomial rewrite strategy exists.

12.2 Relations between FB and subclasses of SS

12.2.1 Comparison with strongly sequential systems

Proposition 12.2.1. $\mathsf{FB} \subset \mathsf{SS}$.

Proof. This follows from the definition of FB in terms of the existence of a forward-branching index tree. In Section 10.2, we have shown that every system admitting an index tree is strongly sequential. As a forward index tree is a special case of an index tree, the inclusion follows. □

Lemma 12.2.2. $\mathsf{FB} \cap \mathsf{C} \subseteq \mathsf{SS} \cap \mathsf{C}$.

Proof. #1 This lemma follows directly from Proposition 12.2.1. □

To make our work independent from index trees, we give a second proof that uses Proposition 8.2.2.

procedure transform-forward-branching($\mathcal{R}$):system;
begin
s_0 := forward-branching-index-tree(LHS_Ω);
Let $\Phi := \{s \mid \phi^{-1}(s) \neq \varnothing\}$;
if $\Phi = \{s_0\}$ **then**
    return $\mathcal{R}$; /* the system is already constructor */
$f(T_1, \ldots, T_k)$:= find-differentiating-term(s_0);
$\{v_1, \ldots, v_k\} := \mathcal{P}\mathsf{os}_\Omega(f(T_1, \ldots, T_k))$;
Let c^f be a new symbol of arity k;
for each $L \in \mathsf{LHS}$ **do**
    for each $u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(L)$ such that $f(T_1, \ldots, T_k) \leq L/u$ **do**
        begin
        let $L/u := f(L_1, \ldots, L_k)[L_1, \ldots, L_k]_{v_1, \ldots v_k}$;
        $L := L[c^f(L_1, \ldots, L_k)]_u$;
        end
$\mathcal{R} := \mathcal{R} \cup \{f(T_1, \ldots, T_k) \to c^f(\vec{\Omega})\}$;
return transform-forward-branching($\mathcal{R}$);
end

Figure 12.3: The transform-forward-branching algorithm with symbol elimination

Proof. #2 Let $\mathcal{R} \in \mathsf{FB} \cap \mathsf{C}$. As $\mathcal{R} \in \mathsf{C}$ and from Lemma 2.3.1, we get $\mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega) = \mathsf{LHS}_\Omega$. Then, according to Theorem 10.3.19, $\forall M$ strict pre-redex, $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M)$ s.t. $\forall N \in \mathsf{LHS}_\Omega$, $M \prec N$, $N/u \neq \Omega$. As $N \in \mathsf{LHS}_\Omega$, $\mathsf{nf_s}(N)$ holds then u is an index of M. From Proposition 8.2.2, it follows that $\mathcal{R} \in \mathsf{SS}$. □

Lemma 12.2.3. $\mathsf{SS} \cap \mathsf{C} \subseteq \mathsf{FB} \cap \mathsf{C}$.

Proof. Let $\mathcal{R} \in \mathsf{SS} \cap \mathsf{C}$. Suppose that $\mathcal{R} \notin \mathsf{FB}$. Then,
$\exists N \in \mathsf{LHS}_\Omega$, $\exists M \prec N$, $\forall u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\exists N' \in \mathcal{S}\mathsf{ub}_\mathcal{D}(\mathsf{LHS}_\Omega)$ s.t. $M \prec N'$ and $N'/u = \Omega$ (1)
As $\mathcal{R} \in \mathsf{C}$, then (1) becomes by lemma 2.3.1 :
$\exists N \in \mathsf{LHS}_\Omega$, $\exists M \prec N$, $\forall u \in \mathcal{P}\mathsf{os}_\Omega(M)$, $\exists N' \in \mathsf{LHS}_\Omega$ s.t. $M \prec N'$ and $N'/u = \Omega$ (2)

We now show that $\mathcal{I}(M) = \varnothing$. (3)
Let $u \in \mathcal{P}\mathsf{os}_\Omega(M)$. From (2), there exists $N' \in \mathsf{LHS}_\Omega$ s.t. $M \prec N'$ and $N'/u = \Omega$. As $N' \in \mathsf{LHS}_\Omega$, N' arbitrarily reduces in one step to any normal form so $\mathsf{nf_s}(N')$ holds. It follows that $u \notin \mathcal{I}(M)$.

By nonambiguity of our systems and as $M \prec N \in \mathsf{LHS}_\Omega$, we know that M is in Ω-normal form. (4)

From (3), (4) and the definition of SS, $\mathcal{R} \notin \mathsf{SS}$, contradiction. □

The next proposition similar to Proposition 11.1.19 holds for FB. It is interesting in itself and also used in Section 11.1 for the characterization of CE.

Proposition 12.2.4. $\mathsf{FB} \cap \mathsf{C} = \mathsf{SS} \cap \mathsf{C}$.

Proof. From Lemma 12.2.2 and Lemma 12.2.3 □

It's interesting to note that a similar result holds for Thatte's *left-sequential* class LS which is a superset of SS: $\mathsf{LS} \cap \mathsf{C} = \mathsf{SS} \cap \mathsf{C}$ as proved by Thatte in [Tha87].

The forward-branching class is a strict subset of the strongly sequential class and contains systems which are nonconstructor systems. This is also true for simple systems (SP) [HL91], constructor equivalent systems (CE) [DS93] and Transitive systems (TR) [TSvEP93b]. In this section, we examine how these classes compare with FB.

Lemma 12.2.5. $\mathsf{LHS}_\Omega^{\prec} \cap \mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^* = \varnothing$.

Proof. This is an easy consequence of the nonambiguity hypothesis. □

Note that $\mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^*$ is denoted by Red_Ω^* in [HL91].

12.2.2 Comparison with Simple systems

In Chapter 11(and [DS93]), we introduced the class of *constructor equivalent systems* (CE) and we showed that $\mathsf{SP} \subset \mathsf{CE} \subset \mathsf{FB}$. So, clearly SP is a strict subclass of FB. The algorithm that we have presented in this article for FB happens to be identical to the one given by Huet and Lévy for SP. This shows that the definition of simple systems was just unnecessarily restrictive. Here is an example of a system which is forward-branching but not simple:

Example 12.2.6. *Let* $\mathsf{LHS}_\Omega = \{\mathsf{f}(\mathsf{g}(\mathsf{a},\Omega,\mathsf{a})), \mathsf{h}(\mathsf{g}(\Omega,\mathsf{a},\mathsf{a})), \mathsf{g}(\mathsf{b},\mathsf{b},\mathsf{b})\}$.
We can easily check that the system is forward-branching. According to Huet and Lévy's definition, this system is not simple because $\{\mathsf{g}(\mathsf{a},\Omega,\mathsf{a}), \mathsf{g}(\Omega,\mathsf{a},\mathsf{a})\} \subset \mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^*$ *is not a* sequential *set.*

12.2.3 Comparison with Transitive systems

The simple characterization that we have given for FB helps us verify that the class of Transitive systems (TR) defined by [TSvEP93b] is the same as the class FB first defined in [Str88]. We define two properties on $\mathsf{LHS}_\Omega^{\prec}$ and show that they are equivalent.

Definition 12.2.7. *Let* $M \in \mathsf{LHS}_\Omega^{\prec}$.
P1(M) holds if $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M), \forall N \in \mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^*$ *with* $M \prec N, u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(N)$.
P2(M) holds if $\exists u \in \mathcal{P}\mathsf{os}_\Omega(M), \forall N \in \mathcal{S}\mathsf{ub}(\mathsf{LHS}_\Omega)^*$ *with* $M \uparrow N, u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(N)$.

Lemma 12.2.8. *Let* $M \in \mathsf{LHS}_\Omega^{\prec}$. $P2(M)$ *implies* $P1(M)$.

Proof. Obvious as $M \prec N$ implies $M \uparrow N$. □

Lemma 12.2.9. *If* $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$ *then* $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P2(M)$.

Proof. Suppose that $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$ holds and that there exists a strict preredex M such that $P2(M)$ does not hold. So for every $u \in \mathcal{P}\mathsf{os}_{\Omega}(M)$ there exists a term $N_u \in \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^*$ such that $M \uparrow N_u$ and $u \notin \mathcal{P}\mathsf{os}_{\overline{\Omega}}(M)$. Let $\mathcal{P}\mathsf{os}_{\Omega}(M) = \{u_1, \ldots u_n\}$. Consider the strict preredex M' defined as the greatest lower bound (w.r.t prefix ordering on Ω-terms) of $\{N_{u_1}, \ldots, N_{u_n}\}$. We claim that $P1(M')$ does not hold. Let $u \in \mathcal{P}\mathsf{os}_{\Omega}(M')$. Either (1) $u \in \mathcal{P}\mathsf{os}_{\Omega}(M)$ or (2) $u \in \mathcal{P}\mathsf{os}_{\Omega}(N_v)$ for some $v \in \mathcal{P}\mathsf{os}_{\Omega}(M)$. In case (1) we obtain $M' \prec N_v$ ($M' \neq N_v$ by Lemma 12.2.5) and $u \in \mathcal{P}\mathsf{os}_{\Omega}(N_u)$. In case (2) we have $M' \prec N_v$ with $u \in \mathcal{P}\mathsf{os}_{\Omega}(N_v)$. So $P1(M')$ cannot hold, contradicting our assumption. □

Corollary 12.2.10. $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$ *iff* $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P2(M)$.

Proof. From Lemma 12.2.8 and Lemma 12.2.9. □

Lemma 12.2.11. *Let* $M \in \mathsf{LHS}^{\prec}_{\Omega}$.
$\{N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_{\Omega}) \mid M \prec N\} = \{N \in \mathcal{S}\mathsf{ub}(\mathsf{LHS}_{\Omega})^* \mid M \prec N\}$.

Proof. Obvious as a strict preredex M cannot be prefix of a term N if N has a constructor symbol at the root. □

Lemma 12.2.12. *Property 10.3.4 holds iff* $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$.

Proof. By definition Property 10.3.4 holds iff
$\forall M \in \mathsf{LHS}^{\prec}_{\Omega}, \exists u \in \mathcal{P}\mathsf{os}_{\Omega}(M), \forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_{\Omega})$ with $M \prec N, N/u \neq \Omega$ which is equivalent to
$\forall M \in \mathsf{LHS}^{\prec}_{\Omega}, \exists u \in \mathcal{P}\mathsf{os}_{\Omega}(M), \forall N \in \mathcal{S}\mathsf{ub}_{\mathcal{D}}(\mathsf{LHS}_{\Omega})$ with $M \prec N, u \in \mathcal{P}\mathsf{os}_{\overline{\Omega}}(N)$.
Using Lemma 12.2.11, this is equivalent to $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$. □

The following lemma characterizes the Transitive systems:

Lemma 12.2.13. *[TSvEP93b]*
A system is transitive iff $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P2(M)$ *holds.*

Proposition 12.2.14. $\mathsf{TR} = \mathsf{FB}$.

Proof. Consider an orthogonal system. From Lemma 12.2.13, the system is transitive iff $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P2(M)$. From Corollary 12.2.10, this is equivalent to $\forall M \in \mathsf{LHS}^{\prec}_{\Omega}$, $P1(M)$. From Lemma 12.2.12, this is equivalent to Property 10.3.4. From Theorem 10.3.19, this is equivalent to say that the system is forward-branching. □

12.3 Comparison between subclasses of SS

Figure 12.4 shows the inclusion relations between all the subclasses of SS discussed earlier.

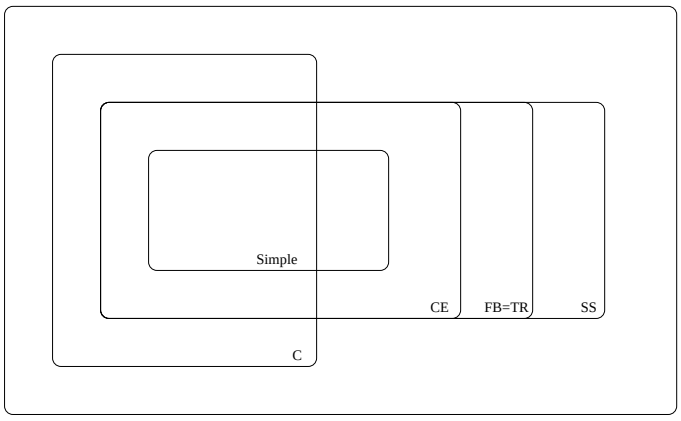


Figure 12.4: Inclusion relations between subclasses of SS

Chapter 13

Complexity of SS

Not much is published about the complexity of the problem of deciding membership in one of the classes that guarantees a computable call-by-need strategy to normal form. The class of forward-branching systems (Strandh [Str89]), a proper subclass of the class of orthogonal strongly sequential systems, coincides with the class of transitive systems (Toyama *et al.* [TSvEP93a]) and can be decided in quadratic time (see Section 10.4, [Dur94b]). For classes higher in the hierarchy the known upper bounds range from double to triple exponential (see Chapter 5, [DM98]). Comon [Com00] showed that strong sequentiality of a left-linear rewrite system can be decided in exponential time. In [KM91], Klop and Middeldorp conjectured that the same problem but for orthogonal systems is NP-complete. This is still an open problem (listed as number 9, in the RTA list of problems in rewriting).

Our interest in this conjecture was one of the reasons we first invited Aart Middeldorp as an invited professor in Bordeaux in 1995. That visit ended up in a very fruitful collaboration in the domain of call-by-need classes! But no result was obtained concerning the conjecture.

We have been working on this conjecture since 1992, full or part-time but without reaching the ultimate goal after 13 years.

We have our own sub-conjectures which are that the problem is both in NP and co-NP which would reduce its chances to be NP-complete. But they are still nothing more than conjectures.

Conjecture 13.0.1. *The problem of deciding whether an orthogonal system is in* SS *is in* NP*.*

Conjecture 13.0.2. *The problem of deciding whether an orthogonal system is in* SS *is in co-*NP*.*

We have often felt really close to showing either one of theses conjectures but was always stopped at some critical point. After working for so many years on these conjectures, we feel that we know the problem quite well but that there are intrinsic difficulties that we have not been able to overcome. The total time

spent in the search of a solution has become so significant that it does not seem reasonnable to keep on working on the problem.

We shall here report some intermediate work and notions that could be useful for people willing to attack these problems. Please note that both sections 13.1 and 13.2 contain **unfinished** work.

Section 13.2 gives interesting examples of systems having index-trees of exponential size which cannot be trivially transformed into index-trees of polynomial size (every non-deterministic index point being the target of a failure-transition). To our knowledge, such examples have never been published.

13.1 Work on the co-NP-conjecture (13.0.2)

We would like first to refer to Klop and Middeldorp's decidability proof for strong sequentiality which is based on the search of a minimal free term [KM91]. By definition, the problem of deciding whether a system is not strongly sequential is the same as the problem of the existence of a free term (see p37).

In [KM91], they show examples of systems with very deep minimal free terms; but although they are very deep, their size is linear. So, our first hope was to show that if there exists a free term then there exists at least one of polynomial size. Given a term of polynomial size, it is possible to check in polynomial time that the term is free by applying the index decision procedure to each one of its Ω-positions. Then we would have concluded that the problem is in co-NP.

In fact, we hope not to prove that a minimal free term has necessarily a polynomial size. However we would like to prove that if the system is not strongly sequential there exists a free term which can be folded into a DAG of polynomial size. In addition, we would like that the DAG satisfies certain conditions which allow checking whether the corresponding term is free without unfolding the DAG, so in polynomial time.

13.2 Work on the NP-conjecture (13.0.1)

A system is strongly sequential if and only if it admits an index-tree (see Section 10.2). We are convinced that every strongly sequential system admits an index-tree with a polynomial number of index points (even that a minimal index-tree has no more branches than its number of defined subterms (proper or not)). Given a set containing a polynomial number of index points, it is possible to check in polynomial time whether this set corresponds to an index-tree (check that induced tranfer and failure transitions are correct). Then we would conclude that the problem is in NP.

Given a strongly sequential system $\mathcal{R}$, we assimilate an index-tree $\mathcal{S}$ for $\mathcal{R}$ with its set of index points (the non-final states) (from which one can easily derive the transfer and failure fonctions), the final states being always $\mathsf{LHS}_\Omega(\mathcal{R})$.

Definition 13.2.1. *Let $s \in \mathcal{S}$. s is* non-deterministic *if $\exists t \in \mathcal{S}$ and a symbol f such that $s \in \delta(t, f)$ and $s' \in \mathcal{S}$ such that $s' \neq s$ and $s' \in \delta(t, f)$. By $\mathsf{ND}(\mathcal{S})$ we denote the set of non-deterministic index points of $\mathcal{S}$.*

Example. *In the index-tree of Figure 13.1, $[\mathsf{g}(\Omega, \Omega), 2]$ and $[\mathsf{g}(\Omega, \Omega), 1]$ are non-deterministic index points.*

In a minimal index-tree every non-deterministic index point s must also be a failure point (reached by a failure transition). Otherwise we could obtain another index-tree by removing s and all index points that become unaccessible after removing s.

