Sequentiality, Monadic Second-Order Logic and Tree Automata

Hubert Comon*
Bat. 490, Université de Paris Sud
91405 ORSAY cedex, France.
E-mail {Hubert.Comon}@lri.fr.

*This research was partly supported by the ESPRIT working group CCL

Proposed running head: "Sequentiality and Tree Automata"

Author to whom proofs should be sent:

Hubert Comon
Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan
61, Avenue du Président Wilson
94235 Cachan cedex
France

Abstract

Given a term rewriting system R and a normalizable term t, a redex is *needed* if in any reduction sequence of t to a normal form, this redex will be contracted. Roughly, R is *sequential* if there is an optimal reduction strategy in which only needed redexes are contracted. More generally, G. Huet and J.-J. Lévy define in [9] the sequentiality of a predicate P on partially evaluated terms. We show here that the sequentiality of P is definable in SkS, the monadic second-order logic with k successors, provided P is definable in SkS. We derive several known an new consequences of this remark: 1–*strong sequentiality*, as defined in [9], of a left linear (possibly overlapping) rewrite system is decidable, 2– *NV-sequentiality*, as defined in [17] is decidable, even in the case of overlapping rewrite systems 3– *sequentiality* of any linear shallow rewrite system is decidable.

Then we describe a direct construction of a tree automaton recognizing the set of terms that do have needed redexes, which, again, yields immediate consequences: 1– Strong sequentiality of possibly overlapping linear rewrite systems is decidable in EXPTIME, 2– For strongly sequential rewrite systems, needed redexes can be read directly on the automaton.

1 Introduction

Besides confluence, there are two important issues concerning non-terminating computations in term rewriting theory. One is to find a *normalizing reduction strategy*, which has been investigated in, e.g., [16, 11, 1]. The other is to find an *optimal reduction strategy*, for which only *needed redexes* are contracted. This question was first investigated by Huet and Lévy in 1978 [9]. They call *sequential* a rewrite system for which there exists such an optimal strategy. We focus here on the latter issue.

A typical example is the "parallel or", whose definition contains the two rules $\top \vee x \to \top$ and $x \vee \top \to \top$. Given an expression $e_1 \vee e_2$, which of e_1 and e_2 should be evaluated first? If e_1 is tried first, its evaluation may be unnecessary because e_2 evaluates to $\top$, and the whole expression can be reduced to $\top$. Hence, this strategy is not optimal. Evaluating e_2 first is not optimal either: there is no optimal (sequential) reduction strategy for the "parallel or".

Given a term rewriting system R, can we decide whether R is *sequential* ? In case it is, is it possible to compute (and compile) an optimal strategy? These questions have been addressed in several papers, starting with [9]. Unfortunately, the sequentiality of R is in general undecidable. In their landmark paper, Huet and Lévy introduce a sufficient criterion: *strong sequentiality*, and show that this property is decidable for *orthogonal term rewriting systems*, in which left hand sides do not overlap nor contain repeated occurrences of a same variable. The original proof is quite intricate. J.-W. Klop and A. Middeldorp [14] give a simpler proof to the price of an increased complexity. The case of linear, possibly overlapping rewrite systems was considered first by Toyama [24] and later shown decidable by Jouannaud and Sadfi [10]. M. Oyamaguchi defines *NV-sequentiality* a property intermediate between sequentiality and strong sequentiality, which is also decidable for orthogonal rewrite systems [17].

In this paper we use another quite simple approach, though less elementary: we show that the sequentiality of P is definable in SkS (resp. WSkS), the second-order monadic logic with k successors, provided that P is definable in SkS (resp. WSkS). It allows to easily derive all aforementioned decidability results. Relying on automata theory, the decidability of strong sequentiality (resp. NV-sequentiality) of possibly overlapping left linear rewrite systems becomes straightforward. This sheds new light on which properties of rewrite systems are indeed necessary in proving (NV-, strong-) sequentiality. Then, it becomes possible to derive new decidability results, for example NV-sequentiality for overlapping left linear rewrite systems or sequentiality of shallow linear rewrite systems. We may also add a *sort discipline* to the rewrite systems without loosing decidability.

This method has however several drawbacks: first, the complexity of SkS is non-ELEMENTARY, which is far too complex in general for "effective" methods. Second, for non-left linear rewrite system, the reducibility predicate is not expressible in SkS. Hence we cannot derive that strong sequentiality of R is expressible in SkS in such a case. Last, but not least, even if we know that the formula expressing the sequentiality of R is valid, how do we effectively find needed redexes in a term?

In order to answer these questions, we construct directly a tree automaton which accepts all terms that have a needed redex. By the well-known correspondence between WSkS and finite tree automata (see e.g. the survey [23]), we know in advance that such an automaton exists. Here, we show that it can be constructed in exponential time (for k fixed). This has several consequences. First, deciding strong sequentiality of any left linear rewrite system is in EXPTIME, since it reduces to an emptiness problem for tree automata, which can be decided in polynomial time. Then, the automaton which accepts all terms that have a needed redex yields directly the algorithm for searching needed redexes in a term.

There are still many issues to be investigated with profit in this framework: is strong sequentiality decidable for any (possibly non-linear) rewrite systems? Though automata with constraints [2, 3] cannot be used directly, we might consider some tree automata inspired by these definitions. What is the exact complexity of all decision questions in this area? We have only shown an EXPTIME inclusion. However there is no evidence that this is the best we can do. Also, what happens in the case of orthogonal rewrite systems? The automata should have a particular form, from which it might be possible to deduce more efficient procedures. Finally, we do not show how to *compile* an optimal reduction strategy, avoiding any backtrack in the input term, as done in [9]. Again, this should be possible from the tree automaton. Finally, other (sequential) reduction strategies as in [1] should also be investigated within this framework.

The paper is organized as follows: section 2 gives the definitions of an index and sequentiality and we recall the necessary background on SkS and tree automata. In section 3, we show how to express the sequentiality of a predicate P in SkS and apply this result to rewrite systems in section 4. In section 5 we construct directly the automaton accepting all terms that have an index (using the characterization of [14]) and derive extensions as well as complexity results. We also explain how an index search can be read on the automaton.

2 Basic Definitions

2.1 Terms

T is the set of terms built over a fixed alphabet $\mathcal{F}$ of function symbols. Each $f \in \mathcal{F}$ comes with its *arity* $a(f)$, a nonnegative integer. Terms may also be viewed as labeled trees, i.e. mappings from a finite prefix-closed subset of words of positive integers (the *positions* in the tree) into $\mathcal{F}$, in such a way that the successors of a position p are exactly the strings $p \cdot i$ for $1 \leq i \leq a(f)$ when p is labeled with f. We will use the notations of [7]: $t|_p$ is the *subterm* at position p, $t[u]_p$ is the term obtained by replacing $t|_p$ with u. $\mathcal{F}$ is assumed to be finite. [1]

$\mathcal{T}_\Omega$ is the set of terms obtained by augmenting the set $\mathcal{F}$ of function symbols with a new constant Ω (which stands intuitively for "unevaluated terms"). We assume that terms in $\mathcal{T}_\Omega$ always contain at least one occurrence of Ω. Such a set of terms is classically considered as the set of terms which are partially evaluated, i.e. terms in T which are "cut" on some branches.

Definition 1 *Let $t, u \in T \cup \mathcal{T}_\Omega$, $t \sqsubseteq u$ iff u can be obtained from t by replacing some occurrences of Ω in t with terms in $\mathcal{T}_\Omega$.*

[1] Note that if one wants to consider terms with possibly infinitely many "variables" (actually constants, but we use the standard terminology), it is always possible to represent the variables $x_0, x_1, \ldots, x_n, \ldots$ using an additional constant x and an additional unary function symbol s; they will be respectively represented by $x, s(x), \ldots, s(s(\ldots(s(x))\ldots)), \ldots$. In such a case, T is a regular subset of the set of all terms, but this does not cause any additional problem, as we will see later in the general case of sorted terms. The status of variables in T (or $\mathcal{T}_\Omega$) is different from the status of the variables in the rewrite system: the former are actually considered as constants along the evaluation process, while the latter may be instanciated since several distinct instances of the rules can be used.

$s \sqsubseteq t$ intuitively means that "t is more evaluated than s".

2.2 Sequentiality

Definition 2 (index,[9]) *Let P be a predicate on $\mathcal{T}_\Omega \cup T$. Let $t \in \mathcal{T}_\Omega$ and $p \in Pos(t)$. p is an* index *of P in t iff $t|_p = \Omega$ and*

$$\forall u \in \mathcal{T}_\Omega \cup T, \;\; (t \sqsubseteq u \wedge P(u) = true) \Rightarrow u|_p \neq \Omega$$

The set of indexes of a term $t \in \mathcal{T}_\Omega$ (which were also called *needed redexes* in the introduction; there is a confusion here between needed redexes and positions of Ω, which is justified in section 4) is written Index(t). Intuitively, p is an index for P in t if, for all successful evaluations of t (the predicate P becomes true), the term at position p has been evaluated.

Definition 3 (sequentiality,[9]) *A predicate P on $\mathcal{T}_\Omega \cup T$ is* sequential *if*

$$\begin{aligned}\forall t \in \mathcal{T}_\Omega \cup T, (\exists u \in \mathcal{T}_\Omega \cup T, P(u) = true \wedge t \sqsubseteq u) \\ \Rightarrow (P(t) = true \vee \exists p \in Pos(t), p \in Index(t))\end{aligned}$$

Intuitively, P is sequential if, for every partially evaluated term t such that P is false and P becomes true for some further evaluation of t, then there is an index of P in t.

2.3 Term Rewriting

$\mathcal{X}$ is an infinite set of constant function symbols called *variables* and the set of terms build on $\mathcal{F} \cup \mathcal{X}$ is traditionally written $\mathcal{T}(\mathcal{F}, \mathcal{X})$. For any $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, Var$(s)$ is the set of variables occurring in s. Substitutions are mappings from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which are extended into endomorphisms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We use the postfix notation for substitution applications.

A *term rewriting system* is a (finite) set of pairs of terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, each pair (s, t) is written $s \to t$ (we do not require Var$(t) \subseteq$ Var(s)). A term t *rewrites to* s through a rewrite system R, which is written $t \underset{R}{\longrightarrow} s$ if there is a position p in t, a substitution σ and a rule $l \to r \in R$ such that $t|_p = l\sigma$ and $s = t[r\sigma]_p$. Rewriting using zero or many single rewriting steps, i.e. the reflexive transitive closure of $\underset{R}{\longrightarrow}$ is written $\overset{*}{\underset{R}{\longrightarrow}}$.

2.4 Tree Automata

We recall here some basic definitions about tree automata (see e.g. [8]).

Definition 4 *A* finite (bottom-up) tree automaton *consists of a ranked alphabet $\mathcal{F}$, a finite set of states Q, a subset Q_f of final states and a set of transition rules of the form $f(q_1, \ldots, q_n) \to q$ where $f \in \mathcal{F}$, $n = a(f)$ and $q_1, \ldots, q_n, q \in Q$ or $q \to q'$ where $q, q' \in Q$ (the latter transition rules are called* ϵ-transitions*). A tree automaton* accepts *t if t can be rewritten to a final state using the transition rules (see e.g. [8] for more details).*

Definition 5 *The* language *accepted by a tree automaton $\mathcal{A}$ is the set of terms t which are accepted by $\mathcal{A}$.*

A set L of trees is recognizable *when there is a tree automaton $\mathcal{A}$ such that L is the language accepted by $\mathcal{A}$.*

Definition 6 *A* run *of the automaton on a tree t is a mapping ρ from the positions of t into Q such that $\rho(p) = q, \rho(p \cdot 1) = q_1, \ldots \rho(p \cdot n) = q_n$ and $t(p) = f$ only if there is a transition rule $f(q_1, \ldots, q_n) \to q'$ and a sequence of ϵ-transitions from q' to q. A run ρ is* successful *if $\rho(\Lambda)$ is a final state. t is accepted by A iff there is a successful run of A on t.*

We will see in what follows several examples of recognizable sets of terms.

2.5 The logic (W)SkS

Missing definitions can be found in [23]. *Terms* of SkS are formed out of individual variables $(x, y, z, \ldots)$, the empty string Λ and right concatenation with $1, \ldots, k$. Atomic formulas are equations between terms, inequations $w < w'$ between terms or expressions "$w \in X$" where w is a term and X is a (second-order) variable. Formulas are built from atomic formulas using the logical connectives $\wedge, \vee, \Rightarrow, \neg, \ldots$ and the quantifiers $\exists, \forall$ of both individual and second-order variables. Individual variables are interpreted as elements of $\{1, \ldots, k\}^*$ and second-order variables as subsets of $\{1, \ldots, k\}^*$. Equality is the string equality and inequality is the strict prefix ordering. In the *weak* second-order monadic logic WSkS, second-order variables only range over *finite* sets. Finite union an finite intersection, as well as inclusion and equality of sets are definable in (W)SkS in an obvious way. Hence we may use these additional connectives in the following.

The most remarkable result is the decidability of SkS (a result due to M. O. Rabin, see e.g. [18, 23] for comprehensive surveys). The main idea of the proof is to associate each formula ϕ whose free variables are $X_1, \ldots, X_n$ with a (Rabin) tree automaton which accepts the set of n-tuples of trees (or sets of strings) that satisfy the formula. Then decidability follows from closure and decidability properties of the corresponding class of tree languages. We only use here the weak case, in which only finite state tree automata are used. We will extensively use the following without any further mention:

Theorem 7 (Thatcher and Wright, 1969) *A set of finite trees is definable in WSkS iff it is recognized by a finite tree automaton.*

Formally, this correspondence needs to define a term in WSkS. We recall below how it can be done.

3 Relationship between Sequentiality and Recognizability

Let k be the maximal arity of a function symbol in $\mathcal{F}$ and n be the cardinal of $\mathcal{F}$. A term t is represented in WSkS using $n+2$ set variables X, X_Ω and $X_f, f \in \mathcal{F}$ (which will be written $\vec{X}$ in the following). X will be the set of positions of t and X_Ω and each X_f will be the sets of positions that are labeled with the corresponding function symbol. We express in WSkS that some $n+2$-tuple of finite sets of words are indeed encoding a term, which can be achieved using the formula

$$\begin{array}{ll} \mathrm{Term}(\vec{X}) \stackrel{\mathrm{def}}{=} & \\ & X = X_\Omega \cup \bigcup_{i=1}^{n} X_{f_i} \\ \wedge & \bigwedge_{i \neq j} (X_{f_i} \cap X_{f_j} = \emptyset \wedge X_\Omega \cap X_{f_i} = \emptyset) \\ \wedge & \forall x \in X, \forall y < x, y \in X \\ \wedge & \bigwedge_{f \in \mathcal{F} \cup \{\Omega\}} \forall x \in X_f, \bigwedge_{l=1}^{a(f)} x \cdot l \in X \quad \bigwedge_{l=a(f)+1}^{k} x \cdot l \notin X \end{array}$$

In this setting, it is quite easy to express the sequentiality of P in (W)SkS as shown by the following lemmas.

Lemma 8 $\sqsubseteq$ *is definable in WSkS.*

Proof: Assume that t, u are represented by $\vec{X}$ and $\vec{Y}$ respectively. Then $t \sqsubseteq u$ iff

$$X \subseteq Y \wedge \bigwedge_{f \in \mathcal{F}, f \neq \Omega} X_f \subseteq Y_f \qquad \Box$$

Lemma 9 *Let P be a predicate on $\mathcal{T}_\Omega \cup T$ which is definable in (W)SkS. Then the set of terms in $\mathcal{T}_\Omega$ which have an index w.r.t. P is definable in (W)SkS.*

Proof: Let $\phi(\vec{X})$ be the definition of P in (W)SkS. Then the set of terms which have an index is defined by translating definition 2:

$$\begin{array}{l} \mathrm{Index}(\vec{X}) \stackrel{\mathrm{def}}{=} \mathrm{Term}(\vec{X}) \wedge \exists x \in X.\, x \in X_\Omega \wedge \forall \vec{Y}. \\ \qquad (\mathrm{Term}(\vec{Y}) \wedge \vec{X} \sqsubseteq \vec{Y} \wedge \phi(\vec{Y})) \Rightarrow x \notin Y_\Omega \qquad \Box \end{array}$$

Theorem 10 *If P is definable in (W)SkS, then the sequentiality of P is decidable.*

Proof: Using the previous lemma and assuming that P is defined by $\phi(\vec{X})$, P is sequential iff the following formula holds:

$$\begin{array}{l} \forall \vec{X}.(\mathrm{Term}(\vec{X}) \wedge \exists \vec{Y}.\mathrm{Term}(\vec{Y}) \wedge \phi(\vec{Y}) \wedge \vec{X} \sqsubseteq \vec{Y}) \\ \qquad\qquad \Rightarrow (\phi(\vec{X}) \vee \mathrm{Index}(\vec{X})) \end{array}$$

which is a translation of definition 3. Then we conclude using Rabin's theorem [19, 18]. $\Box$

4 Application to term rewriting systems

In this section, we show how to apply theorem 10 to various sequentiality results for term rewriting. We assume here the reader familiar with term rewriting (see e.g. [7] for missing definitions). We will say in particular that a term t is *linear* if each variable occurs at most once in t (A term is *shallow* if it is a variable or if all its variables occur at depth 1. A rewrite system R is *left linear* (resp. *linear*, resp. *shallow*) if all its left hand sides are linear (resp. all left and right hand sides are linear, resp. all left and right hand sides are shallow). Two terms t_1, t_2 are *similar* if there is a renaming of the variables of t_1 which yields t_2.

This section is organized as follows: we first state the basic definitions of sequentiality and strong sequentiality in the case of rewrite systems in section 4.1. Then, we establish basic properties about the reducibility predicate in section 4.2. All following proofs look similar. We try to factorize here as much as possible the common patterns. Then we show in section 4.3 that the so-called "NVNF-sequentiality" is decidable for left linear (possibly overlapping) rewrite systems. This is a new result which is an application of theorem 10: we show that an appropriate predicate is definable in WSkS. We give actually two proofs of the latter property: one is a 5 lines proofs, relying on previous results by Dauchet et al and the other is a 1 page direct construction.

The decidability of NVNF sequentiality implies in particular the decidability of strong sequentiality of possibly overlapping left linear rewrite systems (a result proved in [10]).

Then we show in section 4.4 that sequentiality (which is in general undecidable) is decidable for shallow rewrite system, again as an application of theorem 10.