Definition 13.2.2. *An index-tree such that every non-deterministic index point is also a failure point is called* strict.

For instance, for $\mathcal{R} = \mathcal{R}_3$ of Example A.0.3, the index-tree presented in Figure 13.1 can be simplified into the forward-branching and minimal one presented in Figure 10.3 (p102).

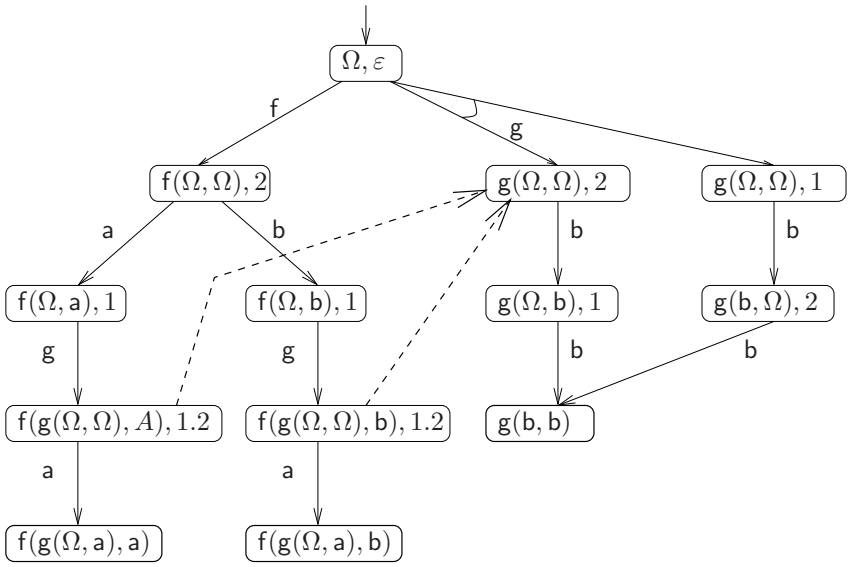


Figure 13.1: A non minimal index-tree for $\mathsf{LHS}_\Omega(\mathcal{R}_3)$

To give an idea of the difficulty of the problem, we now show with examples that even strict index-trees may be of exponential size. However for all these examples a polynomial index-tree can be found which supports our conjecture.

We will construct a family of sets of redex schemes Π_n over the signature $\{\mathsf{a}_0, \mathsf{f}_n\}$.

First for $i, j \in [0, n[$, we denote by $o(i, j)$ the position $k_1 \dots k_j$ where each $k_l = ((i+l) mod\ n)+1$. Consider the family of set of redex schemes Π_n over the

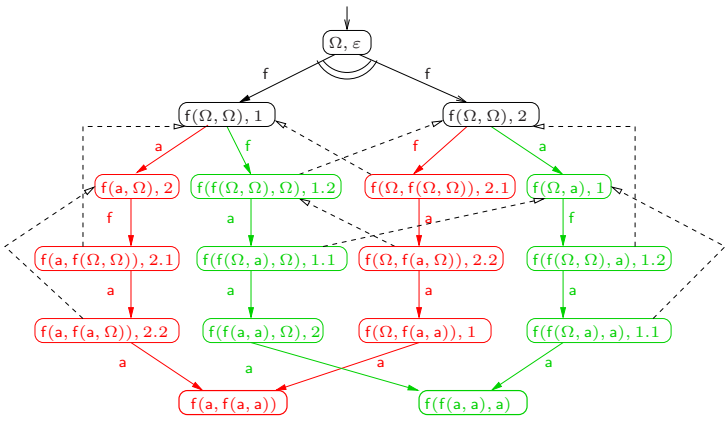


Figure 13.2: An index-tree for Π_2

signature $\{\mathsf{a}_0, \mathsf{f}_n\}$ defined by $\Pi_n = \{\pi_0, \ldots, \pi_{n-1}\}$ where each π_j is such that

$$\mathcal{P}_{\mathsf{f}}(\pi_i) = \{o(i,j) \mid j \in [0,n[\}$$
$$\mathcal{P}_{\mathsf{a}}(\pi_j) = \mathcal{P}(\pi_j) \setminus \mathcal{P}_{\mathsf{f}}(\pi_j)$$

For instance, $\Pi_2 = \{\pi_1, \pi_2\}$ with

$$\pi_0 = \mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{a}, \mathsf{a}))$$
$$\pi_1 = \mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{a}), \mathsf{a})$$

and $\Pi_3 = \{\pi_1, \pi_2, \pi_3\}$ with

$$\pi_0 = \mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{a})), \mathsf{a})$$
$$\pi_1 = \mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{a}), \mathsf{a}, \mathsf{a}))$$
$$\pi_2 = \mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{a}, \mathsf{a}, \mathsf{a}), \mathsf{a}), \mathsf{a}, \mathsf{a})$$

Clearly $|\Pi_n| = n(n^2+1)$.

Figure 13.2 gives a strict index-tree for the set of redex schemes Π_1.

However it is not minimal. Figure 13.3 gives a minimal index-tree for Π_1.

In the index tree of Figure 13.2, non-deterministic index points appear at level 1 roughly multiplying the number of branches of the index-tree by 2.

For Π_3 an index tree might start as in Figure 13.4 (to save space we do not represent the implicit prefixes inside the index points). However the minimal index tree shown in Figure 13.5 has a linear number of index points.

More generally we can show that for Π_n we can build a strict index-tree with $n^{n-1} + n(n-1)^{n-1}$ branches which is exponential with regard to the size of Π_n. However, for every n, a minimal index tree with n branches exists.

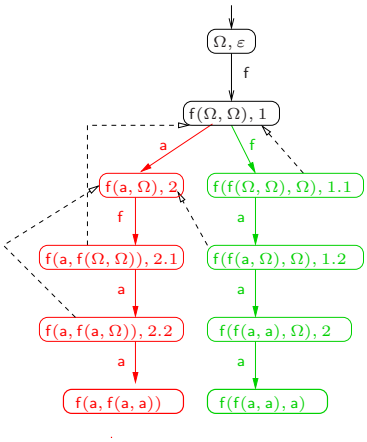


Figure 13.3: A minimal index-tree for Π_2

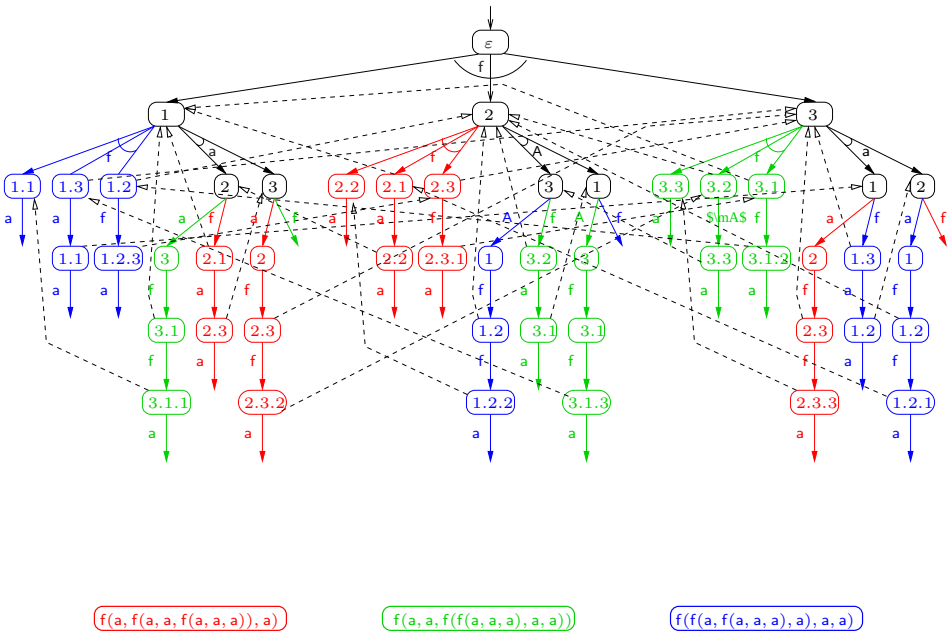


Figure 13.4: An exponential index-tree for Π_3

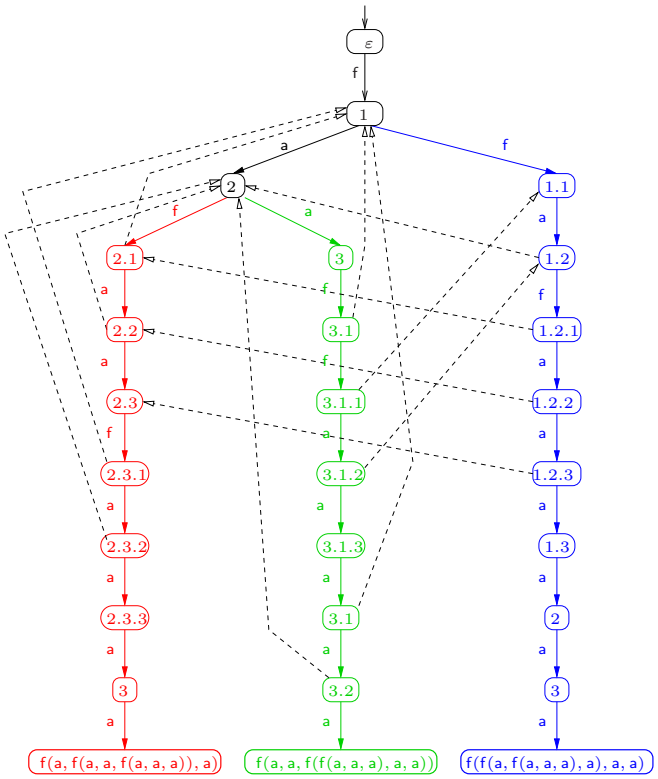


Figure 13.5: A minimal index-tree for Π_3

Chapter 14

Compilation of Call-by-need Strategies

In Part I, we have given a framework for the study of call-by-need computations.

What we did not address in that part is the important issue of compiling call-by-need strategies. The knowledge that every reducible term has at least one computable needed redex is clearly insufficient to obtain an efficient call-by-need strategy. Testing the redexes in a reducible term one by one until a needed redex is encountered is unattractive.

The theory of strongly sequential systems from Huet and Lévy solves the problem of finding a needed redex in time polynomial with respect to the size of the input term using a matching DAG or index tree. Since in our framework neededness of a redex may depend on other redexes in a term, it is highly unlikely that a similar data structure exists for the efficient compilation of call-by-need strategies for the systems in CBN_α, $\alpha \in \{\mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$.

The second problem to address in order to implement an efficient call-by-need strategy is the problem of optimizing sequences of rewrite steps: after a needed redex is identified and contracted, the search for a needed redex in the obtained term has to start from scratch.

Huet and Lévy do not address this problem. The forward-branching class is the greatest subclass of the strongly sequential class which brings a solution to this problem.

Forward-branching systems(FB) presented in Chapter 10 have the nice property that an index-position in a term can be recursively root-stabilized before inspecting it, without changing the output behaviour of the computation. When a redex is found, it is replaced by its contractum and the root-stabilization process continues without restarting at the root of the term. Such algorithm will examine at most twice every symbol created along the sequence of rewrite steps (including the symbols of the initial term). In other words, *outermost evaluation* can be preserved while doing *innermost root-stabilization*. This means that every part of the subject term that we need to inspect can be reduced to strong

root-stable form *before* being inspected. This surprising property means that we can do ordinary recursive descent using the stack architecture of a traditional machine instead of keeping track of complicated state transitions. This gives us the advantage of high execution speed in combination with the advantages of the semantics of outermost evaluation.

The third possibility for optimizing sequences of rewrite steps is to try to avoid building intermediate contracta (instances of right-handsides) that will not be part of the final normal form). Is is argued in [Str88] that most of the time is spent in building right-handsides. That problem is present with all types of rewriting stategies, even for classes where the redex search is trivial. This problem can be classified into the domain of *partial evaluation*. The forward-branching class is well suited to perform this type of optimization. Work in that direction can be found in [Str88], [SSD91].

Much of the right-handside structure created in the course of execution is used solely to drive further pattern-matching, and does not appear in the final output. An approach is needed that avoids creating this strictly intermediate structure. Wadler addressed this need for the case of linear terms in so-called *treeless* form in[Wad88]; in this work, he exhibited a program transformation that avoided construction of tree structures for functions that could be computed with bounded internal storage.

In forward-branching systems, a further important advantage of innermost root-stabilization is that we can easily predict pattern-matching moves immediately following replacement by a right-handside: we simply run the new right-handside through the positive arcs to predict the automaton moves. Predicting these moves at compile-time means that we don't have to perform them at run-time. Doing so can often save us the trouble of actually constructing the right-handside.

From the forward-branching index tree, we generate code in an intermediate language called EM code [SS90]. That code is then further processed to create code for the physical machine. It is showed in [Str88] how to transform the intermediate code before generating machine code from it using a technique related to partial evaluation [BAOE76] [Har77] [JSS85]. An algorithm was given in [Str88] to perform the partial evaluation, but it used complicated program and data structures. Work by Bondorf[Bon89] used similar techniques to improve constructor-based term-rewriting systems, with the specific application of producing good compilers from interpretive language specifications.

[SS90] presents an improved intermediate language with more precise definition and stricter formal properties. These formal properties enable us to show that our transformations preserve the semantics of the program, as well as to perform more general optimizations. In [SSD91], we give an overview of optimizations that are needed in order to achieve an efficient treatment of right-handsides. Then we translate the optimizations to transformations on the intermediate language; these transformations are implemented with local rewrite rules on the intermediate code. We finally discuss the effectiveness of the method, and discuss methods to prove its correctness and its termination.

Part IV

`Autowrite`: a tool for handling systems and term automata

Chapter 15

Autowrite

15.1 What is Autowrite?

Autowrite is an experimental software tool written in Common Lisp for handling term rewrite systems and bottom-up tree automata. A graphical interface has been written using McCLIM, (the free implementation of the CLIM specification) in order to free the user of any Lisp knowledge. Software and documentation can be found at

http://dept-info.labri.u-bordeaux.fr/~idurand/autowrite

Autowrite was initially designed to check call-by-need properties of term rewrite systems. For this purpose, it implements the tree automata constructions used in [Jac96b, DM97, DM98, NT02] and many useful operations on terms, term rewrite systems and tree automata. In the first version of Autowrite [Dur02], only the call-by-need properties and a few other simple properties were available from the graphical interface. This new version of Autowrite [Dur04, Dur05] includes many new functionalities. There are new functionalities related to systems, but the most interesting new feature is the possibility to directly handle (load, save, combine with boolean operations) bottom-up tree automata. In addition, we have added on-line timing information. Since the first version the run-times have been considerably improved due to better choices of data structures. The first version of Autowrite was used to check call-by-need for most of the examples presented in [DM01]. Most of the time no alternative proofs exists. The new features allowed testing many properties of examples presented in [DM05] for which no easy proof can be written.
The following example is taken from [KM91].

Example 15.1.1.

$$\mathcal{R}_1 = \begin{cases} \mathsf{f}(\mathsf{g}(\mathsf{a}, x), \mathsf{b}) & \to & x \\ \mathsf{f}(\mathsf{g}(x, \mathsf{a}), \mathsf{c}) & \to & x \\ \mathsf{f}(\mathsf{d}, x) & \to & x \\ \mathsf{g}(\mathsf{e}, \mathsf{e}) & \to & \mathsf{e} \end{cases}$$

For $\mathcal{R}_1$, we obtain the following approximated systems:

$$\mathsf{s}(\mathcal{R}_1) = \begin{cases} \mathsf{f}(\mathsf{g}(\mathsf{a},x),\mathsf{b}) & \to & y \\ \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{c}) & \to & y \\ \mathsf{f}(\mathsf{d},x) & \to & y \\ \mathsf{g}(\mathsf{e},\mathsf{e}) & \to & y \end{cases} \qquad \mathsf{nv}(\mathcal{R}_1) = \begin{cases} \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{b}) & \to & y \\ \mathsf{f}(\mathsf{g}(\mathsf{a},x),\mathsf{c}) & \to & y \\ \mathsf{f}(\mathsf{d},x) & \to & y \\ \mathsf{g}(\mathsf{e},\mathsf{e}) & \to & \mathsf{e} \end{cases}$$

$$\mathsf{g}(\mathcal{R}_1) = \begin{cases} \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{b}) & \to & y \\ \mathsf{f}(\mathsf{g}(\mathsf{a},x),\mathsf{c}) & \to & y \\ \mathsf{f}(\mathsf{d},x) & \to & x \\ \mathsf{g}(\mathsf{e},\mathsf{e}) & \to & \mathsf{e} \end{cases}$$

Example 15.1.2. *Example $\mathcal{R}_2$ comes from [Oya93].*

$$\mathsf{nv}(\mathcal{R}_2) = \begin{cases} \mathsf{f}(\mathsf{g}(\mathsf{a},x),\mathsf{a}) & \to & \mathsf{c} \\ \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{b}) & \to & \mathsf{c} \\ \mathsf{f}(\mathsf{k}(\mathsf{a}),x) & \to & \mathsf{c} \\ \mathsf{g}(\mathsf{b},\mathsf{b}) & \to & \mathsf{h}(\mathsf{b}) \\ \mathsf{h}(x) & \to & \mathsf{k}(y) \end{cases} \qquad \mathsf{g}(\mathcal{R}_2) = \mathcal{R}_2$$

Example 15.1.3. *Example $\mathcal{R}_3$ is an extension of Berry's example [Ber78].*

$$\mathcal{R}_3 = \begin{cases} \mathsf{f}(x,\mathsf{a},\mathsf{b}) & \to & \mathsf{h}(x) \\ \mathsf{f}(\mathsf{b},x,\mathsf{a}) & \to & \mathsf{h}(x) \\ \mathsf{f}(\mathsf{a},\mathsf{b},x) & \to & \mathsf{h}(x) \\ \mathsf{h}(\mathsf{k}(x)) & \to & \mathsf{g}(x,x) \\ \mathsf{g}(\mathsf{a},\mathsf{a}) & \to & \mathsf{g}(\mathsf{a},\mathsf{a}) \\ \mathsf{g}(\mathsf{a},\mathsf{b}) & \to & \mathsf{a} \\ \mathsf{g}(\mathsf{b},\mathsf{a}) & \to & \mathsf{b} \end{cases} \qquad \mathsf{g}(\mathcal{R}_3) = \begin{cases} \mathsf{f}(x,\mathsf{a},\mathsf{b}) & \to & \mathsf{h}(x) \\ \mathsf{f}(\mathsf{b},x,\mathsf{a}) & \to & \mathsf{h}(x) \\ \mathsf{f}(\mathsf{a},\mathsf{b},x) & \to & \mathsf{h}(x) \\ \mathsf{h}(\mathsf{k}(x)) & \to & \mathsf{g}(y,y) \\ \mathsf{g}(\mathsf{a},\mathsf{a}) & \to & \mathsf{g}(\mathsf{a},\mathsf{a}) \\ \mathsf{g}(\mathsf{a},\mathsf{b}) & \to & \mathsf{a} \\ \mathsf{g}(\mathsf{b},\mathsf{a}) & \to & \mathsf{b} \end{cases}$$

`Autowrite` computes $\alpha(\mathcal{R})$ for any approximation α in $\{\mathsf{s},\mathsf{nv},\mathsf{g}\}$.

Theorem 15.1.4. *The approximation mappings* s, nv, *and* g *are regularity preserving.*

Nagaya and Toyama [NT02] proved the above result for the growing approximation; the tree automaton that recognizes $(\to_{\mathsf{g}}^*)[L]$ is defined as the limit of a finite saturation process. This saturation process is similar to the ones defined in Comon [Com00] and Jacquemard [Jac96b], but by working exclusively with deterministic tree automata, non-right-linear rewrite rules can be handled.

15.2 Real Problems solved by `Autowrite`

15.2.1 Convince someone that $\mathcal{R} \in \mathsf{CBN}$ for a given $\mathcal{R}$

It is quite easy to convince someone that a system is not in CBN by exhibiting a term with no $\mathcal{R}$-needed redex. However convincing someone that a system

is in CBN is not an easy matter; any attempt generally ends up in a long and tedious proof which in addition will work only for the particular system considered. Often, in papers about Call-By-Need (or Sequentiality) the authors always prove that some $\mathcal{R} \notin \mathsf{CBN}$ but never that some $\mathcal{R} \in \mathsf{CBN}$. Usually, they just conjecture or say they think that the system is in CBN.

For systems of reasonable size, we can use `Autowrite` to convince the reader that a systems is in CBN. Also, when looking for a system in CBN and having particular properties, we were often surprised to learn from `Autowrite` that the systems was not in CBN contrary to our intuition. With the term with no $\mathcal{R}$-needed redex exposed by `Autowrite` we would be right away convinced of our mistake.

Take for instance the example of Oyamaguchi who in [Oya93] conjectured that the system $\mathcal{R}_2$ is nv-sequential. With `Autowrite` one can easily check that $\mathcal{R}_2 \in \mathsf{CBN}_{\mathsf{nv}}$. This does not imply that it is nv-sequential but shows that there exists an optimal and computable strategy for $\mathcal{R}_2$.

15.2.2 Properties related to signature extension

Let $\mathcal{R}$ be a left-linear growing esystem. In Chapter 4, [DM01, DM05] we have studied the question whether the property that $\mathcal{R} \in \mathsf{CBN}$ is preserved after adding new function symbols. For that problem, we need to specify the underlying signature in our notation. We write $(\mathcal{R}, \mathcal{G})$ instead of just $\mathcal{R}$ to indicate which signature is used. We write $\mathsf{NF}(\mathcal{R}, \mathcal{F})$ for the set of ground normal forms of an esystem $\mathcal{R}$ over a signature $\mathcal{F}$. We write $\mathsf{WN}_\bullet(\mathcal{R}, \mathcal{G}, \mathcal{F})$ for $\mathsf{WN}(\mathcal{R}_\bullet, \mathcal{G}_\bullet, \mathcal{F}_\bullet)$.

Proposition 4.1.15 states that for $(\mathcal{R}, \mathcal{F}) \in \mathsf{CBN}$, if $(\mathcal{R}, \{\mathcal{F} \cup @\}) \in \mathsf{CBN}$ (for some fresh constant @) then $(\mathcal{R}, \mathcal{G}) \in \mathsf{CBN}$ for any $\mathcal{G}$ such that $\mathcal{F} \subseteq \mathcal{G}$.

This why `Autowrite` provides the possibility of testing whether $(\mathcal{R}, \mathcal{G}) \in \mathsf{CBN}$ with $\mathcal{G} = \mathcal{F} \cup \{@\}$.

$\mathsf{ENF}(\mathcal{R}) \neq \varnothing$ is a sufficient condition for $\mathcal{R} \in \mathsf{CBN}$ being preserved CBN under signature extension.

When $\mathsf{ENF}(\mathcal{R}) = \varnothing$, orthogonality is needed in all the sufficient conditions that we have obtained.

For orthogonal nv esystems, the condition that $\mathcal{R}$ is collapsing and $\mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}, \mathcal{F})$ is sufficient. `Autowrite` helped us find examples showing that both restrictions where essential.

The following example shows the necessity of the collapsing condition:

Example 15.2.1. *Let $\mathcal{R}_4$ be the following orthogonal system:*

$$\begin{aligned}
\mathsf{f}(x,\mathsf{a},\mathsf{b}(y,z)) &\to \mathsf{c}(\mathsf{i}) & \mathsf{f}(\mathsf{c}(x),\mathsf{c}(y),z) &\to \mathsf{i}\\
\mathsf{f}(x,\mathsf{a},\mathsf{c}(y)) &\to \mathsf{i} & \mathsf{g}(x) &\to \mathsf{b}(x,\mathsf{i})\\
\mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{a}) &\to \mathsf{i} & \mathsf{h}(\mathsf{a}) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{a},\mathsf{b}(x,y),z) &\to \mathsf{a} & \mathsf{h}(\mathsf{b}(\mathsf{a},x)) &\to \mathsf{a}\\
\mathsf{f}(\mathsf{a},\mathsf{c}(x),y) &\to \mathsf{i} & \mathsf{h}(\mathsf{b}(\mathsf{b}(x,y),z)) &\to \mathsf{b}(\mathsf{i},\mathsf{i})\\
\mathsf{f}(\mathsf{b}(x,y),z,\mathsf{a}) &\to \mathsf{a} & \mathsf{h}(\mathsf{b}(\mathsf{c}(x),y)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x,y),\mathsf{b}(z,u),\mathsf{b}(v,w)) &\to \mathsf{i} & \mathsf{h}(\mathsf{c}(x)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x,y),\mathsf{b}(z,u),\mathsf{c}(v)) &\to \mathsf{i} & \mathsf{j}(\mathsf{a},\mathsf{a}) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x,y),\mathsf{c}(z),\mathsf{b}(u,v)) &\to \mathsf{i} & \mathsf{j}(\mathsf{a},\mathsf{b}(x,y)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x,y),\mathsf{c}(z),\mathsf{c}(u)) &\to \mathsf{i} & \mathsf{j}(\mathsf{a},\mathsf{c}(x)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{c}(x),\mathsf{a},\mathsf{a}) &\to \mathsf{i} & \mathsf{j}(\mathsf{b}(x,y),z) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{c}(x),\mathsf{b}(y,z),\mathsf{a}) &\to \mathsf{i} & \mathsf{j}(\mathsf{c}(x),y) &\to \mathsf{a}\\
\mathsf{f}(\mathsf{c}(x),\mathsf{b}(y,z),\mathsf{c}(u)) &\to \mathsf{i} & \mathsf{i} &\to \mathsf{i}\\
\mathsf{f}(\mathsf{c}(x),\mathsf{b}(y,z),\mathsf{b}(u,v)) &\to \mathsf{i}
\end{aligned}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules and let $\mathcal{G} = \mathcal{F} \cup \{@\}$.

Autowrite is able to check that

- $\mathsf{ENF}(\mathcal{R}_4) = \emptyset$,
- $(\mathcal{R}_4, \mathcal{F}) \in \mathsf{CBN}_{\mathsf{nv}}$,
- $(\mathcal{R}_4, \mathcal{G}) \notin \mathsf{CBN}_{\mathsf{nv}}$ *as shown by the term with no* $(\mathsf{nv}(\mathcal{R}_4), \mathcal{G})$*-needed redex* $\mathsf{j}(\mathsf{f}(\Delta,\Delta,\Delta),@)$ *with* $\Delta = \mathsf{h}(\mathsf{g}(@))$,
- $\mathsf{WN}(\mathsf{nv}(\mathcal{R}_4), \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathsf{nv}(\mathcal{R}_4), \mathcal{F})$.

One can verify easily that $\mathsf{j}(\mathsf{f}(\Delta,\Delta,\Delta),@)$ *has no* $\mathsf{nv}(\mathcal{R}_4), \mathcal{G})$*-needed redex:*

$\Delta = \mathsf{h}(\mathsf{g}(@)) \to_{\mathsf{nv}} \mathsf{h}(\mathsf{b}(\mathsf{a},\mathsf{i})) \to_{\mathsf{nv}} \mathsf{a}$ $\quad$ $\Delta = \mathsf{h}(\mathsf{g}(@)) \to_{\mathsf{nv}} \mathsf{h}(\mathsf{b}(\mathsf{b}(\mathsf{a},\mathsf{a}),\mathsf{i})) \to_{\mathsf{nv}} \mathsf{b}(\mathsf{i},\mathsf{i})$

$$\begin{array}{llllllll}
\mathsf{j}(\mathsf{f}(\bullet,\Delta,\Delta),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{f}(\bullet,\mathsf{a},\mathsf{b}(\mathsf{i},\mathsf{i})),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{c}(\mathsf{i}),@) & \to_{\mathsf{nv}} & \mathsf{a} \in \mathsf{NF}(\mathcal{R},\mathcal{G})\\
\mathsf{j}(\mathsf{f}(\Delta,\bullet,\Delta),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{f}(\mathsf{b}(\mathsf{i},\mathsf{i}),\bullet,\mathsf{a}),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{a},@) & & \in \mathsf{NF}(\mathcal{R},\mathcal{G})\\
\mathsf{j}(\mathsf{f}(\Delta,\Delta,\bullet),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{f}(\mathsf{a},\mathsf{b}(\mathsf{i},\mathsf{i}),\bullet),@) & \to_{\mathsf{nv}} & \mathsf{j}(\mathsf{a},@) & & \in \mathsf{NF}(\mathcal{R},\mathcal{G})
\end{array}$$

The next example in this section shows the necessity of the restriction to $\alpha \in \{\mathsf{s},\mathsf{nv}\}$.

Example 15.2.2. *Let $\mathcal{R}_5$ be the following orthogonal esystem:*

$$\begin{aligned}
\mathsf{f}(x,\mathsf{a},\mathsf{b}(y),z) &\to \mathsf{g}(z) & \mathsf{g}(\mathsf{a}) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x),y,\mathsf{a},z) &\to \mathsf{g}(z) & \mathsf{g}(\mathsf{b}(x)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{a},\mathsf{b}(x),y,z) &\to \mathsf{g}(z) & \mathsf{h}(\mathsf{a}) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{a},\mathsf{a},\mathsf{a},x) &\to \mathsf{i} & \mathsf{h}(\mathsf{b}(x)) &\to \mathsf{j}(\mathsf{i},x)\\
\mathsf{f}(\mathsf{b}(x),\mathsf{b}(y),\mathsf{b}(z),u) &\to \mathsf{i} & \mathsf{j}(x,\mathsf{a}) &\to \mathsf{a}\\
\mathsf{i} &\to \mathsf{i} & \mathsf{j}(x,\mathsf{b}(y)) &\to \mathsf{b}(\mathsf{a})
\end{aligned}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules. Note that the growing approximation only modifies the rule $\mathsf{h}(\mathsf{b}(x)) \to \mathsf{j}(\mathsf{i},x)$ into $\mathsf{h}(\mathsf{b}(x)) \to \mathsf{j}(\mathsf{i},y)$. Let $\mathcal{G} = \mathcal{F} \cup \{@\}$.

Autowrite is able to check that

- $\mathsf{ENF}(\mathcal{R}_5) = \emptyset$,
- $(\mathcal{R}_5,\mathcal{F}) \in \mathsf{CBN}_\mathsf{g}$,
- $(\mathcal{R}_5,\mathcal{G}) \notin \mathsf{CBN}_\mathsf{g}$ *as shown by the term with no* $\mathsf{g}(\mathcal{R}_5)$*-needed redex* $\mathsf{f}(\Delta,\Delta,\Delta,@)$, *with* $\Delta = \mathsf{h}(\mathsf{j}(@))$,
- $\mathsf{WN}(\mathsf{g}(\mathcal{R}_5),\mathcal{F}) = \mathsf{WN}(\mathsf{g}(\mathcal{R}_5),\mathcal{G},\mathcal{F})$.

Note that $\mathcal{R}$ is not collapsing. This is not essential, since adding the single collapsing rule $\mathsf{k}(x) \to x$ to $\mathcal{R}$ does not affect any of the above properties.

For an nv esystem $\mathcal{R}$, we have the nice property that $\mathsf{WN}(\mathcal{R},\mathcal{G},\mathcal{F}) = \mathsf{WN}(\mathcal{R},\mathcal{F}) \Rightarrow \mathsf{WN}_\bullet(\mathcal{R},\mathcal{G},\mathcal{F}) = \mathsf{WN}_\bullet(\mathcal{R},\mathcal{F})$. `Autowrite` helped us show that the restriction to nv is essential for this implication.

Example 15.2.3. *Let $\mathcal{R}_6$ be the following orthogonal system:*

$$\begin{aligned}
\mathsf{f}(x,\mathsf{a}) &\to \mathsf{a} & \mathsf{h}(x,\mathsf{a},\mathsf{a}) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{a},\mathsf{b}(x)) &\to \mathsf{i} & \mathsf{h}(x,\mathsf{a},\mathsf{b}(y)) &\to \mathsf{i}\\
\mathsf{f}(\mathsf{b}(x),\mathsf{b}(y)) &\to \mathsf{i} & \mathsf{h}(x,\mathsf{b}(y),\mathsf{a}) &\to \mathsf{i}\\
\mathsf{g}(\mathsf{a},\mathsf{a}) &\to \mathsf{i} & \mathsf{h}(x,\mathsf{b}(y),\mathsf{b}(z)) &\to \mathsf{b}(\mathsf{g}(y,\mathsf{f}(x,z)))\\
\mathsf{g}(\mathsf{b}(x),\mathsf{a}) &\to \mathsf{i} & \mathsf{i} &\to \mathsf{b}(\mathsf{i})\\
\mathsf{g}(x,\mathsf{b}(y)) &\to \mathsf{a}
\end{aligned}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules and let $\mathcal{G} = \mathcal{F} \cup \{@\}$.