4.1 Strong sequentiality of left linear term rewriting systems

Let $N_{\mathcal{R}}$ be the predicate symbol on $\mathcal{T}_\Omega \cup T$ which holds true iff t has a normal form (w.r.t. $\mathcal{R}$) belonging to T.

Definition 11 ([9]) *A term rewriting system $\mathcal{R}$ is* sequential *if the predicate $N_{\mathcal{R}}$ is sequential.*

This captures the intuitive notion sketched in introduction: when $\mathcal{R}$ is sequential, then there is an optimal reduction strategy. Since sequentiality of $\mathcal{R}$ is undecidable in general, a sufficient condition for sequentiality (called strong sequentiality) has been introduced in [9].

Let $\mathrm{Red}_{\mathcal{R}}$ be the predicate symbol on $\mathcal{T}_\Omega \cup T$ which holds true iff t is reducible by $\mathcal{R}$ and $NF_{\mathcal{R}}$ be the set of irreducible terms in T. If L is a set of terms, we also write $\xrightarrow[L]{?}$ the binary relation on $T \cup \mathcal{T}_\Omega$ defined by $s \xrightarrow[L]{?} t$ iff there is a position p in s, a term $l \in L$, a substitution σ and a term $u \in T \cup \mathcal{T}_\Omega$ such that $s|_p = l\sigma$ and $t = s[u]_p$. This is the usual definition of rewriting, except that we do not consider right hand sides: the left hand side can be replaced with any term u.

Let $\mathrm{N?}_{\mathcal{R}}$ be the predicate on $T \cup \mathcal{T}_\Omega$ which is true on t iff t has a normal form in T for the relation $\xrightarrow[L]{?}$ where L is the set of left hand sides of $\mathcal{R}$. Of course, $N_{\mathcal{R}} \subseteq \mathrm{N?}_{\mathcal{R}}$.

Definition 12 ([9]) *A term rewriting system $\mathcal{R}$ is* strongly sequential *if the predicate $N?_{\mathcal{R}}$ is sequential.*

Since $N_{\mathcal{R}} \subseteq \mathrm{N?}_{\mathcal{R}}$, an index for $\mathrm{N?}_{\mathcal{R}}$ is also an index for $N_{\mathcal{R}}$, hence any strongly sequential rewrite system is also sequential. (But the converse is false).

We will show, as a consequence of lemma 21 that $\mathrm{N?}_{\mathcal{R}}$ is actually recognized by a finite tree automaton. We will also give a direct construction of the automaton which accepts the terms having an index (resp. no index) in section 5. As a consequence, we have:

Corollary 13 *The strong sequentiality of left linear (possibly overlapping) rewrite systems is decidable.*

Proof: This is a consequence of lemma 24 and theorem 10 since recognizable sets of terms are definable in WSkS [22]. □

A result which is also known from [10].

4.2 The reducibility and normal form predicates

In the following constructions we will basically compute fixed points starting from the set of reducible (resp. irreducible) terms. Let us therefore state and prove some basic (well-known) results about $\mathrm{Red}_{\mathcal{R}}$.

Lemma 14 *When $\mathcal{R}$ is left linear, $Red_{\mathcal{R}}$ is recognized by a finite bottom-up tree automaton.*

Proof: If $\mathcal{R}$ contains a rule whose left member is a variable, then the result is obvious; this case is discardedin the following. For each non-variable strict subterm u of a left hand side of a rule, consider a state q_u. In addition, we have a state q_r (the final state, or the state in which we know that the term is reducible) and the state $q_\top$ which accepts all terms. Then, to each

$u = f(u_1, \ldots, u_n)$, we associate the production rule $f(q_{u_1}, \ldots, q_{u_n}) \rightarrow q_u$ where q_{u_i} is understood as $q_\top$ when u_i is a variable. To each left hand side of a rule $l = f(t_1, \ldots, t_n)$, we associate the rule $f(q_{t_1}, \ldots, q_{t_n}) \rightarrow q_r$ and the states q_r are propagated: we have the rules $f(q_\top, \ldots, q_\top, q_r, q_\top, \ldots, q_\top) \rightarrow q_r$ for all function symbols f. Finally, if not already present, we add the rules $f(q_\top, \ldots, q_\top) \rightarrow q_\top$. $\square$

Note that this doesn't work for non-left linear rewrite systems because then $\mathrm{Red}_{\mathcal{R}}$ is not definable by a finite bottom-up tree automaton: we need disequality tests. Actually, adding the corresponding tests to the logic (W)SkS yields an undecidable logic.[2]

Another consequence of lemma 14 is the recognizability of the set of irreducible terms in $T(\mathcal{F})$: this is a consequence of the closure property of recognizable tree languages by complement. Let us show however an explicit construction of such an automaton, since we will re-use it for further analysis in the following.

Given two terms $s, t \in T(\mathcal{F}, \mathcal{X})$, we write $s \downarrow t$ for a most general instance (when it exists) of s and a renaming t' of t such that t' and s do not share variables. Given a left linear rewrite system $\mathcal{R}$, let $S(\mathcal{R})$ be the set of strict subterms of the left hand sides of $\mathcal{R}$, up to similarity, which we close under $\downarrow$. (This may yield an exponential number of terms in $S(\mathcal{R})$: one for each set of subterms of the lhs of $\mathcal{R}$). With each term t in $S(\mathcal{R})$, which is not an instance of a lhs of $\mathcal{R}$, we associate a state q_t. (we write q_x the state associated with all variables. We assume that q_x is in the set of states). Let $Q = \{q_t \mid t \in S(\mathcal{R})\} \cup \{q_r\}$ and Q_f be all states but q_r. Intuitively, all reducible terms will be accepted in q_r. The terms accepted in a state q_t will be all instances of t that are not instances of any $t\sigma$. More precisely, we

[2]This result can either be derived from undecidability of the emptiness for automata with constraints (Lille group, around 1980; the result is reported and proved in several papers), or from the undecidability of extensions of WSkS with the equal length predicate (several authors, reported in [23]).

consider the following set of production rules:

$$\begin{array}{lrcl}
\textbf{S1} & f(q_{t_1},\ldots,q_{t_n}) & \rightarrow & q_t \\
& & & \text{If } f(t_1,\ldots,t_n) \text{ is an instance of } t \\
& & & \text{and not an instance of some } t\sigma \neq t \text{ s.t. } q_{t\sigma} \in Q. \\
& & & \text{(In other words, } t \text{ is the maximal prefix of} \\
& & & f(t_1,\ldots,t_n) \text{ which belongs to } S(\mathcal{R})\text{)} \\
\textbf{S2} & f(q_{t_1},\ldots,q_{t_n}) & \rightarrow & q_r \\
& & & \text{If } f(t_1,\ldots,t_n) \text{ is an instance} \\
& & & \text{of some lhs of } R \\
\textbf{S3} & f(q_1,\ldots,q_n) & \rightarrow & q_r \\
& & & \text{If } q_r \in \{q_1,\ldots,q_n\}
\end{array}$$

Let us call $\mathcal{A}_{NF(\mathcal{R})}$ the above constructed automaton.

Example 15

$$R = \left\{ \begin{array}{rcl} h(x) & \rightarrow & g(f(a,a)) \\ f(x,a) & \rightarrow & a \\ f(g(a),x) & \rightarrow & f(x,g(a)) \end{array} \right.$$

The automaton $\mathcal{A}_{NF(R)}$ consists of

- The set of states $Q = \{q_a, q_{g(a)}, q_x, q_r\}$
- The set of final states $Q_f = \{q_a, q_{g(a)}, q_r\}$
- The Alphabet is supposed to contain exactly h, f, g, a.
- the production rules (assuming for simplicity that there are no additional function symbols besides h, f, g, a)

$$\begin{array}{rclrcl}
a & \rightarrow & q_a & g(q_a) & \rightarrow & q_{g(a)} \\
f(q_a,q_a) & \rightarrow & q_r & h(q_a) & \rightarrow & q_r \\
f(q_a,q_{g(a)}) & \rightarrow & q_x & f(q_{g(a)},q_a) & \rightarrow & q_r \\
f(q_{g(a)},q_{g(a)}) & \rightarrow & q_r & h(q_{g(a)}) & \rightarrow & q_r \\
g(q_{g(a)}) & \rightarrow & q_x & f(q_x,q_a) & \rightarrow & q_r \\
f(q_x,q_{g(a)}) & \rightarrow & q_x & f(q_x,q_x) & \rightarrow & q_x \\
f(q_a,q_x) & \rightarrow & q_x & f(q_{g(a)},q_x) & \rightarrow & q_r \\
h(q_x) & \rightarrow & q_r & g(q_x) & \rightarrow & q_x \\
f(q_r,q) & \rightarrow & q_r & f(q,q_r) & \rightarrow & q_r \\
g(q_r) & \rightarrow & q_r & h(q_r) & \rightarrow & q_r
\end{array}$$

where q stands for any state in Q. For instance, $a \to q_a$, $g(q_{g(a)}) \to q_x$ are rules obtained from the set **S1**, $f(q_a, q_a) \to q_r$ is a rule from **S2**, and $g(q_r) \to q_r$ is a rule from the third set **S3**.