Autowrite is able to check

- $\mathsf{ENF}(\mathcal{R}_6) = \emptyset$,
- $\mathsf{WN}(\mathsf{g}(\mathcal{R}_6),\mathcal{G},\mathcal{F}) = \mathsf{WN}(\mathsf{g}(\mathcal{R}_6),\mathcal{F})$,
- $\mathsf{WN}_\bullet(\mathsf{g}(\mathcal{R}_6),\mathcal{G},\mathcal{F}) \neq \mathsf{WN}_\bullet(\mathsf{g}(\mathcal{R}_6),\mathcal{F})$ *as shown by the term* $t = \mathsf{h}(\bullet,\mathsf{i},\mathsf{i})$.

15.2.3 Forward-branching systems

Let $\mathcal{R}$ be an orthogonal system. `Autowrite` provides the possibility of testing whether $\mathcal{R}$ is forward-branching using either the characterization of Theorem 10.3.19 (cubic time complexity) or by attempting to construct of forward-branching index-tree (quadratic time complexity) (see Section 10.4.3). It also implements the transformation from forward-branching systems to constructor strongly sequential systems described in Section 11.1.1.

15.3 The Inside of `Autowrite`

The most important object in `Autowrite` is the tree automaton. Since the first version of `Autowrite` [Dur02], much care has been devoted to improve the representation of automata. Consequently, the performances have improved significantly.

Each state of an automaton is represented by a unique Common Lisp object. Comparing two states is then very cheap: we just need to compare the references of the states. An automaton is represented by its signature (a list of symbols), a list of references to its states and its rules. A system is represented by its signature and its rules. The set of rules (of an automaton or a systems) is represented by a hash-table which given a key associated with a left-hand side of a rule gives the corresponding right-handside (or a list of corresponding right-handsides if the automaton is not deterministic). Given a left-hand side $f(q_1, \ldots, q_n)$, the corresponding key consists of a list containing the root symbol f followed by the references of the states $q_1, \ldots, q_n$.

During the construction of an automaton (for instance during the construction of the $\mathcal{C}_{\mathcal{R},\mathcal{A}}$) the rules of an automaton may be represented as a simple list of rules. But as soon as the construction is completed, the list of rules is converted into a hash-table as described above.

In general we use as much as possible '"sharing" versus "copying" data structures and use hashtables instead of lists. When possible we use memoizing techniques to avoid recomputing several times identical calls. The later may explain differences of timing when the same operations are performed in different order.

15.4 The Outside of `Autowrite`

15.4.1 `Autowrite` specifications

`Autowrite` handles a set of specifications that can be loaded interactively. A specification consists of a signature, possibly a set of variables, followed by a list of `Autowrite` objects. `Autowrite` objects are systems, automata, sets of terms and single terms. Figure 15.4.1 shows an example of such a specification. That specification defines a signature in which integers and arithmetic expressions using `+` and `*` may be represented, a system named `R` that may be used to

simplify arithmetic expressions, an automaton named EVEN which recognizes the set of even integers, two sets of terms named RS (for root-stable) and T(F), and four ground terms.

```
Ops 0:0 s:1 +:2 *:2
Vars x y
TRS R
; addition
+(0,x) -> x
+(s(x),y) -> s(+(x,y))
; product
*(0,x) -> 0
*(s(x),y) -> +(*(x,y),y)

Automaton EVEN
States odd even
Final States even
Transitions
0 -> even
s(even) -> odd
s(odd) -> even

Termset RS 0 s(x)
Termset "T(F)" x

Term *(*(0,s(0)),+(0,s(0)))
Term *(o,+(0,s(0)))
Term *(*(0,s(0)),o)
Term s(s(s(0)))
```

Figure 15.1: Example of an Autowrite specification

15.4.2 Automata operations performed by Autowrite

Checking properties of an automaton

Here are the different decision problems about an automaton that can be solved with Autowrite.

- Given an automaton $\mathcal{A}$: decide whether $L(\mathcal{A})$ is empty.
- Given two automata $\mathcal{A}$ and $\mathcal{B}$: decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$.
- Given two automata $\mathcal{A}$ and $\mathcal{B}$: decide whether $L(\mathcal{A}) = L(\mathcal{B})$.

- Given two automata $\mathcal{A}$ and $\mathcal{B}$: decide whether $L(\mathcal{A}) \cap L(\mathcal{B})$ is empty.
  For this latter operation the intersection automaton is not computed, rather we incrementally compute its accessible states and stop as soon as a final state is found.

When a property is not satisfied `Autowrite` exhibits a ground term exposing the failure.

The screenshot of Figure 15.4.2 shows operations concerning the current term and the current automaton performed after loading the specification shown in Figure 15.4.1. The automaton recognizing $\mathsf{NF}(\mathcal{R})$ is first computed. Then we check that the current term is not recognized by the current automaton (as it is not a normal form). Next we compute the complement of the current automaton (which recognizes reducible terms) and check that the current term is recognized by the complement automaton. Finally, we check that the complement automaton does not recognize the empty language.

Building new automata

Here are the different automata transformations or constructions handled by `Autowrite`.

- Given an automaton $\mathcal{A}$: compute $Det(\mathcal{A})$, the determinized version of $\mathcal{A}$.
- Given an automaton $\mathcal{A}$: compute $\mathcal{A}^c$ recognizing $L(\mathcal{A})^c$ the complement of $L(\mathcal{A})$ in the whole set of ground terms.
- Given an automaton $\mathcal{A}$: compute $Red(\mathcal{A})$, the *reduced* version of $\mathcal{A}$ *i.e.* such that every state is accessible.
- Given two automata $\mathcal{A}$ and $\mathcal{B}$: compute $\mathcal{A} \cap \mathcal{B}$.
- Given two automata $\mathcal{A}$ and $\mathcal{B}$: compute $\mathcal{A} \cup \mathcal{B}$.
- Given a set of linear terms $\mathcal{L}$: compute an automaton $\mathcal{A}_\mathcal{L}$ such that $L(\mathcal{A}_\mathcal{L}) = \{\sigma(t) \mid t \in \mathcal{L} \text{ and } \sigma \text{ is a ground substitution } \}$.

The screenshot of Figure 15.4.2 shows how to perform boolean operations using `Autowrite`. We compute the intersection of the automaton recognizing normal forms and its complement. We check that the resulting automaton recognizes the empty language.

15.4.3 Building automata related to left-linear esystems

Let $\mathcal{R}$ be a left-linear esystem. `Autowrite` can build the following automata:

- Build an automaton $\mathcal{A}_{\mathsf{NF}(\mathcal{R})}$ such that $L(\mathcal{A}_{\mathsf{NF}(\mathcal{R})}) = \mathsf{NF}(\mathcal{R})$.
- Build an automaton $\mathcal{A}_{\mathsf{ENF}(\mathcal{R})}$ such that $L(\mathcal{A}_{\mathsf{ENF}(\mathcal{R})}) = \mathsf{ENF}(\mathcal{R})$.

The two following automata can be constructed only if $\mathcal{R}$ also growing:

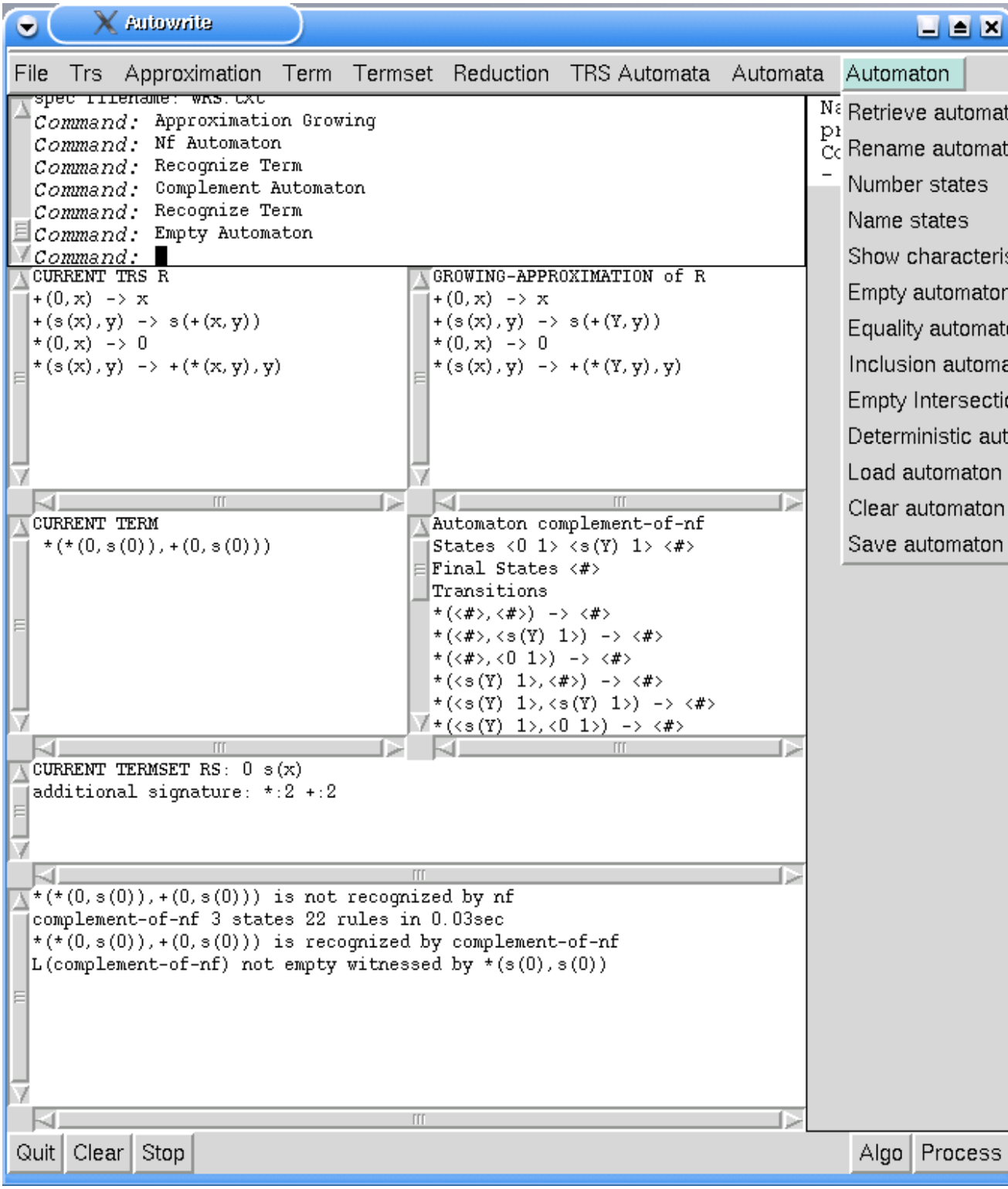


Figure 15.2: Operations on the current term and the current automaton

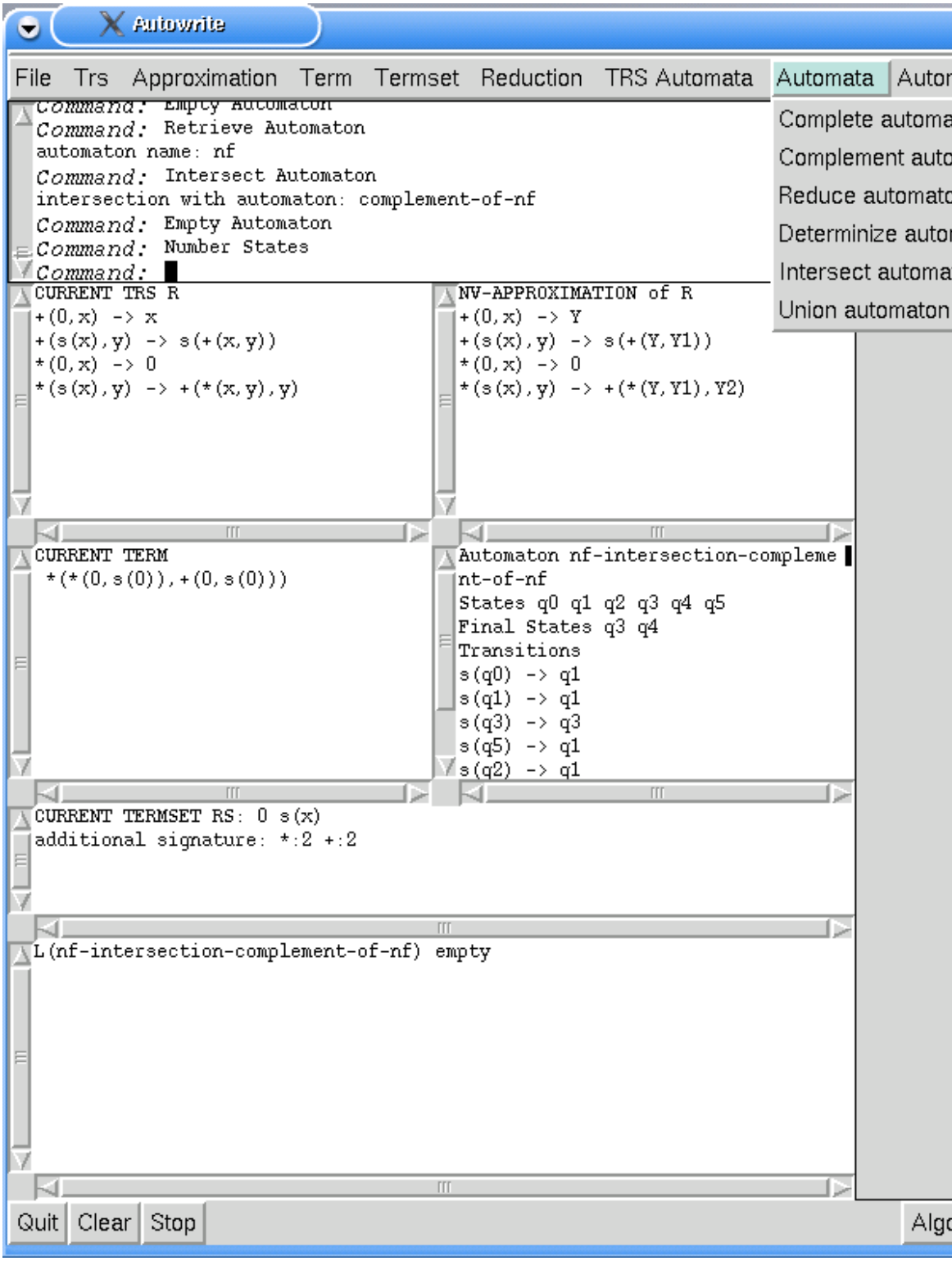


Figure 15.3: Boolean operations on automata

- Given a tree automaton $\mathcal{A}$: build a deterministic (Toyama and Nagaya's algorithm) or non-deterministic (Jaquemard's algorithm) automaton $\mathcal{C}_{\mathcal{R},\mathcal{A}}$ (as described in [NT02]) such that $L(\mathcal{C}_{\mathcal{R},\mathcal{A}}) = (\stackrel{*}{\rightarrow})[L(\mathcal{A})]$.

- Build an automaton $\mathcal{D}_{\mathcal{R}}$ such that $L(\mathcal{D}_{\mathcal{R}}) = \varnothing$ is equivalent to $\mathcal{R} \in \mathsf{CBN}$ [DM98].

For $\mathcal{C}_{\mathcal{R},\mathcal{A}}$, both Jacquemard's algorithm for linear-growing systems and Toyama and Nagaya's for left-linear-growing systems have been implemented. For $\mathcal{D}_{\mathcal{R}}$, we have implemented the algorithm presented in [DM98]. In fact these three algorithms have been adapted in order to directly compute automata with only accessible states. This complicates the code but reduces considerably the size of the construction.

The main idea is to compute the automaton incrementally. We start start building the rules having a constant left-hand side. This gives the first set of accessible states. Then we compute the rules whose left-hand sides contain the current accessible states which may give new accessible states. We stop when no new accessible state is created.

15.4.4 General properties of esystem

These are easy properties but may be useful for checking big systems.

- Check whether a system is left-linear
- Check whether a system is overlapping
- Check whether a system is orthogonal
- Check whether a system is collapsing

15.4.5 Properties of left-linear esystems

Let $\mathcal{R}$ be a left-linear esystem. The first set of properties concern only the left-hand sides of $\mathcal{R}$.

- Decide whether the set of normal forms is empty.

- Decide whether the set of external normal forms is empty.
  We will see in section 15.2.2 that non-emptiness of $\mathsf{ENF}(\mathcal{R})$ is a sufficient condition for preserving the property that $\mathcal{R} \in \mathsf{CBN}$ when the signature is extended.