Lemma 16 *$\mathcal{A}_{NF(\mathcal{R})}$ accepts the set of irreducible terms in $T(\mathcal{F})$. This automaton is deterministic and completely specified.*

<u>**Proof:**</u>

The automaton is deterministic since, assuming that $f(t_1, \ldots, t_n)$ is an instance of both t and u ($t, u \in S(\mathcal{R})$), then it is an instance of $t \downarrow u$, hence the only rule which can be applied to $f(q_{t_1}, \ldots, q_{t_n})$ is either $f(q_{t_1}, \ldots, q_{t_n}) \to q_r$ (when $f(t_1, \ldots, t_n)$ is an instance of a lhs of $\mathcal{R}$) or $f(q_{t_1}, \ldots, q_{t_n}) \to q_{u_1 \downarrow \ldots \downarrow u_m}$ if $\{u_1, \ldots u_m\}$ is the set of terms in $S(\mathcal{R})$ which are not instances of a lhs of R and such that $f(t_1, \ldots, t_n)$ is an instance of each u_i (and not an instance of a lhs).

The automaton is completely specified since every term is (at least) an instance of x. Then either one of its direct subterms is accepted in state q_r, or it is itself accepted in state q_r, or ther is a state q_t in which it is accepted.

We show by induction on the size of u that u is accepted in the state q_t iff u is not reducible, it is an instance of t and not an instance of some other $t\sigma \in S(\mathcal{R})$. If u is a constant, then either $u \in S(\mathcal{R})$ (in which case it is accepted in state q_r or q_u depending on its reducibility) or it is accepted in state q_x. Now, consider a term $t = f(t_1, \ldots, t_n)$. If some t_i is reducible, then it is accepted in state q_r by induction hypothesis and t is accepted in state q_r too. Otherwise, by induction hypothesis, for every i, t_i is accepted in state q_{u_i} s.t. $t_i = u_i \sigma_i$ and t_i is not an instance of any other $u_i \sigma$. Then either t is reducible, hence an instance of a lhs of a rule, and it is accepted in state q_r, or else, there is a rule $f(q_{u_1}, \ldots, q_{u_n}) \to q_u$ such that t is accepted in state q_u, $f(u_1, \ldots, u_n)$ is an instance of u and not an instance of any other $u\sigma$. Then, if $t = f(v_1, \ldots, v_n)\sigma$ for some σ, $t_i = v_i \sigma$ for all i. Hence, for all i, u_i is an instance of v_i, which implies that $f(v_1, \ldots, v_n)$ is an instance of u (assuming the disjointness of variables).

It follows that t is reducible iff it is accepted in the state q_r, hence it is irreducible iff it is accepted in another state, thanks to determinism and complete specification. $\square$

These constructions can be simplified for a particular class of rewrite system:

Lemma 17 *Assume that for any two strict subterms s, t of some left hand side(s) of $\mathcal{R}$, if s and t are unifiable, then either s is an instance of t or t is an instance of s. Then $Red_{\mathcal{R}}$ and $NF(\mathcal{R})$ are accepted by deterministic bottom-up tree automata with $O(|\mathcal{R}|)$ states.*

This is a direct consequence of the construction of $\mathcal{A}_{NF(\mathcal{R})}$: the set of states is $O(|\mathcal{R}|)$ in this case.

In the rest of this section, we are going to follow several times the following scheme (depending on the choice of $\mathcal{R}$): we prove first that the set $\{t \mid \exists u, t \xrightarrow[\mathcal{R}]{*} u, u \in NF_{\mathcal{R}}\}$ is recognizable. Then we may derive the recognizability of the predicate $N_{\mathcal{R}}$, hence the decidability of sequentiality, thanks to theorem 10. This last step can be factorized:

Definition 18 *A rewrite system $\mathcal{R}$* preserves regularity *if for any recognizable subset L of $T \cup \mathcal{T}_\Omega$, the set $\{s \in T \cup \mathcal{T}_\Omega \mid \exists t \in L, s \xrightarrow[\mathcal{R}]{*} t\}$ is recognizable.*

Lemma 19 *If $\mathcal{R}$ is left linear and preserves regularity, then $N_{\mathcal{R}}$ is recognizable and the sequentiality of $\mathcal{R}$ is expressible in WSkS.*

Proof:
It is sufficient to note that the set $\{u \in T \cup \mathcal{T}_\Omega \mid \exists t \in NF_{\mathcal{R}}, u \xrightarrow[\mathcal{R}]{*} t\}$ is recognizable, thanks to the left linearity of $\mathcal{R}$ and lemma 16. □

4.3 NVNF-sequentiality of left linear rewrite systems

Instead of approximating the rewrite system by forgetting about its right hand sides, as in the case of strong sequentiality, Oyamaguchi introduced in [17] a refined approximation, forgetting only the relationship between the variables of the left and right hand sides respectively. More precisely, if $\mathcal{R}$ is a rewrite system, we consider the rewrite system $\mathcal{R}_V$ where all occurrences of variables in the right hand sides have been replaced with new distinct variables. The strong sequentiality of $\mathcal{R}$ implies the sequentiality of $\mathcal{R}_V$ which in turn implies the sequentiality of $\mathcal{R}$. (All implications are strict). Oyamaguchi has shown in [17] the decidability, for orthogonal rewrite systems, of the predicate τ which is true on t when there is a $s \in T$ (i.e. without Ωs) such that $t \xrightarrow[\mathcal{R}_V]{*} s$. This does not correspond exactly to the sequentiality of $\mathcal{R}_V$ since s is not required to be in normal form. When s is moreover required to be in normal form, we find again the sequentiality of $\mathcal{R}_V$, which is called *NVNF-sequentiality* in [15]. In this latter paper, the authors show how to compute an index, assuming NVNF-sequentiality. It turns out that NVNF-sequentiality is again definable in WSkS (without the orthogonality assumption of [17]).

In what follows, we assume that no left hand side of R_V is a variable, but the constructions can be easily extended to this case.

Lemma 20 *For every left linear rewrite system R, R_V preserves regularity.*

Proof:

Let L be a recognizable subset of $T \cup \mathcal{T}_\Omega$ and $\mathcal{A}_0$ an automaton which accepts L. The idea of the proof is simple: we are going to start with $\mathcal{A}_0$ and "complete it backwards using the rules of R_V". For, we need to know wether we have an instance of a right or left member of R_V.

We consider an automaton $\mathcal{A}'_0$ whose states are the subterms of the right hand sides of R_V and such that the set of terms accepted in a state q_s is the set of instances of s. Finally we consider an automaton $\mathcal{A}''_0$ whose states are the strict subterms of the left hand sides of R_V and such that the set of terms accepted in q_s is the set of instances of s.

Our automaton $\mathcal{A}$ is constructed as follows: first consider the union of the automata $\mathcal{A}_0, \mathcal{A}'_0, \mathcal{A}''_0$: Q is the union of the three sets of states Q_0, Q'_0, Q''_0, Q_f is the set of final states of $\mathcal{A}_0$, the production rules are those of the three automata. At this stage, $\mathcal{A}$ accepts the terms that can be reduced in 0 or less rewriting steps into a term accepted by $\mathcal{A}_0$.

We saturate $\mathcal{A}$ with the following inference rules:

$$\frac{g(l_1, \dots, l_n) \to f(r_1, \dots, r_m) \in R_V \quad f(q_1^*, \dots, q_m^*) \to q^* \in P}{g(q''_{l_1}, \dots, q''_{l_n}) \to q^* \in P}$$

If, for every i there is an instance of r_i which is accepted in state q_i^* and q_i^* is either q_i or q'_i or q''_i. (Note that this condition is decidable as the set of instances of r_i is accepted by a finite tree automaton and by decision properties for tree automata).

$$\frac{g(l_1, \dots, l_n) \to x \in R_V \quad q^* \in Q}{g(q'_{l_1}, \dots, q'_{l_n}) \to q^* \in P}$$

If x is a variable and q^* is either q or q' or q''.

The saturation process does terminate since no new state is added.

We claim that the resulting automaton accepts the terms of $T \cup \mathcal{T}_\Omega$ that can be reduced to some term accepted by $\mathcal{A}_0$. For, we have two inclusions to prove.

Any term that can be reduced to a term in L is accepted by $\mathcal{A}$ We prove that $\xrightarrow[R_V]{} \circ \xrightarrow[\mathcal{A}]{*} \subseteq \xrightarrow[\mathcal{A}]{*}$. This is sufficient since, in such a case, if $t \xrightarrow[R_V]{*} u$ and u is accepted by $\mathcal{A}_0$, then $t \xrightarrow[R_V]{*} \circ \xrightarrow[\mathcal{A}]{*} q_f$ for some final state q_f of $\mathcal{A}_0$ and then $t \xrightarrow[\mathcal{A}]{*} q_f$, which proves that t is accepted by $\mathcal{A}$.

Assume that $t \xrightarrow[g(l_1,\ldots,l_n)\to f(r_1,\ldots,r_m)]{} u$: $t|_p = g(l_1\sigma, \ldots, l_n\sigma)$ and $u = t[f(r_1\theta, \ldots, r_m\theta)]_p$. Let moreover

$$u \xrightarrow[\mathcal{A}]{*} t[f(q_1^*, \ldots, q_m^*)]_p \xrightarrow[\mathcal{A}]{} t[q^*]_p \xrightarrow[\mathcal{A}]{*} q$$

Then, by construction, there is a rule $g(q''_{l_1}, \ldots, q''_{l_n}) \to q^*$ in $\mathcal{A}$. And

$$t \xrightarrow[\mathcal{A}]{*} t[g(q''_{l_1}, \ldots, q''_{l_n})]_p \xrightarrow[\mathcal{A}]{} t[q^*]_p \xrightarrow[\mathcal{A}]{*} q$$

The case $t \xrightarrow[g(l_1,\ldots,l_n)\to x]{} u$ is similar.

Any term accepted by $\mathcal{A}$ can be reduced to a term in L Let $\mathcal{A}_N$ be the automaton obtained after applying N inference steps. We prove, by induction on N that if $t \xrightarrow[\mathcal{A}_N]{*} q$ and $q \in Q_0$, then there is a term u such that $t \xrightarrow[R_V]{*} u$ and $u \xrightarrow[\mathcal{A}_0]{*} q$

For $N = 0$, there is nothing to prove. Assume now that the rule ρ is added at step $N+1$. We prove the result by induction on the number of times ρ is used in the reduction $t \xrightarrow[\mathcal{A}_{N+1}]{*} q$. If it is used 0 times, then we are back to our first induction hypothesis. Assume now that

$$t \xrightarrow[\mathcal{A}_N]{*} t_1 \xrightarrow[\rho]{} t_2 \xrightarrow[\mathcal{A}_{N+1}]{*} q.$$

Let ρ be the rule $g(q''_{l_1}, \ldots, q''_{l_n}) \to q^*$ and assume for instance that it is obtained from the rule $f(q_1^*, \ldots, q_m^*) \to q^*$ of $\mathcal{A}_N$. There is a position p such that $t_1|_p = g(q''_{l_1}, \ldots, q''_{l_n})$ and $t_2 = t_1[q^*]_p$. By construction, for each i, there is a term u_i, which is an instance of r_i

and which is accepted in state q_i^*. Now, let u be $t[f(u_1, \ldots, u_m)]_p$. $u \xrightarrow[\mathcal{A}_N]{*} t_2$ and hence $u \xrightarrow[R_V]{*} v \xrightarrow[\mathcal{A}_0]{*} q$ by induction hypothesis. Moreover, $t \xrightarrow[g(l_1,\ldots,l_n) \to f(r_1,\ldots,r_m)]{} u$ since the variables of the right and left hand sides are disjoint. Hence $t \xrightarrow[R_V]{*} v$, which is accepted by $\mathcal{A}_0$ in state q.

□

Lemma 21 *For any linear rewrite system R, N_{R_V} is definable in WSkS.*

Proof:

Follows from lemmas 19 and 20. □

Example 22 Let us consider the very simple example:

$$R = \begin{cases} h(h(x)) & \to & f(g(x)) \\ g(x) & \to & h(a) \\ f(a) & \to & f(a) \end{cases}$$

In the system R_V, the variable on the right side of the first rule is replaced with another variable y. In order to compute an automaton which accepts N_{R_V}, we use the construction of lemma 20, starting with the recognizable subset of T: $NF(R_V)$ $(= NF(R))$.

The states of the automaton $\mathcal{A}_{NF(R)}$ are $q_a, q_{h(x)}, q_x, q_r$ (the strict subterms of left hand sides and a failure state). The production rules, besides the rules yielding q_r, consist of:

$$\begin{array}{rclrclrcl} a & \to & q_a & h(q_a) & \to & q_{h(x)} & f(q_{h(x)}) & \to & q_x \\ h(q_x) & \to & q_{h(x)} & f(q_x) & \to & q_x & & & \end{array}$$

Final states will be $q_a, q_{h(x)}$ and q_x.

The automata $\mathcal{A}'_0$ and $\mathcal{A}''_0$ contain the following rules:

$$\begin{array}{rclrclrcl} a & \to & q'_a & \Omega & \to & q'_x & f(q'_a) & \to & q'_{f(a)} \\ f(q'_x) & \to & q'_x & f(q'_{g(x)}) & \to & q'_{f(g(x))} & g(q'_x) & \to & q'_{g(x)} \\ h(q'_a) & \to & q'_{h(a)} & h(q'_x) & \to & q'_{h(x)} & q'_a & \to & q'_x \\ q'_{h(a)} & \to & q'_{h(x)} & q'_{f(a)} & \to & q'_x & q'_{h(x)} & \to & q'_x \\ q'_{g(x)} & \to & q'_x & q'_{f(g(x))} & \to & q'_x & a & \to & q''_a \\ \Omega & \to & q''_x & h(q''_x) & \to & q''_{h(x)} & q''_a & \to & q''_x \\ q''_{h(x)} & \to & q''_x & f(q''_x) & \to & q''_x & g(q''_x) & \to & q''_x \end{array}$$

We arrive at the saturation process which produces the following rules (we exclude the rules yielding q_r, q'_x, q''_x). By superposition with the second rewriting rule, we get

$$\begin{array}{lllllllll} g(q''_x) & \rightarrow & q_{h(x)} & g(q''_x) & \rightarrow & q'_{h(x)} & g(q''_x) & \rightarrow & q'_{h(a)} \\ g(q''_x) & \rightarrow & q''_{h(x)} & & & & & & \end{array}$$

By superposition with the first rule, we get:

$$h(q''_{h(x)}) \rightarrow q'_{f(g(x))} \qquad h(q''_{h(x)}) \rightarrow q_x$$

For instance the last rule is obtained as follows: there is an instance of $g(y)$ which is accepted in state $q_{h(x)}$ (thanks to the rule $g(q''_x) \rightarrow q_{h(x)}$). Hence, from the rules $h(h(x)) \rightarrow f(g(y))$ and $f(q_{h(x)}) \rightarrow q_x$, we deduce $h(q''_{h(x)}) \rightarrow q_x$.

Corollary 23 *NVNF-sequentiality is decidable for left linear (possibly overlapping) rewrite systems.*

Proof: This is a consequence of theorem 10 and lemma 21. □

Similarly, we have the decidability of NV-sequentiality (as defined in [17]): it suffices to start with T instead of starting the construction with $NF(\mathcal{R})$. Note that this gives a much simpler proof than in [17], and in a more general case.

We could also prove this result using ideas similar to [5, 6]:

4.3.1 An indirect (very short) proof of lemma 21:

Using a construction similar to that of lemma 14, the relation $\xrightarrow[\mathcal{R}_V]{}$ is recognized by a Ground Tree Transducer, as defined in [5] and hereafter called GTT. Such a construction is only valid for rewrite systems such that the right and left hand sides do not share variables.

As shown in [5], the class of binary relations which are accepted by a GTT is closed under transitive closure: $\xrightarrow[\mathcal{R}_V]{*}$ is recognized by a GTT, hence recognizable as a set of pairs: $\xrightarrow[\mathcal{R}_V]{*}$ is definable in WSkS (see e.g. [6]). Now, $NF(\mathcal{R})$ is also definable in WSkS (lemma 16), hence N_{R_V} is also definable in WSkS. □

Note the tricks of this proof: there are (at least) three notions of recognizability for sets of pairs of trees (i.e. binary relations on $T \cup \mathcal{T}_\Omega$):

- Rec_1 is the class of cartesian products of recognizable sets.
- Rec_2 is the class of languages accepted by a GTT.
- Rec_3 is the class of trees over the square of the alphabet that are recognizable (each tree over the square alphabet corresponds roughly to a pair of trees according to the well known encoding: see e.g. [6]).

These three classes are distinct and ordered according to the hierarchy (all valid inclusions are displayed):

$$Rec_1 \subset Rec_3 \quad \text{and} \quad Rec_2 \subset Rec_3$$

Rec_3 corresponds to the definability in WSkS. However, if R is in Rec_3, the transitive closure of R might not be in Rec_3. Having no relations between the variables of the left and right hand sides implies immediately that $\xrightarrow[\mathcal{R}_V]{}$ is in Rec_3. But the main trick is to notice that it is actually in Rec_2, which is closed under transitive closure.

4.3.2 Back to strong sequentiality

Strong sequentiality of R can actually be viewed as the sequentiality of a rewrite system R'_V in which all right hand sides have been replaced by a variable not occurring in the corresponding left hand side. Hence, we get, as a corollary of lemma 21:

Lemma 24 *For left linear rewrite systems R, $N?_R$ is recognized by a finite tree automaton.*

This lemma can also be proved, of course, using the Ground Tree Transducers.

4.4 Sequentiality of shallow linear rewrite systems

Lemma 25 *If $\mathcal{R}$ is a shallow linear rewrite system, then $\mathcal{R}$ preserves regularity.*

Proof: As before, we start with an automaton $\mathcal{A}_0$ which accepts a language L and close $\mathcal{A}_0$ backwards w.r.t. $\mathcal{R}$.

Let $\mathcal{A}_1$ be an automaton whose states are the ground subterms of the lefy and right hand sides of $\mathcal{R}$ and an aditional state q_x. The production rules of $\mathcal{A}_1$ are such that all terms are accepted in q_x and, for a ground term t, only t is accepted in q_t.