Much more interesting problems can be solved when we consider left-linear **growing** esystems:

- Given a tree automaton $\mathcal{A}$ and a term t, decide whether $t \in (\rightarrow_{\mathcal{R}}^*)[L(\mathcal{A})]$. This is done by computing $\mathcal{C}_{\mathcal{R},\mathcal{A}}$ (see section 15.4.3) and check whether t is recognized by $\mathcal{C}_{\mathcal{R},\mathcal{A}}$.
  Note that this solves the accessibility problem (given two terms t, s, does $t \rightarrow_{\mathcal{R}}^* s$) as a single term s forms a regular language recognizable by a tree automaton.

- Decide whether $\mathcal{R} \in \mathsf{CBN}$.
  The method consists of building the automaton $\mathcal{D}_{\mathcal{R}}$ (see section 15.4.3) and then check whether $L(\mathcal{D}_{\mathcal{R}}) = \varnothing$. If so `Autowrite` concludes that $\mathcal{R} \in \mathsf{CBN}$, otherwise it exhibits a ground term of $L(\mathcal{D}_{\mathcal{R}})$ which is a term with no $\mathcal{R}$-needed redex. Note that to build $\mathcal{D}_{\mathcal{R}}$, `Autowrite` must previously compute $\mathcal{C}_{\mathcal{R},\mathcal{A}_{\mathsf{NF}(\mathcal{R})}}$.

- Decide whether $\mathcal{R}$ is *arbitrary i.e.* whether there exists a ground term $t \in \mathcal{T}(\mathcal{F})$ such that $t \rightarrow_{\mathcal{R},\mathcal{F}}^* \bullet$ (this means that there exists a term that may reduce to any other term).
  That latter property is relevant for the problem of signature extension (see section 15.2.2).

Concerning checking that $\mathcal{R} \in \mathsf{CBN}$, one cannot hope to use `Autowrite` for big systems because the size of the constructed automaton $\mathcal{D}_{\mathcal{R}}$ is in $\mathcal{O}(2^{2^{\|\mathcal{R}\|}})$ as shown in [DM98].

The screenshot of Figure 15.4.5 shows the use of `Autowrite` to decide membership to CBN_α classes. We show that the current system $(\mathcal{R}, \mathcal{F})$ does not belong to CBN_s, that it belongs to CBN_nv but that its extension $(\mathcal{R}, \mathcal{F}_@)$ does not belong to CBN_nv.

15.5 Experimental Results

In Table 15.1 (page 162), we present results obtained when testing membership of the above systems to CBN_α classes with various approximations α. We present the number of states (st) and rules (rl) of the automata $\mathcal{C}_{\alpha(\mathcal{R}_\bullet)}$ and $\mathcal{D}_{\alpha(\mathcal{R})}$ built to decide whether $\mathcal{R} \in \mathsf{CBN}_\alpha$. If the system is not in CBN_α, we give the witness term with no $\mathcal{R}_\alpha$-needed redex found by `Autowrite`.

Table 15.2 (page 163) shows the results obtained for the computation of the non-deterministic automaton $\mathcal{C}_{\alpha(\mathcal{R}_\bullet)}$ with Jacquemard's algorithm which is applicable only in the linear case. The three last columns show the results given by the determinization of this automaton in order to obtain an automaton similar to the deterministic one given by Toyama and Nagaya's algorithm. "NA" means that the method is not applicable (as Jacquemard's construction is only applicable to linear systems).

In Table 15.3 (page 163), we report the results of tests of the type $\mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}, \mathcal{F})$ and $\mathsf{WN}_\bullet(\mathcal{R}, \mathcal{G}, \mathcal{F}) = \mathsf{WN}_\bullet(\mathcal{R}, \mathcal{F})$. The time needed for these computations may vary depending on the fact the automaton $\mathcal{C}_{\alpha(\mathcal{R})}$ has been computed or not by previous computations.

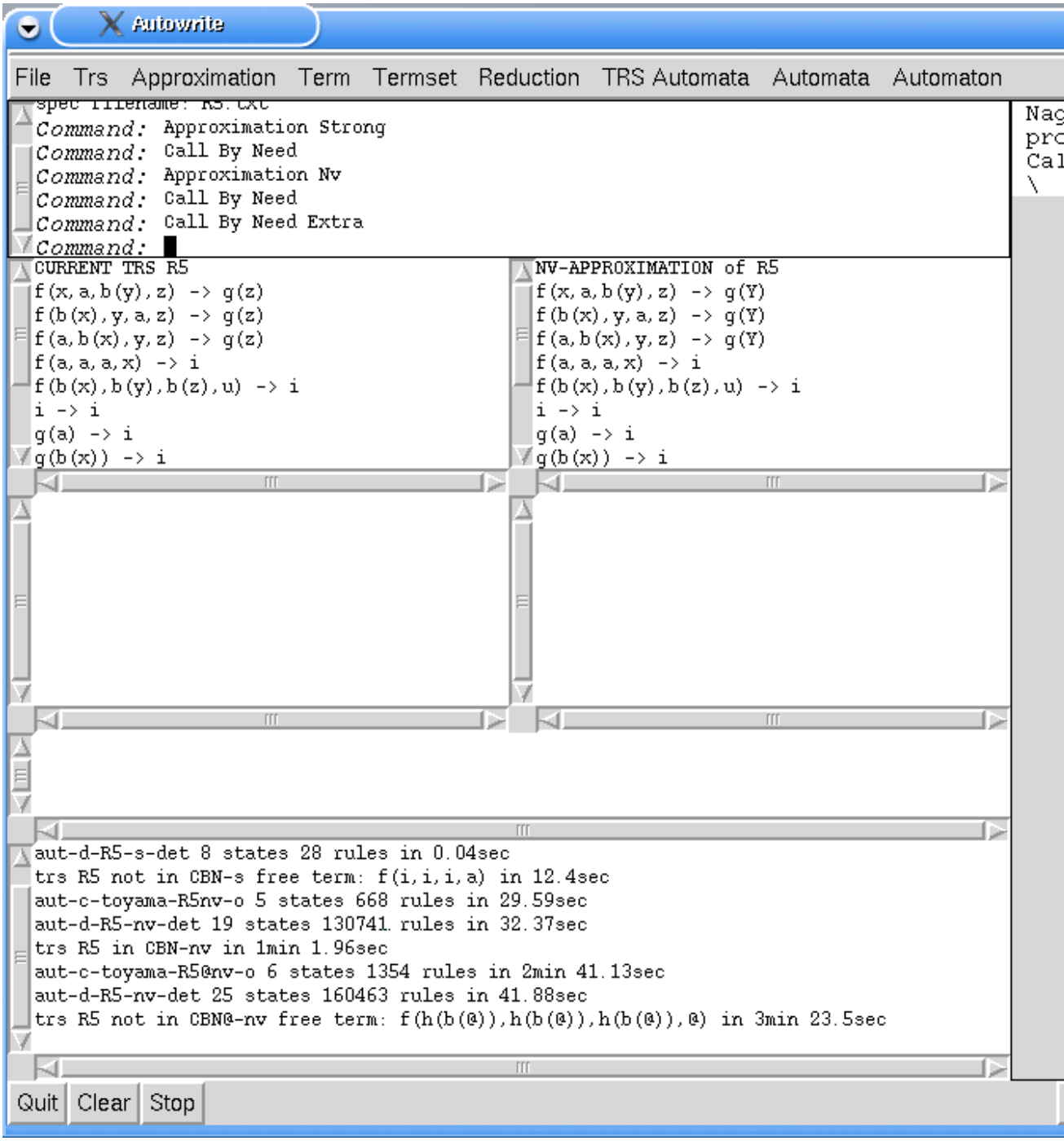


Figure 15.4: Call by need queries

Table 15.1: Call by need results

$\mathcal{R}$	α	$\mathcal{C}_{\alpha(\mathcal{R}_\bullet)}$ Toyama			$\mathcal{D}_{\alpha(\mathcal{R})}$			$(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_\alpha$
		st	rl	Time	st	rl	Time	
$\mathcal{R}_1,\mathcal{F}$	s	17	584	21s	32	727	0.12s	f(g(f(d, a), f(d, a)), f(d, a))
	nv	21	888	41s	45	4055	0s76	Yes
	g	29	1688	3m10	174	60557	48s	Yes
$\mathcal{R}_2,\mathcal{F}$	s	16	548	13s	31	687	13s	f(g(g(b, b), g(b, b)), g(b, b))
	nv	11	268	4s	17	615	0.08s	Yes
$\mathcal{R}_3,\mathcal{F}$	s	7	409	13s	11	162	0.03s	f(f(a, a, b), f(a, a, b), f(a, a, b))
	nv	9	831	50s	20	594	0.09	f(f(a, a, b), f(a, a, b), f(a, a, b))
	g	6	267	4s	17	5238	0.7s	Yes
$\mathcal{R}_4,\mathcal{F}$	s	14	3181	4m35s	12	24	0.11s	f(i, i, i)
	nv	18	6537	31m36s	85	628832	7m46s	Yes
$\mathcal{R}_4,\mathcal{G}$	nv	26	19010	4h59m	115	353013	5m27s	j(f(h(g(@)), h(g(@)), h(g(i))), @)
$\mathcal{R}_5,\mathcal{F}$	s	5	668	12s	8	28	0.04s	f(i, i, i, a)
	nv	5	668	30s	19	130741	32s	Yes
$\mathcal{R}_5,\mathcal{G}$	nv	6	1354	2m41s	25	160463	3m24s	f(h(b(@)), h(b(@)), h(b(@)), @)
$\mathcal{R}_6,\mathcal{F}$	s	5	183	1s	8	650	0.15	Yes

15.6 Comparison with other Systems

We are aware of two other distributed tools implementing tree automata: Timbuk [GT01] and RX [Wal98]. Timbuk requires the installation of `ocaml` and `RX` requires the installation of `ghc` while `Autowrite` comes self-contained. We were able able to use Timbuk (easier to install than RX). Timbuk was initially designed for computing over-approximations of the set of descendants $(\rightarrow^*_{\mathcal{R}})[L]$ for a regular language L and a system $\mathcal{R}$ and then, use it to prove unreachability. `Autowrite` can be used to check reachability (whether some term $t \in (\rightarrow^*_{\mathcal{R}})[L]$) but only for left-linear growing systems. Timbuk can handle some non-growing or non-linear cases. However, concerning efficiency of tree automata operations, `Autowrite` seems much faster: we have tried the determinization of the automaton $\mathcal{C}_{\mathsf{nv}(\mathcal{R}_5)}$ computed by Jacquemard's algorithm which runs in 2 seconds with `Autowrite`and took about 3 hours with Timbuk. The lattest version of `Autowrite` is able to load Timbuk specifications defining systems, sets of terms and automata.

15.7 Practical Information and Perspectives

The `Autowrite` project has a web page:

`http://dept-info.labri.u-bordeaux.fr/~idurand/autowrite`

From that page one can download the graphical version of `Autowrite`. The file is rather big because it contains a big part of Common Lisp and McCLIM. But the advantage is that `Autowrite` is self-contained and requires no other software. The `Autowrite` sources contain about 7500 lines of Common Lisp (including the

Table 15.2: Comparison between Jacquemard's and Toyama-Nagaya's automata

$\mathcal{R}$	α	$\mathcal{C}_{\alpha(\mathcal{R}_\bullet)}$ Jacquemard			$Det(\mathcal{C}_{\alpha(\mathcal{R}_\bullet)})$		
		st	rl	Time	st	rl	Time
$\mathcal{R}_1, \mathcal{F}$	s	12	343	20.84s	17	584	0.36s
	nv	12	333	19.09s	21	888	0.54s
	g	12	201	5.14s	29	1688	1.14s
$\mathcal{R}_2, \mathcal{F}$	s	11	381	11.44s	16	548	0.27s
	nv	11	180	2.26s	11	268	0.76s
$\mathcal{R}_3, \mathcal{F}$	s	6	214	0.22s	7	409	1.34s
	nv	6	163	3.69s	9	831	0.32s
	g	NA	NA	NA	NA	NA	NA
$\mathcal{R}_4, \mathcal{F}$	s	11	1330	1m51s	14	3181	4.26s
	nv	11	250	5.93s	18	6537	34m35s
$\mathcal{R}_5, \mathcal{F}$	s	4	287	12.95s	5	668	0.33s
	nv	4	95	1.0s	5	668	0.19s
$\mathcal{R}_6, \mathcal{F}$	s	4	126	1.52s	5	183	0.67s

Table 15.3: Preservation of normalizable terms by signature extension

$\mathcal{R}$	α	$\mathsf{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}, \mathcal{F})$	$\mathsf{WN}_\bullet(\mathcal{R}, \mathcal{G}, \mathcal{F}) = \mathsf{WN}_\bullet(\mathcal{R}, \mathcal{F})$	Time
$\mathcal{R}_5$	g	Yes	Yes	2m33
$\mathcal{R}_6$	nv	Yes	Yes	21.7s
	g	Yes	$\mathsf{h}(\bullet, \mathsf{i}, \mathsf{i})$	6.6s

graphical interface). On the Web page one can find installation directives, an on-line User's Guide and useful links. The example of an `Autowrite` session should be useful for a new user. The code can still be improved for better performances. We plan to add the possibility of minimizing a tree automaton and extend the system to other classes of automata.

`Autowrite` could be used to help students with learning about term rewriting.

Chapter 16

Conclusion

The works presented in this document were both theoretical (decidability, modularity, complexity results, algorithms) and practical (`Autowrite`).

However they were all rather technical and entirely inside the domain of term rewrite systems and call-by-need. This domain has been extensively studied and the remaining open problems seem to be extremely difficult. In particular, the problem of the complexity of deciding whether a system is in SS remains open despite our numerous efforts and attempts to solve it.

This document corresponds to a will to uniformize more than ten years of work published or unpublished in the domain. A big effort was made in order to homogenize definition and notation in order to obtain a coherent presentation.

Over the years we have appreciated the usefulness of working in parallel from a theoretical and a practical point of vue. Often, programming an algorithm helped us correct it or complete it with forgotten cases. Conversely, making the effort of describing an algorithm on a high level and in a more understandable way has often led to improvements and sometimes to a complete rewrite of the program implementing the algorithm.

In the practical perspective of effectively realizing an efficient software implementing term rewriting, the ideas presented in this work like call-by-need, modularity would be useful. However, to really obtain an interesting result one should also take into account other ideas and techniques that have been investigated in orthogonal but connected directions like partial evaluation, congruence closure, parallelism (when no sequential strategy can be used).

On a theorical perspective, we are convinced that one of the most important concept underlying this work is the notion of *preservation of recognizability* through rewriting. Each identification of a more general class of systems preserving recognizability, yields almost directly a new decidable call-by-need class, decidability results for confluence, accessibility, joinability. Also, recently, recognizability preservingness has been used to prove termination of systems for which none of the already known termination techniques work [GHWZ05]. Consequently, the seek of new decidable classes of systems that preserve recognizability is well motivated. The classes that we have presented are the easily

defined ones. However, more complicated such classes like Finite-path Overlapping systems [TKS00], Layered Tranducing systems [STFK02] have already been defined. We have ideas to extend these classes into a simplified and more general framework.

When recognizability is not preserved, we may however obtain an more general but interesting well-known structure. In that case, some results could also be derived easily. In all cases, it is interesting to study the structure of the set of descendants (or ancestors) of a set of terms with a particular structure.

Finally, with our knowledge in term rewriting and our interest in computational linguistics, our hope is to succeed in applying some rewriting techniques to that particular domain. We are also convinced that our programming experience will be of use.