Now, let us start with $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1$. More precisely, the set of states of $\mathcal{A}$ is the union of the sets of states of $\mathcal{A}_0$ and $\mathcal{A}_1$ respectively. Only the final states of $\mathcal{A}_0$ are final states of $\mathcal{A}$. The set P of production rules is set initially to the union of production rules in $\mathcal{A}_0$ and $\mathcal{A}_1$. Then we saturate P (compute a least fixed point) using the following inference rules:

$$\frac{f(t_1, \ldots, t_n) \rightarrow g(u_1, \ldots, u_m) \in \mathcal{R} \quad g(q_1, \ldots, q_m) \rightarrow q \in P}{f(q'_1, \ldots, q'_n) \rightarrow q \in P}$$

If

- when u_i is a ground term, then u_i is accepted in state q_i (by the current automaton)
- $q'_i = q_j$ whenever $t_i = u_j$
- $q'_i = q_{t_i}$ when $t_i \in T$
- $q'_i = q_x$ when t_i is a variable not occurring in the right side of the rule

and

$$\frac{f(t_1, \ldots, t_n) \rightarrow x \in \mathcal{R} \quad q \in Q}{f(q_1, \ldots, q_n) \rightarrow q \in P}$$

If

- x is a variable
- $q_i = q$ whenever $t_i = x$
- $q_i = q_{t_i}$ whenever t_i is a ground term.
- $q_i = q_x$ when t_i is a variable distinct from x.

This terminates since the set of states being fixed, the number of possible inferred production rules is bounded. It yields a finite bottom-up tree automaton accepting all terms that can be reduced to some term in L. There are two inclusions to prove:

All terms that can be reduced to a term in L are accepted by $\mathcal{A}$.

We prove that $\xrightarrow[\mathcal{R}]{} \circ \xrightarrow[\mathcal{A}]{*} \subseteq \xrightarrow[\mathcal{A}]{*}$. (Then, by a simple induction on the number of reduction steps, we get the desired inclusion).

Let $t \xrightarrow[\mathcal{R}]{} t_1$: $t|_p = f(l_1, \ldots, l_n)\sigma$ and $t_1 = t[g(r_1, \ldots, r_m)\sigma]_p$ for some rewrite rule $f(l_1, \ldots, l_n) \to g(r_1, \ldots, r_m) \in \mathcal{R}$ (the case where the right hand side is a variable is similar). And let

$$t_1 \xrightarrow[\mathcal{A}]{*} t[g(q_1, \ldots, q_m)]_p \xrightarrow[g(q_1,\ldots,q_m)\to q]{} t[q]_p \xrightarrow[\mathcal{A}]{*} q'$$

By construction of $\mathcal{A}$, there is a production rule $r \stackrel{\text{def}}{=} f(q'_1, \ldots, q'_n) \to q$ in P such that r_i is accepted in state q_i when r_i is a ground term, $q'_i = q_j$ whenever $l_i = r_j$, $q'_i = q_{l_i}$ when l_i is ground and $q'_i = q_x$ when l_i is a variable which does not occur in $g(r_1, \ldots, r_m)$. Then there is a reduction of each $t|_{p\cdot i}$ to q'_i:

- if l_i is a variable which does not occur in $g(r_1, \ldots, r_m)$, then $l_i\sigma \xrightarrow[\mathcal{A}]{*} q_x$ by definition of q_x
- if $l_i \in T$, then $l_i \xrightarrow[\mathcal{A}]{*} q_{l_i}$ by definition of q_{l_i}
- if l_i is a variable and $l_i = r_j$ for some j, then $l_i\sigma = r_j\sigma$ and hence $l_i\sigma \xrightarrow[\mathcal{A}]{*} q_j = q'_i$

Now, there is a reduction of $t|_p$ to $f(q'_1, \ldots, q'_n)$, yielding

$$t \xrightarrow[\mathcal{A}]{*} t[f(q'_1, \ldots, q'_n)]_p \xrightarrow[\mathcal{A}]{} t[q]_p \xrightarrow[\mathcal{A}]{*} q'$$

All terms accepted by $\mathcal{A}$ can be reduced to a term in L .

This is proved by induction on the number of times the above inference rules have been applied. After 0 inference step, the automaton only accepts terms that are accepted by $\mathcal{A}_0$. Consider now the automaton $\mathcal{A}^*_N$ computed after N inference steps and assume that $\mathcal{A}^*_N$ only accepts terms that that can be reduced to aterm in L. Let $\mathcal{A}^*_{N+1}$ be the automaton obtained by augmenting the set of production rules of $\mathcal{A}^*_N$ with the rule $r \stackrel{\text{def}}{=} f(q'_1, \ldots, q'_n) \to q$ following the conditions of one of the inference rules. Assume $t \xrightarrow[\mathcal{A}^*_{N+1}]{*} q_f$. We prove by induction on the number of times r is applied in this reduction that there is a term u such that $t \xrightarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{A}_0]{*} q_f$. If r is not applied at all, then $t \xrightarrow[\mathcal{A}_N^*]{*} q_f$ and we apply the first induction hypothesis. Now, if $t \xrightarrow[\mathcal{A}_N^*]{*} t_1 \xrightarrow[r]{} t_2 \xrightarrow[\mathcal{A}_{N+1}^*]{*} q_f$, then there is a position p of t_1 such that $t_1|_p = f(q'_1, \ldots, q'_n)$ and $t_2|_p = q$.

We investigate now the possible constructions of r:

First inference rule We build a term u' as follows: $u' = t[g(u_1, \ldots, u_m)]_p$ where $u_j = t|_{p \cdot i}$ whenever $q'_i = q_j$ and $u_j = v$ whenever $q_j = q_v$ (for v a ground term). Then $t \xrightarrow[\mathcal{R}]{} u'$. On the other hand, $u' \xrightarrow[\mathcal{A}_N^*]{*} t_2$, by construction of r and since for every $i = 1, \ldots, n$, $t|_{p \cdot i} \xrightarrow[\mathcal{A}_N^*]{*} q'_i$. Hence, by induction hypothesis, there is a term u such that $u' \xrightarrow[\mathcal{R}]{*} u' \xrightarrow[\mathcal{A}_0]{*} q_f$ and finally, $t \xrightarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{A}_0]{*} q_f$.

Second inference rule We proceed in a similar way: we let $u' = t[t|_{p \cdot i}]_p$; $t \xrightarrow[\mathcal{R}]{} u'$. $t|_{p \cdot i} \xrightarrow[\mathcal{A}_N^*]{*} q_i$ and $q_i = q$. Hence $u' \xrightarrow[\mathcal{A}_N^*]{*} t_2$ by construction. Now, using the induction hypothesis on u', there is a term u such that $u' \xrightarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{A}_0]{*} q_f$, hence $t \xrightarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{A}_0]{*} q_f$.

□

Lemma 26 *If $\mathcal{R}$ is a shallow linear rewrite system, then $N_{\mathcal{R}}$ is recognizable.*

Proof: This is a consequence of lemmas 19 and 25. □

Let us show an example of the automaton accepting all terms that can be reduced to a normal form.

Example 27 We use the example 15. The construction of lemma 25 is applied to $\mathcal{A}_0 = \mathcal{A}_{NF(\mathcal{R})}$. In what follows, for sake of simplicity, we will not consider the rules which involve a state q_r (they do not play any role). We use here the notation $q_\top$ instead of the state q_x of lemma 25 in order to avoid the confusion with the state q_x of example 15. Also, some of the rules of $\mathcal{A}_1$ are already in $\mathcal{A}_0$. Then we do not need to duplicate them. In the end, the initial automaton $\mathcal{A}$ is the automaton $\mathcal{A}_0$ augmented with the following rules:

$$
\begin{array}{rclrclrcl}
f(q_a, q_a) & \rightarrow & q_{f(a,a)} & g(q_{f(a,a)}) & \rightarrow & q_{g(f(a,a))} & a & \rightarrow & q_\top \\
g(q_\top) & \rightarrow & q_\top & h(q_\top) & \rightarrow & q_\top & f(q_\top, q_\top) & \rightarrow & q_\top \\
\Omega & \rightarrow & q_\top & & & & & &
\end{array}
$$

Now, we start the saturation process, yielding to the new rules (in order of computation, and excluding the rules yielding $q_\top$ which are irrelevant):

$$
\begin{array}{rclrclrcl}
h(q_\top) & \rightarrow & q_{g(f(a,a))} & f(q_\top, q_a) & \rightarrow & q_a & f(q_{g(a)}, q_x) & \rightarrow & q_x \\
f(q_{g(a)}, q_a) & \rightarrow & q_x & h(q_\top) & \rightarrow & q_{g(a)} & & &
\end{array}
$$

This last rule is obtained after noticing that $f(a,a) \xrightarrow{*} q_a$, hence from $h(x) \rightarrow g(f(a,a)) \in R$ and $g(q_a) \rightarrow q_{g(a)}$, we can deduce $h(q_\top) \rightarrow q_{g(a)}$.

Let us consider two examples of computations using the resulting automaton:

$$h(\Omega) \rightarrow h(q_\top) \rightarrow q_{g(a)}$$

hence $h(\Omega)$ is accepted by the automaton.

$$f(g(a), \Omega) \rightarrow f(g(q_a), \Omega) \rightarrow f(g(q_a), q_\top) \rightarrow f(q_{g(a)}, q_\top)$$

the reduction cannot be continued any longer (except by going to $q_\top$). Moreover, there is no other computation sequence: $f(g(a), \Omega)$ is not accepted.

As a consequence of theorem 10 and lemma 26 we have the new decidability result:

Corollary 28 *The sequentiality of shallow linear rewrite systems is decidable.*

For non-shallow systems, lemma 25 does no longer hold. For instance, consider the rewrite system consisting in the single rule $f(g(x), h(y)) \rightarrow f(x, y)$ and $L = \{f(0,0)\}$. Then the set of terms that reduces to some term of L is $\{f(g^n(0), h^n(0)) \mid n \geq 0\}$ which is not recognizable.