Appendix A

Examples

Example A.0.1.

$$\mathcal{R}_1 = \left\{ \begin{array}{rcl} \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{a}) & \to & x \\ \mathsf{f}(\mathsf{g}(\mathsf{a},x),\mathsf{b}) & \to & \mathsf{g}(x,x) \\ \mathsf{g}(\mathsf{b},\mathsf{b}) & \to & \mathsf{b} \end{array} \right.$$

$\mathsf{LHS}_\Omega = \{\mathsf{f}(\mathsf{g}(\Omega,\mathsf{a}),\mathsf{a}), \mathsf{f}(\mathsf{g}(\mathsf{a},\Omega),\mathsf{b}), \mathsf{g}(\mathsf{b},\mathsf{b})\} \quad \mathcal{S}\mathsf{ub}^+_{\mathcal{D}}(\mathsf{LHS}_\Omega) = \{\mathsf{g}(\Omega,\mathsf{a}), \mathsf{g}(\mathsf{a},\Omega)\}$

Example A.0.2.

$$\mathcal{R}_2 = \left\{ \begin{array}{rcl} \mathsf{f}(\mathsf{a},\mathsf{g}(x,\mathsf{a})) & \to & \mathsf{b} \\ \mathsf{f}(x,\mathsf{a}) & \to & x \\ \mathsf{f}(\mathsf{b},\mathsf{g}(\mathsf{a},x)) & \to & \mathsf{b} \\ \mathsf{g}(\mathsf{b},\mathsf{b}) & \to & \mathsf{a} \end{array} \right.$$

Example A.0.3.

$$\mathcal{R}_3 = \left\{ \begin{array}{rcl} \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{a}) & \to & x, \\ \mathsf{f}(\mathsf{g}(x,\mathsf{a}),\mathsf{b}) & \to & \mathsf{g}(x,x), \\ \mathsf{g}(\mathsf{b},\mathsf{b}) & \to & \mathsf{b} \end{array} \right.$$

Example A.0.4.

$$\mathcal{R}_4 = \left\{ \begin{array}{rcl} \mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{b},x),\mathsf{a}) & \to & x, \\ \mathsf{f}(\mathsf{g}(\mathsf{a},x,\mathsf{a}),\mathsf{b}) & \to & \mathsf{g}(x,x,x), \\ \mathsf{g}(\mathsf{b},\mathsf{b},\mathsf{b}) & \to & \mathsf{b} \end{array} \right.$$

Example A.0.5.

$$\mathcal{R}_5 = \left\{ \begin{array}{rcl} \mathsf{f}(x,\mathsf{g}(y),\mathsf{h}(z)) & \to & x \\ \mathsf{f}(\mathsf{h}(z),x,\mathsf{g}(y)) & \to & x \\ \mathsf{f}(\mathsf{g}(y),\mathsf{h}(z),x) & \to & x \\ \mathsf{a} & \to & \mathsf{a} \end{array} \right.$$

Appendix B

Proofs for Sections 4.1 and 4.2

The proofs of our signature extension results follow the same strategy. We consider a system $\mathcal{R}$ over a signature $\mathcal{F}$ such that $(\mathcal{R},\mathcal{F}) \in \mathsf{CBN}_\alpha$. Let $\mathcal{G}$ be an extension of $\mathcal{F}$. Assuming that $(\mathcal{R},\mathcal{G}) \notin \mathsf{CBN}_\alpha$, we consider a minimal $(\mathcal{R}_\alpha,\mathcal{G})$-free term t in $\mathcal{T}(\mathcal{G})$. By replacing the maximal subterms of t that start with a function symbol in $\mathcal{G}\backslash\mathcal{F}$—such subterms will be called *aliens*or more precisely $\mathcal{G}\backslash\mathcal{F}$-aliens in the sequel—by a suitable term in $\mathcal{T}(\mathcal{F})$, we obtain an $(\mathcal{R}_\alpha,\mathcal{F})$-free term t' in $\mathcal{T}(\mathcal{F})$. Hence $(\mathcal{R},\mathcal{F}) \notin \mathsf{CBN}_\alpha$, contradicting the assumption.

We start with a useful lemma which is used repeatedly in the sequel.

The subset of $\mathsf{WN}(\mathcal{R},\mathcal{G},\mathcal{F})$ consisting of those terms that admit a normalizing rewrite sequence in $(\mathcal{R},\mathcal{G})$ containing a root rewrite step is denoted by $\mathsf{WNR}(\mathcal{R},\mathcal{G},\mathcal{F})$. If $\mathcal{F}=\mathcal{G}$ then we just write $\mathsf{WNR}(\mathcal{R},\mathcal{F})$ or even $\mathsf{WNR}(\mathcal{R})$ if the signature is clear from the context. We also find it convenient to write $\mathsf{WN}_\bullet(\mathcal{R},\mathcal{G},\mathcal{F})$ for $\mathsf{WN}(\mathcal{R}_\bullet,\mathcal{G}_\bullet,\mathcal{F}_\bullet)$ and $\mathsf{WNR}_\bullet(\mathcal{R},\mathcal{G},\mathcal{F})$ for $\mathsf{WNR}(\mathcal{R}_\bullet,\mathcal{G}_\bullet,\mathcal{F}_\bullet)$.

Lemma B.0.6. *Let $\mathcal{R}$ be a left-linear system and α an approximation mapping. Every minimal $\mathcal{R}_\alpha$-free term belongs to $\mathsf{WNR}(\mathcal{R}_\alpha)$.*

Proof. p Let $\mathcal{F}$ be the signature of $\mathcal{R}$ and let $t \in \mathcal{T}(\mathcal{F})$ be a minimal free term. For every redex position p in t we have $t[\bullet]_p \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha)$. Let p' be the minimum position above p at which a contraction takes place in any rewrite sequence from $t[\bullet]_p$ to a normal form in $\mathcal{T}(\mathcal{F})$ and define $P=\{p' \mid p \text{ is a redex position in } t\}$. Let p^* be a minimal position in P. We show that $p^*=\varepsilon$. If $p^*>\varepsilon$ then we consider the term t/p^*. Let q be a redex position in t/p^*. There exists a redex position p in t such that $p=p^*q$. We have $t/p^*[\bullet]_q=(t[\bullet]_p)/p^* \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha)$ by the definition of p^*. Since t/p^* has at least one redex, it follows that t/p^* is free. As t/p^* is a proper subterm of t we obtain a contradiction to the minimality of t. Hence $p^*=\varepsilon$. So there exists a redex position p in t and a rewrite sequence $A\colon t[\bullet]_p \to^+_{\mathcal{R}_\alpha,\mathcal{F}_\bullet} u \in \mathsf{NF}(\mathcal{R},\mathcal{F})$ that contains a root rewrite step. Because $\mathcal{R}_\alpha$ is left-linear and $\bullet$ does not occur in the rewrite rules of $\mathcal{R}_\alpha$, $\bullet$ cannot contribute

to this sequence. It follows that if we replace in A every position of $\bullet$ by t/p we obtain an $(\mathcal{R}_\alpha, \mathcal{F})$-rewrite sequence from t to u with a root rewrite step. □

In particular, minimal free terms are not root-stable.

Proof. PROOF of Theorem 4.1.7. Let $(\mathcal{R}, \mathcal{F}) \in \mathsf{CBN}_\alpha$ and let $c \in \mathsf{NF}(\mathcal{R}, \mathcal{F})$ be an external normal form. Let $\mathcal{F} \subseteq \mathcal{G}$. We have to show that $(\mathcal{R}, \mathcal{G}) \in \mathsf{CBN}_\alpha$. Suppose to the contrary that $(\mathcal{R}, \mathcal{G}) \notin \mathsf{CBN}_\alpha$. According to Lemma B.0.6 there exists a term $t \in \mathsf{WNR}(\mathcal{R}_\alpha, \mathcal{G})$ without $(\mathcal{R}_\alpha, \mathcal{G})$-needed redex. Let t' be the term in $\mathcal{T}(\mathcal{F})$ obtained from t by replacing every $\mathcal{G}\backslash\mathcal{F}$-alien by c. Because t is not root-stable, we have $t \to^*_{\mathcal{R}_\alpha,\mathcal{G}} L\sigma$ for some left-hand side L. Replacing in this sequence every $\mathcal{G} \setminus \mathcal{F}$-alien by c, yields a sequence $t' \to^*_{\mathcal{R}_\alpha,\mathcal{F}} l\sigma'$. So t' cannot be a normal form. Since $(\mathcal{R}, \mathcal{F}) \in \mathsf{CBN}_\alpha$, t' contains an $(\mathcal{R}_\alpha, \mathcal{F})$-needed redex Δ, say at position p. Because c is an external normal form, Δ is also a redex in t. Since t has no $(\mathcal{R}_\alpha, \mathcal{G})$-needed redexes, there exists a rewrite sequence $t[\bullet]_p \to^+_{\mathcal{R}_\alpha,\mathcal{G}_\bullet} u$ with $u \in \mathsf{NF}(\mathcal{R}_\bullet, \mathcal{G})$. If we replace in this rewrite sequence every $\mathcal{G}\backslash\mathcal{F}$-alien by c, we obtain a rewrite sequence $t'[\bullet]_p \to^+_{\mathcal{R}_\alpha,\mathcal{F}_\bullet} u'$. Because c does not unify with a proper non-variable subterm of a left-hand side of a rewrite rule, it follows that $u' \in \mathsf{NF}(\mathcal{R}_\bullet, \mathcal{F})$. Hence Δ is not an $(\mathcal{R}_\alpha, \mathcal{F})$-needed redex in t', yielding the desired contradiction. □

B.1 Proof of Theorem 4.1.9

Before we can prove Theorem 4.1.9, we need a few preliminary results.

Definition B.1.1. *Let $\mathcal{R}$ be a system. Two redexes Δ_1, Δ_2 are called* pattern equal*, denoted by $\Delta_1 \approx \Delta_2$, if they have the same redex pattern, i.e., they are redexes with respect to the same rewrite rule.*

Lemma B.1.2. *Let $\mathcal{R}$ be an orthogonal system, $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$, and suppose that $\Delta \approx \Delta'$. If $C[\Delta] \in \mathsf{WN}(\mathcal{R}_\alpha)$ then $C[\Delta'] \in \mathsf{WN}(\mathcal{R}_\alpha)$.*

Proof. Let $C[\Delta] \to^* t$ be a normalizing rewrite sequence in $\mathcal{R}_\alpha$. If we replace every descendant of Δ by Δ' then we obtain a (possibly shorter) normalizing rewrite sequence $C[\Delta'] \to^* t$. The reason is that every descendant Δ'' of Δ satisfies $\Delta'' \approx \Delta$ due to orthogonality and hence if Δ'' is contracted to some term then Δ rewrites to the same term because the variables in the right-handsides of the rewrite rules in $\mathcal{R}_\alpha$ are fresh, due to the assumption $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$. Moreover, as t is a normal form, there are no descendants of Δ left. Note that the resulting sequence can be shorter since rewrite steps below a descendant of Δ are not mimicked. □

The above lemma does not hold for the growing approximation, as shown by the following example.

Example B.1.3. *Consider the system $\mathcal{R}$*

$$\mathsf{f}(x) \to x \qquad \mathsf{a} \to \mathsf{b} \qquad \mathsf{c} \to \mathsf{c}$$

We have $\mathcal{R}_{\mathsf{g}} = \mathcal{R}$. Consider the redexes $\Delta = \mathsf{f}(\mathsf{a})$ and $\Delta' = \mathsf{f}(\mathsf{c})$. Clearly $\Delta \approx \Delta'$. Redex Δ admits the normal form b, but Δ' has no normal form.

Orthogonality is also necessary for Lemma B.1.2.

Example B.1.4. *Consider the system $\mathcal{R}$*

$$\mathsf{f}(\mathsf{a}) \to \mathsf{b} \qquad \mathsf{f}(\mathsf{g}(\mathsf{a})) \to \mathsf{a} \qquad \mathsf{g}(x) \to \mathsf{a} \qquad \mathsf{b} \to \mathsf{b}$$

We have $\mathcal{R}_{\mathsf{nv}} = \mathcal{R}$. Consider the context $C = \mathsf{f}(\Box)$ and the pattern equivalent redexes $\Delta = \mathsf{g}(\mathsf{a})$ and $\Delta' = \mathsf{g}(\mathsf{b})$. The term $C[\Delta]$ admits the normal form a, but $C[\Delta']$ has no normal form.

Lemma B.1.5. *Let $\mathcal{R}$ be an orthogonal system over a signature $\mathcal{F}$, $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$, and $\mathcal{F} \subseteq \mathcal{G}$. If $\mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ then $\mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F}) = \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$.*

Proof. The inclusion $\mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F}) \subseteq \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ is obvious. For the reverse inclusion we reason as follows. Let $t \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ and consider a rewrite sequence A in $(\mathcal{R}_\alpha, \mathcal{G}_\bullet)$ that normalizes t. We may write $t = C[t_1, \ldots, t_n]$ such that $t_1, \ldots, t_n$ are the maximal subterms of t that are rewritten in A at their root positions. Hence A can be rearranged into A':

$$t \to^*_{\mathcal{R}_\alpha, \mathcal{G}_\bullet} C[\Delta_1, \ldots, \Delta_n] \to^*_{\mathcal{R}_\alpha, \mathcal{G}_\bullet} C[u_1, \ldots, u_n]$$

for some redexes $\Delta_1, \ldots, \Delta_n$ and normal form $C[u_1, \ldots, u_n] \in \mathcal{T}(\mathcal{G})$. Since the context C cannot contain $\bullet$, all positions of $\bullet$ are in the substitution parts of the redexes $\Delta_1, \ldots, \Delta_n$. If we replace in $C[\Delta_1, \ldots, \Delta_n]$ every $\mathcal{G}_\bullet \backslash \mathcal{F}$-alien by some ground term $c \in \mathcal{T}(\mathcal{F})$, we obtain a term $t' = C[\Delta'_1, \ldots, \Delta'_n]$ with $\Delta'_i \in \mathcal{T}(\mathcal{F})$ and $\Delta_i \approx \Delta'_i$ for every i. Repeated application of Lemma B.1.2 yields $t' \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$. Because $\bullet$ cannot contribute to the creation of a normal form, we actually have $t' \in \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G})$ and thus $t' \in \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ as $t' \in \mathcal{T}(\mathcal{F})$. The assumption yields $t' \in \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F})$. Since $\mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F}) \subseteq \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F})$ clearly holds, we obtain $t' \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F})$. Now, if we replace in the first part of A' every $\mathcal{G} \backslash \mathcal{F}$-alien by c then we obtain a (possibly shorter) rewrite sequence $t \to^*_{\mathcal{R}_\alpha, \mathcal{F}_\bullet} C[\Delta''_1, \ldots, \Delta''_n] \in \mathcal{T}(\mathcal{F}_\bullet)$ with $\Delta_i \approx \Delta''_i$ and thus also $\Delta'_i \approx \Delta''_i$ for every i. Repeated application of Lemma B.1.2 yields $C[\Delta''_1, \ldots, \Delta''_n] \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F})$ and therefore $t \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F})$ as desired. □

We note that for $\alpha = \mathsf{s}$ the preceding lemma is a simple consequence of Lemma 4.1.12 below. The following example shows that the restriction to $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$ is essential.

Example B.1.6. *Consider system $\mathcal{R}$*

$$\begin{aligned}
\mathsf{f}(x, \mathsf{a}) &\to \mathsf{a} & \mathsf{h}(x, \mathsf{a}, \mathsf{a}) &\to \mathsf{i} \\
\mathsf{f}(\mathsf{a}, \mathsf{b}(x)) &\to \mathsf{i} & \mathsf{h}(x, \mathsf{a}, \mathsf{b}(y)) &\to \mathsf{i} \\
\mathsf{f}(\mathsf{b}(x), \mathsf{b}(y)) &\to \mathsf{i} & \mathsf{h}(x, \mathsf{b}(y), \mathsf{a}) &\to \mathsf{i} \\
\mathsf{g}(\mathsf{a}, \mathsf{a}) &\to \mathsf{i} & \mathsf{h}(x, \mathsf{b}(y), \mathsf{b}(z)) &\to \mathsf{b}(\mathsf{g}(y, \mathsf{f}(x, z))) \\
\mathsf{g}(\mathsf{b}(x), \mathsf{a}) &\to \mathsf{i} & \mathsf{i} &\to \mathsf{b}(\mathsf{i}) \\
\mathsf{g}(x, \mathsf{b}(y)) &\to \mathsf{a}
\end{aligned}$$

over the signature $\mathcal{F}$ consisting of all symbols appearing in the rewrite rules and let $\mathcal{G} = \mathcal{F} \cup \{\mathsf{c}\}$ with c a constant. The term $t = \mathsf{h}(\bullet, \mathsf{i}, \mathsf{i})$ belongs to $\mathsf{WN}_\bullet(\mathcal{R}_\mathsf{g}, \mathcal{G}, \mathcal{F})$:

$$t \to^+_{\mathcal{R}_\mathsf{g},\mathcal{G}_\bullet} \mathsf{h}(\bullet, \mathsf{b}(\mathsf{i}), \mathsf{b}(\mathsf{i})) \to_{\mathcal{R}_\mathsf{g},\mathcal{G}_\bullet} \mathsf{b}(\mathsf{g}(\mathsf{c}, \mathsf{f}(\bullet, \mathsf{a}))) \to_{\mathcal{R}_\mathsf{g},\mathcal{G}_\bullet} \mathsf{b}(\mathsf{g}(\mathsf{c}, \mathsf{a}))$$

However, one easily verifies that there is no normal form $u \in \mathsf{NF}(\mathcal{R}_\mathsf{g}, \mathcal{F})$ such that $t \to^*_{\mathcal{R}_\mathsf{g},\mathcal{F}_\bullet} u$. Hence $\mathsf{WN}_\bullet(\mathcal{R}_\mathsf{g}, \mathcal{F}) \neq \mathsf{WN}_\bullet(\mathcal{R}_\mathsf{g}, \mathcal{G}, \mathcal{F})$. Using the observations that (i) every term $t \in \mathcal{T}(\mathcal{F})$ rewrites to a or a term of the form $\mathsf{b}(u)$ and (ii) the only rewrite rule of $\mathcal{R}_\mathsf{g}$ where c can be introduced is $\mathsf{h}(x, \mathsf{b}(y), \mathsf{b}(z)) \to \mathsf{b}(\mathsf{g}(y', \mathsf{f}(x, z')))$ but every redex in $\mathcal{T}(\mathcal{F})$ of the form $\mathsf{h}(s, \mathsf{b}(t), \mathsf{b}(u))$ rewrites to $\mathsf{b}(\mathsf{a})$ without using c:

$$\begin{aligned}\mathsf{h}(s, \mathsf{b}(t), \mathsf{b}(u)) &\to_{\mathcal{R}_\mathsf{g},\mathcal{F}} \mathsf{b}(\mathsf{g}(\mathsf{a}, \mathsf{f}(s, \mathsf{b}(\mathsf{a})))) \\ &\to^+_{\mathcal{R}_\mathsf{g},\mathcal{F}} \mathsf{b}(\mathsf{g}(\mathsf{a}, \mathsf{i})) \qquad \textit{because } s \to^* \mathsf{a} \textit{ or } s \to^* \mathsf{b}(s') \\ &\to_{\mathcal{R}_\mathsf{g},\mathcal{F}} \mathsf{b}(\mathsf{g}(\mathsf{a}, \mathsf{b}(\mathsf{i}))) \to_{\mathcal{R}_\mathsf{g},\mathcal{F}} \mathsf{b}(\mathsf{a})\end{aligned}$$

it can be readily checked that $\mathsf{WN}(\mathcal{R}_\mathsf{g}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\mathsf{g}, \mathcal{G}, \mathcal{F})$. (`Autowrite` is able to check this equality automatically.)

A redex is called *flat* if it does not contain smaller redexes.

Lemma B.1.7. *Let $(\mathcal{R}, \mathcal{F})$ and $(\mathcal{S}, \mathcal{G})$ be orthogonal systems and $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$ such that $(\mathcal{R}, \mathcal{F}) \subseteq (\mathcal{S}, \mathcal{G})$ and $\mathsf{WN}(\mathcal{S}_\alpha, \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F})$. If $t \in \mathsf{WNR}(\mathcal{S}_\alpha, \mathcal{G})$ and $\mathsf{root}(t) \in \mathcal{F}$ then there exists a flat $\mathcal{R}$-redex Θ in $\mathcal{T}(\mathcal{F})$. Moreover, if $\mathcal{R}_\alpha$ is collapsing then we may assume that Θ is $\mathcal{R}_\alpha$-collapsing.*

Proof. From $t \in \mathsf{WNR}(\mathcal{S}_\alpha, \mathcal{G})$ we infer that $t \to^*_{\mathcal{S}_\alpha,\mathcal{G}} \Delta$ for some redex $\Delta \in \mathsf{WN}(\mathcal{S}_\alpha, \mathcal{G})$. By considering the first such redex it follows that Δ is a redex with respect to $(\mathcal{R}_\alpha, \mathcal{G})$. If we replace in Δ the subterms below the redex pattern by an arbitrary ground term in $\mathcal{T}(\mathcal{F})$ then we obtain a redex $\Delta' \in \mathcal{T}(\mathcal{F})$ with $\Delta \approx \Delta'$. Lemma B.1.2 yields $\Delta' \in \mathsf{WN}(\mathcal{S}_\alpha, \mathcal{G})$ and thus $\Delta' \in \mathsf{WN}(\mathcal{S}_\alpha, \mathcal{G}, \mathcal{F}) = \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{F})$. Hence $\mathsf{NF}(\mathcal{R}, \mathcal{F}) = \mathsf{NF}(\mathcal{R}_\alpha, \mathcal{F}) \neq \varnothing$. Therefore, using orthogonality, we obtain a flat redex $\Theta \in \mathcal{T}(\mathcal{F})$ by replacing the variables in the left-hand side of any rewrite rule in $\mathcal{R}$ by terms in $\mathsf{NF}(\mathcal{R}, \mathcal{F})$. If $\mathcal{R}_\alpha$ is collapsing then we take any $\mathcal{R}_\alpha$-collapsing rewrite rule. □

Proof. PROOF of Theorem 4.1.9 If $(\mathcal{R}, \mathcal{F})$ has external normal forms then the result follows from Theorem 4.1.7. So we assume that $(\mathcal{R}, \mathcal{F})$ lacks external normal forms. We also assume that $\mathcal{R} \neq \varnothing$ for otherwise the result is trivial. Suppose to the contrary that $(\mathcal{R}, \mathcal{G}) \notin \mathsf{CBN}_\alpha$. According to Lemma B.0.6 there exists a term $t \in \mathsf{WNR}(\mathcal{R}_\alpha, \mathcal{G})$ without $(\mathcal{R}_\alpha, \mathcal{G})$-needed redex. Lemma B.1.7 (with $\mathcal{S} = \mathcal{R}$) yields a flat redex $\Theta \in \mathcal{T}(\mathcal{F})$. Since $\mathcal{R}_\alpha$ is collapsing, we may assume that Θ is $\mathcal{R}_\alpha$-collapsing. Let t' be the term in $\mathcal{T}(\mathcal{F})$ obtained from t by replacing every $\mathcal{G}\backslash\mathcal{F}$-alien by Θ. Let P be the set of positions of those aliens. Since t' is reducible, it contains an $(\mathcal{R}_\alpha, \mathcal{F})$-needed redex, say at position q. We show that $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$. We consider two cases.

1. Suppose that $q \in P$. Since $t \in \mathsf{WNR}(\mathcal{R}_\alpha, \mathcal{G})$, $t \to^*_{\mathcal{R}_\alpha,\mathcal{G}} \Delta$ for some redex $\Delta \in \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}) \subseteq \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$. Since the root symbol of every alien belongs to $\mathcal{G}\backslash\mathcal{F}$, aliens cannot contribute to the creation of Δ and hence we may replace them by arbitrary terms in $\mathcal{T}(\mathcal{G}_\bullet)$ and still obtain a redex that is pattern equal to Δ. We replace in t the alien at position q by $\bullet$ and every alien at position $p \in P\backslash\{q\}$ by $t'/p = \Theta$. This gives $t'[\bullet]_q \to^*_{\mathcal{R}_\alpha,\mathcal{G}_\bullet} \Delta'$ with $\Delta' \approx \Delta$. Lemma B.1.2 yields $\Delta' \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$ and hence $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$.

2. Suppose that $q \notin P$. Since Θ is flat, it follows by orthogonality that q is also a redex position in t. Since t is an $(\mathcal{R}_\alpha, \mathcal{G})$-free term, $t[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$. Because Θ is a collapsing redex and $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$, we have $\Theta \to_{\mathcal{R}_\alpha,\mathcal{G}} t/p$ for all $p \in P$. Hence $t'[\bullet]_q \to^*_{\mathcal{R}_\alpha,\mathcal{G}_\bullet} t[\bullet]_q$ and thus $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G})$.

As $t' \in \mathcal{T}(\mathcal{F})$, we have $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$ and thus $t'[\bullet]_q \notin \mathsf{WN}_\bullet(\mathcal{R}_\alpha, \mathcal{F})$ by Lemma B.1.5, contradicting the assumption that q is the position of an $(\mathcal{R}_\alpha, \mathcal{F})$-needed redex in t'. □

For the proof of Theorem 4.2.4, the counterpart of Theorem 4.1.9, we need the following preliminary lemma. In the remainder of the appendix we have $\mathcal{S} = \mathcal{R} \cup \mathcal{R}'$ and $\mathcal{G} = \mathcal{F} \cup \mathcal{F}'$.

Lemma B.1.8. *Let $(\mathcal{R}, \mathcal{F})$ and $(\mathcal{R}', \mathcal{F}')$ be disjoint systems. If $\alpha \in \{\mathsf{s}, \mathsf{nv}\}$ then $\mathsf{WN}(\mathcal{S}_\alpha, \mathcal{G}, \mathcal{F}) \subseteq \mathsf{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F})$.*

Proof. We consider here the more complicated case $\alpha = \mathsf{nv}$. Let $s \in \mathsf{WN}(\mathcal{S}_\mathsf{nv}, \mathcal{G}, \mathcal{F})$, so $s \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ for some normal form $t \in \mathsf{NF}(\mathcal{S}_\mathsf{nv}, \mathcal{G})$. By induction on the length n of $s \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ we show that $s \to^*_{\mathcal{R}_\mathsf{nv},\mathcal{G}} t$. In order to make the induction work we prove this statement for all $s \in \mathcal{T}(\mathcal{G})$ such that in $s \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ no redex inside an $\mathcal{G}\backslash\mathcal{F}$-alien of s is contracted. If $n = 0$ then the statement is trivial. If $n > 0$ then there exists a term $s' \in \mathcal{T}(\mathcal{G})$ such that $s \to_{\mathcal{R}_\mathsf{nv},\mathcal{G}} s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$. Note that the rewrite rule $L \to R$ applied in the step from s to s' must come from $\mathcal{R}_\mathsf{nv}$ because redexes inside $\mathcal{G}\backslash\mathcal{F}$-aliens of s are not contracted. We have $s = C[L\sigma]$ and $s' = C[R\sigma]$ for some context C and substitution σ. If $\sigma(x) \in \mathcal{T}(\mathcal{F})$ for all $x \in \mathcal{V}\mathsf{ar}(R)$ then we can apply the induction hypothesis to $s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$. This yields $s' \to^*_{\mathcal{R}_\mathsf{nv},\mathcal{G}} t$ and thus $s \to^*_{\mathcal{R}_\mathsf{nv},\mathcal{G}} t$ as desired. If $\sigma(x) \in \mathcal{T}(\mathcal{G})\backslash\mathcal{T}(\mathcal{F})$ for some $x \in \mathcal{V}\mathsf{ar}(R)$ then s' contains new $\mathcal{G}\backslash\mathcal{F}$-aliens. If no redexes are contracted in these aliens in the $(\mathcal{S}_\mathsf{nv}, \mathcal{G})$-rewrite sequence to t then we can again apply the induction hypothesis. Otherwise we have to modify $s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ first. Let p be the position of a $\mathcal{G}\backslash\mathcal{F}$-alien in s' such that a redex in s'/p is contracted in $s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$. We distinguish two cases. If in $s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ no step takes place at a position strictly above p, then we replace s'/p by t/p. Otherwise, let $u \to_{\mathcal{S}_\mathsf{nv},\mathcal{G}} u'$ be the first step in $s' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ in which a redex is contracted at a position strictly above p. In this case we replace s'/p by u/p. It is easy to see that we end up with a rewrite sequence $s'' \to^*_{\mathcal{S}_\mathsf{nv},\mathcal{G}} t$ whose length is less than $n-1$ and with the property that redexes inside $\mathcal{G}\backslash\mathcal{F}$-aliens of s'' are not contracted. Hence we can apply the induction hypothesis, which yields $s'' \to^*_{\mathcal{R}_\mathsf{nv},\mathcal{G}} t$. Because R is a linear

term, we may write $s'' = C[R\sigma']$ for some substitution σ'. Since we are dealing with the nv approximation, $s \to_{\mathcal{R}_{\mathsf{nv}},\mathcal{G}} s''$ and therefore $s \to^*_{\mathcal{R}_{\mathsf{nv}},\mathcal{G}} t$ as desired. □

Let us illustrate the construction in the above proof on a small example.

Example B.1.9. *Consider the systems* $\mathcal{R}$

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

and $\mathcal{R}' = \{\mathsf{h}(x) \to x\}$ *over the signatures* $\mathcal{F}$ *and* $\mathcal{F}'$ *consisting of function symbols that appear in their respective rewrite rules. The* $(\mathcal{S}_{\mathsf{nv}},\mathcal{G})$*-rewrite sequence*

$$\mathsf{f}(\mathsf{a}) \to_{\mathcal{R}_{\mathsf{nv}}} \mathsf{g}(\mathsf{h}(\mathsf{a}),\mathsf{h}(\mathsf{a})) \to_{\mathcal{R}'_{\mathsf{nv}}} \mathsf{g}(\mathsf{a},\mathsf{h}(\mathsf{a})) \to_{\mathcal{R}'_{\mathsf{nv}}} \mathsf{g}(\mathsf{a},\mathsf{b}) \to_{\mathcal{R}_{\mathsf{nv}}} \mathsf{c}$$

is transformed into

$$\mathsf{f}(\mathsf{a}) \to_{\mathcal{R}_{\mathsf{nv}}} \mathsf{g}(\mathsf{a},\mathsf{b}) \to_{\mathcal{R}_{\mathsf{nv}}} \mathsf{c}$$

Note that simply replacing all $\mathcal{G}\backslash\mathcal{F}$*-aliens by some constant in* $\mathcal{F}$ *does not work.*

The reverse inclusion does not hold in general.

Example B.1.10. *Consider the systems* $\mathcal{R} = \{\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{a}), \mathsf{g}(x) \to \mathsf{f}(x)\}$ *and* $\mathcal{R}' = \{\mathsf{b} \to \mathsf{b}\}$ *over the signatures* $\mathcal{F}$ *and* $\mathcal{F}'$ *consisting of function symbols that appear in their respective rewrite rules. The term* $\mathsf{f}(\mathsf{b})$ *is a normal form with respect to* $(\mathcal{R}_{\mathsf{nv}},\mathcal{G})$ *and hence* $\mathsf{g}(\mathsf{a}) \in \mathsf{WN}(\mathcal{R}_{\mathsf{nv}},\mathcal{G},\mathcal{F})$*. One easily verifies that* $\mathsf{g}(\mathsf{a}) \notin \mathsf{WN}(\mathcal{S}_{\mathsf{nv}},\mathcal{G},\mathcal{F})$*.*

B.2 Proof of Theorem 4.2.4

Proof. PROOF of Theorem 4.2.4. We assume that both $\mathcal{R}$ and $\mathcal{R}'$ are non-empty, for otherwise the result follows from Theorem 4.1.9. Suppose to the contrary that $(\mathcal{S},\mathcal{G}) \notin \mathsf{CBN}_\alpha$. According to Lemma B.0.6 there exists a term $t \in \mathsf{WNR}(\mathcal{S}_\alpha,\mathcal{G})$ without $(\mathcal{S}_\alpha,\mathcal{G})$-needed redex. Assume without loss of generality that $\mathsf{root}(t) \in \mathcal{F}'$. Lemma B.1.7 yields a flat $\mathcal{R}'_\alpha$-collapsing redex $\Theta \in \mathcal{T}(\mathcal{F}')$. Let t' be the term in $\mathcal{T}(\mathcal{F}')$ obtained from t by replacing every $\mathcal{G}\backslash\mathcal{F}'$-alien by Θ. Let P be the set of positions of those aliens. Since t' is reducible, it contains an $(\mathcal{R}'_\alpha,\mathcal{F}')$-needed redex, say at position q. We show that $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha,\mathcal{G})$. Because Θ is a collapsing redex, we have $\Theta \to_{\mathcal{R}_\alpha,\mathcal{G}} t/p$ for all $p \in P$. Hence $t' \to^*_{\mathcal{R}_\alpha,\mathcal{G}_\bullet} t$ and thus, by orthogonality, $t'[\bullet]_q \to^*_{\mathcal{R}_\alpha,\mathcal{G}_\bullet} t[\bullet]_q$. Hence it suffices to show that $t[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha,\mathcal{G})$. We distinguish two cases.

1. Suppose that $q \in P$. Since $t \in \mathsf{WNR}(\mathcal{S}_\alpha,\mathcal{G})$, $t \to^*_{\mathcal{S}_\alpha,\mathcal{G}} \Delta$ for some redex $\Delta \in \mathsf{WN}(\mathcal{S}_\alpha,\mathcal{G}) \subseteq \mathsf{WN}_\bullet(\mathcal{S}_\alpha,\mathcal{G})$. We distinguish two further cases.

   (a) If t/q is a normal form then it cannot contribute to the creation of Δ and hence by replacing it by $\bullet$ we obtain $t[\bullet]_q \to^*_{\mathcal{S}_\alpha,\mathcal{G}} \Delta'$ with $\Delta \approx \Delta'$. Lemma B.1.2 yields $\Delta' \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha,\mathcal{G})$ and thus $t[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha,\mathcal{G})$.