4.5 Sorted systems

All above results can be extended to *order-sorted rewrite systems.* In such systems, variables are restricted to range over some regular sets of trees[3]. In particular, we find again some decidability results of [13] as well as their extension to arbitrary left-linear rewrite systems.

[3] Order-sorted signatures, which include subsort declarations and overloading are exactly tree automata, as noticed e.g. in [4]

5 Direct construction of the automaton

For reasons which have already been explained, we construct here directly the automaton accepting all terms that have an index w.r.t $\mathrm{Red}_{\mathcal{R}}$. We only consider left linear rewrite systems.

For the direct construction of automata, it is more convenient to use the formalism of [14]. Of course, we could construct an automaton using the correspondence between automata and logic, but this construction would be too complex.

5.1 Another characterization of indexes

Let $t \in \mathcal{T}(\mathcal{F} \cup \{\Omega\}, \mathcal{X})$ and $u \in T(\mathcal{F}, X)$. We say that t is *compatible* with u if there is some instance v of u such that $t \sqsubseteq v$.[4] Let R be a rewrite system. Then $\xrightarrow[R_\Omega]{}$ is the relation on $\mathcal{T}_\Omega$ defined by $t \xrightarrow[R_\Omega]{} u$ iff there is a $p \in Pos(t)$ such that $t|_p \neq \Omega$ and $t|_p$ is compatible with some left hand side of a rule and $u = t[\Omega]_p$. $\xrightarrow[R_\Omega]{}$ is a convergent reduction relation. The normal form of a term t w.r.t. $\xrightarrow[R_\Omega]{*}$ is written $t\downarrow_{R_\Omega}$.

Theorem 29 ([14]) *Let $x \in \mathcal{X}$. The position p such that $t|_p = \Omega$ is an index of t (w.r.t. $Red_{\mathcal{R}}$) iff $(t[x]_p \downarrow_{R_\Omega})|_p = x$.*

5.2 Construction of the automaton $\mathcal{A}_{noind}$

We construct first an automaton accepting the set of terms which that can be reduced to Ω by R_Ω.

Lemma 30 *For every linear rewrite system, there is an automaton $\mathcal{A}_{R_\Omega}$ with $O(|R|)$ states which recognizes the set of terms $t \in \mathcal{T}_\Omega$ that can be reduced to Ω using R_Ω.*

Actually this lemma can be seen as a particular case of lemma 21. Let us however show a slightly different construction in detail since we will need additional properties.

Proof: With each strict subterm t of a left hand side of a rule (up to literal similarity), we associate a state q_t. In addition, we have the final state q_Ω and, if necessary, the state q_x (x is a variable). Then, for each $f \in \mathcal{F}$ and each $q_{t_1}, \ldots, q_{t_n}$, we add the production rules

[4] Note that this comptatibility relation is not symmetric.

S1: $f(q_{t_1}, \ldots, q_{t_n}) \to q_u$ if $f \neq \Omega$ and $f(t_1, \ldots, t_n)$ is compatible with u and q_u is in the set of states.

S2: $f(q_{t_1}, \ldots, q_{t_n}) \to q_\Omega$ if, $f(t_1, \ldots, t_n)$ is compatible with a left hand side of a rule

S3: $\Omega \to q_\Omega$

The number of states in $\mathcal{A}_{R_\Omega}$ is $O(|R|)$ and the number of rules is $O(|R|^{k+1})$ where k is the maximal arity of a function symbol.

We have now to show two inclusions.

Every term accepted by $\mathcal{A}_{R_\Omega}$ can be reduced to Ω by R_Ω . We prove, by induction on the length of the reduction (i.e. the size of t) that, if $t \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q_u$ then $t \xrightarrow[R_\Omega]{*} v$ for some v which is compatible with u.

When the length is 1, t is a constant and $t = u$ or u is a variable.

Assume now that $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{A}_{R_\Omega}]{*} f(q_{u_1}, \ldots, q_{u_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} q_u$. By induction hypothesis, for every i, $t_i \xrightarrow[R_\Omega]{*} v_i$ for some v_i which is compatible with u_i. Moreover, either $u = \Omega$ and $f(u_1, \ldots, u_n)$ is compatible with a left hand side of a rule, in which case $f(v_1, \ldots, v_n)$ is also compatible with a left hand side of a rule and we have $f(v_1, \ldots, v_n) \xrightarrow[R_\Omega]{} \Omega$. Or else $f(u_1, \ldots, u_n)$ is compatible with u, in which case $f(v_1, \ldots, v_n)$ is also compatible with u. In all situations $f(t_1, \ldots, t_n) \xrightarrow[R_\Omega]{*} v$ for some v which is compatible with u.

Now, applying this to the case $u = \Omega$, we have that $t \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q_\Omega$ implies $t \xrightarrow[R_\Omega]{*} \Omega$ since Ω is the only term compatible with Ω.

Every term in $\mathcal{T}_\Omega$ that can be reduced to Ω by R_Ω is accepted by $\mathcal{A}_{R_\Omega}$. We prove this part by induction on the length of the reduction to Ω. If the term t is Ω itself, then there is nothing to prove. If $t \xrightarrow[R_\Omega]{} u \xrightarrow[R_\Omega]{*} \Omega$, by induction hypothesis $u \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q_\Omega$. Let $u = t[\Omega]_p$ and $t|_p \xrightarrow[R_\Omega]{} \Omega$. By definition of R_Ω, this means that replacing Ω's with terms in $t|_p$ we get an instance $l\sigma$ of a left hand side of a rule l. $l\sigma$ itself is accepted in state q_Ω by construction. Let ρ_1 be a successful run of the automaton on $l\sigma$. Now, we construct a run on $t|_p$ as follows: if p' is a position of both $l\sigma$ and $t|_p$ and $t(p \cdot p') = l\sigma(p')$, then we let $\rho(p') = \rho_1(p')$.

Otherwise, by hypothesis, we have $t(p \cdot p') = \Omega$, in which case, we let $\rho(p') = q_\Omega$. ρ is a run indeed.

□

Note that $\mathcal{A}_{R_\Omega}$ is in general non-deterministic and that determinizing it may require exponentially many states. For example, if $g(f(x,a))$ and $h(f(x,b))$ are two left members of R, from $f(q_x, q_\Omega)$ we can reach the two states $q_{f(x,a)}$ and $q_{f(x,b)}$ and we cannot commit to any of them before knowing what is the symbol above. One way to prevent this situation is to add, for every subterm of a left hand side of a rule a state for each term obtained by replacing some subterms of t with Ω. But this step is exponential. It is not, in principle, better than determinizing $\mathcal{A}_{R_\Omega}$. Following this, it is possible to compute an automaton $\mathcal{A}_{ind}$ which accepts all terms that have an index. The automaton would be non-deterministic and contain exponetially many states. We will show its construction later on. However, for the decision problem, we need to decide wether all irreducible terms in $\mathcal{T}_\Omega$ are accepted by $\mathcal{A}_{ind}$. This question can only be decided in exponential time w.r.t. the number of states of the automaton (automata inclusion is EXPTIME-complete when the right member of the inclusion can be non-deterministic [20]). This would yield a doubly exponential test. It is however possible to reduce the complexity to a single exponential, computing directly an automaton for the complement of $\mathcal{A}_{ind}$ and deciding its inclusion in the set of reducible terms. That is what we are doing now.

Lemma 31 *For every left linear rewrite system R, it is possible to compute in $O(2^{|R|})$ time an automaton A_{noind} which accepts the terms $t \in \mathcal{T}_\Omega$ that do not have an index.*

Proof: Let Q_{R_Ω} be the set of states of $\mathcal{A}_{R_\Omega}$. The states Q of our automaton will consist of pairs of subsets of Q_{R_Ω}. The first components of such pairs will be written as disjunctions of states $q_1 \vee \ldots \vee q_n$, whereas the second components will be written as conjunction of states $q_1 \wedge \ldots \wedge q_n$. The final states will be pairs $[S; \emptyset]$. Intuitively, the second component in the pair correspond to terms that have to be "eliminated" by R_Ω.

The production rules are defined as follows: first the rule for Ω is

$$\Omega \quad \rightarrow \quad [\{q_\Omega\}, \{q_x\}]$$

On the first component, we will find the behaviour of the automaton as in $\mathcal{A}_{R_\Omega}$. The second component correspond to index guess: if Ω has been replaced with x, we enter the state q_x.

The progression rules are defined as follows:

$$f([S_1; S'_1], \ldots, [S_n; S'_n]) \to [S; S']$$

if

- $S = \{q \mid \forall i = 1 \ldots n, \exists t_i \in S_i, f(q_{t_1}, \ldots, q_{t_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} q\}$
- $S' = \{\phi(t', i) \mid (t', i) \in E\}$ where ϕ is a mapping from E to states such that
$$f(q_{t_1}, \ldots q_{t_{i-1}}, q_{t'}, q_{t_{i+1}}, \ldots, q_{t_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} \phi(t', i)$$
for some $q_{t_1}, \ldots, q_{t_n}$ belonging respectively to $S_1, \ldots, S_n$.
- E is the set of pairs (t', i), $i = 1 \ldots n$, $t' \in S'_i$ such that there is no states $q_{t_1}, \ldots, q_{t_n}$ belonging to $S_1, \ldots, S_n$ respectively such that $f(q_{t_1}, \ldots, q_{t_{i-1}}, q_{t'}, q_{t_{i+1}}, \ldots q_{t_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} q_\Omega$.