(b) Suppose t/q is reducible. Because t is a minimal free term, t/q contains an $(\mathcal{S}_\alpha, \mathcal{G})$-needed redex, say at position q'. So $t/q[\bullet]_{q'} \notin \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$. In particular, $t/q[\bullet]_{q'}$ does not $(\mathcal{S}_\alpha, \mathcal{G})$-rewrite to a collapsing redex, for otherwise it would rewrite to a normal form in one extra step. Hence the root symbol of every reduct of $t/q[\bullet]_{q'}$ belongs to $\mathcal{F}$. Since qq' is not the position of an $(\mathcal{S}_\alpha, \mathcal{G})$-needed redex in t, $t[\bullet]_{qq'} \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$. Since any normalizing $(\mathcal{S}_\alpha, \mathcal{G})$-rewrite sequence must contain a rewrite step at a position above q, we may write $t[\bullet]_{qq'} \to^*_{\mathcal{S}_\alpha, \mathcal{G}} C[\Delta'] \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$ such that Δ' is the first redex above position q. Since $\mathsf{root}(\Delta') \in \mathcal{F}'$, the subterm $t/q[\bullet]_{q'}$ of $t[\bullet]_{qq'}$ does not contribute to the creation of Δ' and hence $t[\bullet]_q \to^*_{\mathcal{S}_\alpha, \mathcal{G}} C[\Delta'']$ with $\Delta'' \approx \Delta'$. Lemma B.1.2 yields $C[\Delta''] \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$ and thus $t[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$.

2. Suppose that $q \notin P$. Since Θ is flat, q cannot be below a position in P. It follows by orthogonality that q is also a redex position in t. Since t is an $(\mathcal{S}_\alpha, \mathcal{G})$-free term, $t[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G})$.

As $t' \in \mathcal{T}(\mathcal{F}')$, we have $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G}, \mathcal{F}')$. Since $\mathsf{WN}_\bullet(\mathcal{S}_\alpha, \mathcal{G}, \mathcal{F}') \subseteq \mathsf{WN}_\bullet(\mathcal{R}'_\alpha, \mathcal{G}, \mathcal{F}') = \mathsf{WN}_\bullet(\mathcal{R}'_\alpha, \mathcal{F}')$ by Lemmata B.1.8 and B.1.5, we obtain $t'[\bullet]_q \in \mathsf{WN}_\bullet(\mathcal{R}'_\alpha, \mathcal{F}')$, contradicting the assumption that q is the position of an $(\mathcal{R}'_\alpha, \mathcal{F}')$-needed redex in t'. □

Proof. PROOF of Theorem 4.2.7. Let $\mathcal{C} = \mathcal{F}_\mathcal{C} \cap \mathcal{F}'_\mathcal{C}$ be the set of common constructors. Let $\mathcal{H} = \mathcal{F} \cup \mathcal{C}$ and $\mathcal{H}' = \mathcal{F}' \cup \mathcal{C}$. According to Theorem 4.1.7 the systems $(\mathcal{R}, \mathcal{H})$ and $(\mathcal{R}', \mathcal{H}')$ belong to CBN_α. Suppose to the contrary that $(\mathcal{S}, \mathcal{G}) \notin \mathsf{CBN}_\alpha$. (As before, $\mathcal{S} = \mathcal{R} \cup \mathcal{R}'$ and $\mathcal{G} = \mathcal{F} \cup \mathcal{F}'$.) According to Lemma B.0.6 there exists a term $t \in \mathsf{WNR}(\mathcal{S}_\alpha, \mathcal{G})$ without $(\mathcal{S}_\alpha, \mathcal{G})$-needed redex. We assume without loss of generality that $\mathsf{root}(t) \in \mathcal{F}_\mathcal{D}$. Let s be an external normal form of $(\mathcal{R}, \mathcal{F})$. Let t' be the term obtained from t by replacing every $\mathcal{G} \setminus \mathcal{H}$-alien by s. Note that $t' \in \mathcal{T}(\mathcal{H})$. Because $\mathcal{R}_\alpha$ is left-linear and $\mathcal{R}'_\alpha$ lacks both collapsing and constructor-lifting rules, contractions in the $\mathcal{G} \setminus \mathcal{H}$-aliens of t cannot create a redex in the non-alien part of t. Since t is not root-stable, the latter exists and thus t' contains a redex as well. Because $(\mathcal{R}, \mathcal{H}) \in \mathsf{CBN}_\alpha$, t' must contain an $(\mathcal{R}_\alpha, \mathcal{H})$-needed redex Δ, say at position p. Because s is an external normal form, Δ is also a redex in t and hence there exists a rewrite sequence $t[\bullet]_p \to^+_{\mathcal{S}_\alpha, \mathcal{G}_\bullet} U$ with $U \in \mathsf{NF}(\mathcal{R}_\bullet, \mathcal{G})$. If we replace in this rewrite sequence every $\mathcal{G} \setminus \mathcal{H}$-alien by s, we obtain a rewrite sequence $t'[\bullet]_p \to^+_{\mathcal{R}_\alpha, \mathcal{H}_\bullet} U'$. Because s does not unify with a proper non-variable subterm of a left-hand side of a rewrite rule, it follows that $U' \in \mathsf{NF}(\mathcal{R}_\bullet, \mathcal{H})$. Hence Δ is not an $(\mathcal{R}_\alpha, \mathcal{H})$-needed redex in t', yielding the desired contradiction. □

Bibliography

[AC75] A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[BAOE76] L. Beckman, Haraldsson A., Oskarsson Ö., and Sandewall E. A partial evaluator, and its use as a programming tool. *Artificial Intelligence*, 7(4):319–357, 1976.

[Ber78] G. Berry. Stable models of typed lambda-calculii. In *Proc. 5 th ICALP*, 1978.

[BMS81] R. M. Burstall, D. B. MacQueen, and D. T. Sannella. Hope: An experimental applicative language. Technical Report CSR-62-80, Computer Science Dept, Univ. of Edinburgh, 1981.

[Bon89] A. Bondorf. A self-applicable partial evaluator for term-rewriting systems. In *International Joint Conference on the Theory and Practice of Software Development (TAPSOFT)*, volume 2. Springer-Verlag, 1989.

[CDG+02] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2002. Draft, available from `www.grappa.univ-lille3.fr/tata/`.

[CDGV94] J.L. Coquidé, M. Dauchet, R. Gilleron, and S. Vágvölgyi. Bottom-up tree pushdown automata: Classification and connection with rewrite systems. *Theoretical Computer Science*, 127:69–98, 1994.

[CG90] J.L. Coquidé and R. Gilleron. Proofs and reachability problem for ground rewrite systems. In *Proceedings of the 6th International Meeting of Young Computer Scientists*, volume 464 of *Lecture Notes in Computer Science*, pages 120–129, 1990.

[Com00] H. Comon. Sequentiality, monadic second-order logic and tree automata. *Information and Computation*, 157:25–51, 2000.

[DHLT90] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewriting systems and of other related term rewriting systems. *Information and Computation*, 88:187–201, 1990.

[DM97] I. Durand and A. Middeldorp. Decidable call by need computations in term rewriting (extended abstract). In *Proceedings of the 14th International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 4–18, 1997.

[DM98] I. Durand and A. Middeldorp. On the complexity of deciding call-by-need. Technical Report 1194-98, LaBRI, Université de Bordeaux I, 1998.

[DM01] I. Durand and A. Middeldorp. On the modularity of deciding call-by-need. In *Proceedings of the Internation Conference on Foundations of Software Science and Computation Structures*, volume 2030 of *Lecture Notes in Computer Science*, pages 199–213, 2001.

[DM05] I. Durand and Aart Middeldorp. Decidable call-by-need computations in term rewriting. *Information and Computation*, 196:95–126, 2005.

[Don70] J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[DS93] I. Durand and B. Salinier. Constructor equivalent term rewriting systems. *Information Processing Letters*, 47:131–137, 1993. Also Technical Report LaBRI 92–58.

[DS94] I. Durand and B. Salinier. Constructor equivalent term rewriting systems are strongly sequential: a direct proof. *Information Processing Letters*, 52:137–145, 1994. Also Technical Report LaBRI 93–20.

[DT85] M. Dauchet and S. Tison. Decidability of confluence for ground term rewriting systems. In *Proceedings of the 1st International Conference on Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 80–89, 1985.

[Dur86] Irène Durand. *Un modèle d'interprétation réparti pour une architecture multiprocesseur Prolog.* PhD thesis, Université Paul Sabatier, Toulouse, France, October 1986. Doctorat d'Université (in French).

[Dur94a] I. Durand. Bounded, strongly sequential and forward-branching term rewriting systems. *Journal of Symbolic Computation*, 18:319–352, 1994.

[Dur94b] I. Durand. Bounded, strongly sequential and forward-branching term rewriting systems. *Journal of Symbolic Computation*, 18:319–352, 1994.

[Dur02] I. Durand. Autowrite: A tool for checking properties of term rewriting systems. In *Proceedings of the 13th International Conference on Rewriting Techniques and Applications*, volume 2378 of *Lecture Notes in Computer Science*, pages 371–375, 2002.

[Dur04] I. Durand. Autowrite: A tool for term rewrite systems and tree automata. In *Proceedings of the Workshop on Rewriting Strategies*, pages 5–14, Aachen, June 2004.

[Dur05] I. Durand. Autowrite: A tool for term rewrite systems and tree automata. *Electronics Notes in Theorical Computer Science*, 124:29–49, 2005.

[GHWZ05] A. Geyser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[GKMD90] Mark E. Giuliano, Madhur Kohli, Jack Minker, and Irène Durand. Prism: A testbed for parallel control. In V. Kumar, P.S. Gopalakrishnan, and L.N. Kanal, editors, *Parallel Algorithms for Machine Intelligence and Vision*. Springer Verlag, 1990.

[GT01] T. Genêt and V. Viet Triem Tong. Reachability analysis of term rewriting systems with timbuk. In *Proc. 8th LPAI*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 691–702. Springer-Verlag, 2001.

[Har77] A. Haraldsson. *A Program Manipulation System Based on Partial Evaluation*. PhD thesis, Department of Mathematics, Linköping University, Linköping, 1977.

[HL79] G. Huet and J.-J. Lévy. Computations in non-ambiguous linear term rewriting systems. Technical Report 359, INRIA, 1979.

[HL91] G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, I and II. In *Computational Logic, Essays in Honor of Alan Robinson*, pages 396–443. The MIT Press, 1991.

[Jac96a] F. Jacquemard. *Automates d'Arbres et Réécriture de Termes*. PhD thesis, Université Paris-Sud, 1996.

[Jac96b] F. Jacquemard. Decidable approximations of term rewriting systems. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 362–376, 1996.

[JS95] J.-P. Jouannaud and W. Sadfi. Strong sequentiality of left-linear overlapping rewrite systems. In *Proceedings of the 4th International Workshop on Conditional Term Rewriting Systems*, volume 968 of *Lecture Notes in Computer Science*, pages 235–246, 1995.

[JSS85] N. D. Jones, P. Sestoft, and H. Sóndergaard. An experiment in partial evaluation: The generation of a compiler generator. Technical report, Institute of Datalogy, University of Copenhagen, Copenhagen, 1985.

[Ken95] R. Kennaway. A conflict between call-by-need computation and parallelism. In *Proceedings of the 4th International Workshop on Conditional Term Rewriting Systems*, volume 968 of *Lecture Notes in Computer Science*, pages 247–261, 1995.

[KKSdV96] R. Kennaway, J.W. Klop, R. Sleep, and F.-J. de Vries. Comparing curried and uncurried rewriting. *Journal of Symbolic Computation*, 21(1):15–39, 1996.

[Klo92] J.W. Klop. Term rewriting systems. In *Handbook of Logic in Computer Science, Vol. 2*, pages 1–116. Oxford University Press, 1992.

[KM91] J.W. Klop and A. Middeldorp. Sequentiality in orthogonal term rewriting systems. *Journal of Symbolic Computation*, 12:161–195, 1991.

[KP78] G. Kahn and G. Plotkin. Domaines concrets. Technical Report 336, IRIA, 1978.

[Mid90] A. Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Vrije Universiteit, Amsterdam, 1990.

[Mid97] A. Middeldorp. Call by need computations to root-stable form. In *Proceedings of the 24th Annual Symposium on Principles of Programming Languages*, pages 94–105. ACM Press, 1997.

[NST95] T. Nagaya, M. Sakai, and Y. Toyama. NVNF-sequentiality of left-linear term rewriting systems. In *Proceedings of the Workshop on Theory of Rewriting Systems and its Applications*, RIMS Technical Report 918, University of Kyoto, pages 109–117, 1995.

[NT02] T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Information and Computation*, 178(2):499–514, 2002.

[O'D77] M.J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*. Springer-Verlag, 1977.

[O'D85] M.J. O'Donnell. *Equational Logic as a Programming Language*. Fondations of Computing. MIT Press, 1985.

[Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.

[Oya93] M. Oyamaguchi. NV-sequentiality: A decidable condition for call-by-need computations in term rewriting systems. *SIAM Journal on Computation*, 22:114–135, 1993.

[Reb93] S. A. Rebelsky. *Lazy Term-based Communication: Issues, Observations, and Applications*. PhD thesis, University of Chicago, Chicago, Illinois, 1993.

[Ros73] B. K. Rosen. Tree-manipulating systems and church-rosser theorems. *Journal of the ACM*, 20:160–187, 1973.

[Sal95] B. Salinier. *Simulation de systèmes de réécriture de termes par des systèmes constructeurs*. PhD thesis, Université Bordeaux I, 1995.

[SS90] David J. Sherman and Robert I. Strandh. Partial evaluation of intermediate code from equational programs. Technical Report R.G. 06.90, Greco de Programmation du CNRS, 1990. Also Technical Report 90-029, University of Chicago Department of Computer Science.

[SS96] B. Salinier and R. Strandh. Efficient simulation of forward-branching systems. *Journal of Symbolic Computation*, 1996.

[SS97] B. Salinier and R. Strandh. Simulating forward-branching systems with constructor systems. In *Proceedings of the Colloquium on Trees in Algebra and Programming (CAAP) of the 7th International Joint Conference on Theory and Practice of Software Development (TAPSOFT)*, volume 1214 of *Lecture Notes in Computer Science*, pages 153–164, April 1997.

[SSD91] David J. Sherman, Robert I. Strandh, and Irène Durand. Optimization of equational programs using partial evaluation. In *Proceedings of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 72–82, 1991.

[STFK02] H. Seki, T. Takai, Y. Fujinaka, and Y. Kaji. Layered transducing term rewriting system and its recognizability preserving property. In *Proceedings of the 13th International Conference on Rewriting Techniques and Applications*, volume 2378 of *Lecture Notes in Computer Science*, pages 98–113, 2002.

[Str88] R. I. Strandh. *Compiling Equational Programs into Efficient Machine Code*. PhD thesis, Johns Hopkins University, Baltimore, Maryland, 1988.

[Str89] R. Strandh. Classes of equational programs that compile into efficient machine code. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 449–461, 1989.

[Tha85] S. Thatte. On the correspondence between two classes of reduction systems. *Information Processing Letters*, 20:83–85, 1985.

[Tha87] S. Thatte. A refinement of strong sequentiality for term rewriting with constructor. *ic*, 72:46–65, 1987.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B*, pages 133–191. North-Holland, 1990.

[TKS00] T. Takai, Y. Kaji, and H. Seki. Right-linear finite path overlapping term rewriting systems effectively preserve recognizability. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications*, volume 1833 of *Lecture Notes in Computer Science*, pages 246–260, 2000.

[Toy92] Y. Toyama. Strong sequentiality of left-linear overlapping term rewriting systems. In *Proceedings of the 7th Annual Symposium on Logic in Computer Science*, pages 274–284, 1992.

[TSvEP93a] Y. Toyama, S. Smetsers, M. van Eekelen, and R. Plasmeijer. The functional strategy and transitive term rewriting systems. In *Term Graph Rewriting: Theory and Practice*, pages 61–75. Wiley, 1993.

[TSvEP93b] Yoshihito Toyama, Sjaak Smetsers, Marko van Eekelen, and Rinus Plasmeijer. The functional strategy and transitive term rewriting systems. In M.J. Plasmeijer M.R. Sleep and M.C.J.D. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, pages 61–75. Wiley Professional Computing, 1993.

[TW68] J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2:57–81, 1968.

[Wad88] P. Wadler. Deforestation: Transforming programs to eliminate trees. In *Second European Symposium on Programming*. Springer-Verlag, 1988.

[Wal98] J. Waldmann. Rx: an interpreter for rational tree languages. `http://www.informatik.uni-leipzig.de/~joe/rx/`, 1998.

Appendix C

Indexes

Notation index

General index