Intuitively, in the first component we record all possible behaviours of $\mathcal{A}_{R_\Omega}$, whereas in the second component, we superpose all behaviours corresponding to all index guesses.

We claim that a term $t \in \mathcal{T}_\Omega$ is accepted by this automaton iff it has no index, i.e. iff for all positions p of Ω in t, $p \notin Pos((t[x]_p) \downarrow_{R_\Omega})$.

We have to prove two inclusions.

If $t \in \mathcal{T}_\Omega$ is accepted by $\mathcal{A}$, then t has no index

First note that, by construction, $t \xrightarrow[\mathcal{A}]{*} [S; S']$ for some S' iff $t \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q_u$ for all $q_u \in S$.

Assume $t \xrightarrow[\mathcal{A}]{*} [S; \emptyset]$. Let p be a position of Ω in t. We will show that there is prefix p' of p such that $t[x]_p|_{p'} \xrightarrow[R_\Omega]{*} \Omega$. More precisely, let $T_{\Omega,x}$ be the set of terms in $\mathcal{T}(\mathcal{F} \cup \{\Omega, x\})$ that contain at most one occurrence of x and T_x be the subset of T_x of terms that do not contain any Ω. If ρ is a run of $\mathcal{A}$ on $t \in T_{\Omega,x}$, we show that

> $\rho(\Lambda) = [S; S']$ implies that, if there is a $t' \in T_x$ such that $t[x]_p \downarrow_{R_\Omega}$ is incompatible with all u such that $q_u \in S'$, then $t[x]_p \xrightarrow[R_\Omega]{*} v$ where $v \in \mathcal{T}_\Omega$.

As a particular case, when $S' = \emptyset$, we will have the desired result.

We show the property by induction on the size of t. If $t = \Omega$, then $t \xrightarrow[\mathcal{A}]{} [\{q_\Omega\}; \{q_x\}]$ and $t[x]_\Lambda \downarrow_{R_\Omega} = x$ is compatible with x and $q_x \in S'$.

Assume $t \xrightarrow[\mathcal{A}]{*} f([S_1; S_1'], \ldots, [S_n; S_n']) \xrightarrow[\mathcal{A}]{} [S; S']$. Assume moreover that $t[x]_p \downarrow_{R_\Omega}$ is incompatible with any u such that $q_u \in S'$. Finally, assume w.l.o.g that for all indices $i \neq 1$, $t[x]_p|_i \in \mathcal{T}_\Omega$ (in other words, the first symbol of p is assumed to be 1). If $t[x]_p|_1 \downarrow_{R_\Omega} \in \mathcal{T}_\Omega$ or if it is incompatible with any u_1 such that $q_{u_1} \in S_1'$, then by induction hypothesis, $t[x]_p|_1 \xrightarrow[R_\Omega]{*} v_1 \in \mathcal{T}_\Omega$, hence $t[x]_p \xrightarrow[R_\Omega]{*} v \in \mathcal{T}_\Omega$. Otherwise, there is a term $t_1' \in T_x$ s.t. $t[x]_p|_1 \downarrow_{R_\Omega} \sqsubseteq t_1'$ and t_1' is an instance of $q_{u_1} \in S_1'$. By contradiction, suppose that $(u_1, 1) \in E$ and let $q_v = \phi(u_1, 1)$. Then $f(q_{u_1}, q_{t_2}, \ldots, q_{t_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} q_v$ for some t_i s.t. $t|_i \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q_{t_i}$. Now, we have, for every $i \geq 2$, $t|_i \downarrow_{R_\Omega} \sqsubseteq t|_i$, if $t_i \neq \Omega$ then $t|_i$ is compatible with t_i and t_i is compatible with $v|_i$ (actually t_i is either Ω or an instance of $v|_i$). Hence, for every $i \geq 2$, $t|_i \downarrow_{R_\Omega}$ is compatible with $v|_i$. On the other hand, $t[x]_p|_1 \downarrow_{R_\Omega}$ is compatible with u_1, hence with $v|_1$ (u_1 is an instance of $v|_1$). It follows that $t[x]_p \downarrow_{R_\Omega}$ is either Ω or

$$f(t[x]_p|_1 \downarrow_{R_\Omega}, t|_2 \downarrow_{R_\Omega}, \ldots, t_n \downarrow_{R_\Omega})$$

and is both cases it is compatible with v. Hence a contradiction.

If $t \in \mathcal{T}_\Omega$ has no index, then it is accepted by $\mathcal{A}$. We prove, by induction on the size of t that there is a pair $[S; S']$ such that $t \xrightarrow[\mathcal{A}]{*} [S; S']$ and $q \in S$ iff $t \xrightarrow[\mathcal{A}_{R_\Omega}]{*} q$ and $q_u \in S'$ iff u is a maximal term (w.r.t. $\sqsubseteq$) s.t. there is a position p of Ω in t s.t. $t[x]_p \downarrow_{R_\Omega} \notin \mathcal{T}_\Omega$ and $t[x]_p \downarrow_{R_\Omega}$ is compatible with u. This will of course imply the desired property.

If $t = \Omega$, then $t \xrightarrow[\mathcal{A}]{} [\{q_\Omega\}; \{q_x\}]$ and $t[x]_\Lambda \downarrow_{R_\Omega} \notin \mathcal{T}_\Omega$ and $t[x]_\Lambda \downarrow_{R_\Omega} = x$ is compatible with x.

Now let $t = f(t_1, \ldots, t_n)$ and $t_i \xrightarrow[\mathcal{A}]{*} [S_i, S_i']$ following the induction hypothesis. Let S' be the set of q_u s.t u is a maximal term (w.r.t. $\sqsubseteq$) s.t. there is a position p of Ω in t s.t. $t[x]_p \downarrow_{R_\Omega} \notin \mathcal{T}_\Omega$ and $t[x]_p \downarrow_{R_\Omega}$ is compatible with u. Similarly, let S be as in the induction conclusion. We only have to show that $f([S_1; S_1'], \ldots, [S_n; S_n']) \xrightarrow[\mathcal{A}]{} [S; S']$. Let p

be a position of Ω in t such that $t[x]_p \downarrow_{R_\Omega} \notin \mathcal{T}_\Omega$. (If there is no such position, then $S' = \emptyset$ and the property holds true). Assume w.l.o.g that $p = 1 \cdot p'$. Then $t[x]_p|_1 \downarrow_{R_\Omega} \notin \mathcal{T}_\Omega$ and $t[x]_p \downarrow_{R_\Omega} = f(t[x]_p|_1 \downarrow_{R_\Omega}, t|_2 \downarrow_{R_\Omega}, \ldots t|_n \downarrow_{R_\Omega})$ is compatible with u. By maximality of u_1, it must be an instance of $u|_1$. For $i \geq 2$, let $t|_i \downarrow_{R_\Omega}$ be accepted in state q_{t_i}. Then letting $\phi(u_1, 1)$ be q_u, we have indeed $f(q_{u_1}, q_{t_2}, \ldots, q_{t_n}) \xrightarrow[\mathcal{A}_{R_\Omega}]{} q_u$

□

Example 32 Assume that the left hand sides of R are

$$\{f(x, g(a)), f(a, a), h(a, x), h(f(b, y), a)\}$$

and let us show a run of the automaton $\mathcal{A}_{noind}$ on $h(f(\Omega, g(\Omega)), \Omega)$:

$$\begin{array}{rcl}
\Omega & \rightarrow & [q_\Omega; q_x] \\
g([q_\Omega; q_x]) & \rightarrow & [q_{g(a)}; q_x] \\
f([q_\Omega; q_x], [q_{g(a)}; q_x]) & \rightarrow & [q_\Omega; q_{f(b,y)}] \\
h([q_\Omega; q_{f(b,y)}], [q_\Omega, q_x]) & \rightarrow & [q_\Omega; \emptyset]
\end{array}$$

Example 33 If the set of left hand sides of R is $\{h(f(x, a), a), h(f(a, x), a)\}$, a run of $\mathcal{A}_{noind}$ on $h(f(\Omega, \Omega), \Omega)$ will be given by:

$$\begin{array}{rcl}
\Omega & \rightarrow & [q_\Omega; q_x] \\
f([q_\Omega; q_x], [q_\Omega; q_x]) & \rightarrow & [q_{f(a,x)} \vee q_{f(x,a)}; q_{f(a,x)} \wedge q_{f(x,a)}] \\
h([q_{f(a,x)} \vee q_{f(x,a)}; q_{f(a,x)} \wedge q_{f(x,a)}], [q_\Omega; q_x]) & \rightarrow & [q_\Omega; q_x]
\end{array}$$

5.3 Complexity issues

As a consequence of lemma 31, we get a complexity result:

Theorem 34 *Strong sequentiality is in EXPTIME when R is left linear (possibly overlapping).*

Proof: R is strongly sequential iff the set of terms that do not have an index is contained in the set of reducible terms. Or, equivalently, R is strongly sequential if there is no term in $\mathcal{T}_\Omega$ which is accepted by both $\mathcal{A}_{noind}$ and $\mathcal{A}_{NF(R)}$. Both automata can be computed in exponential time thanks to lemmas 31 and 16. Intersection can be done in quadratic time and the emptiness decision is again polynomial. □

The complex construction of lemma 31 can only be avoided when any two strict subterm of left hand sides are comparable w.r.t. $\sqsubseteq$ whenever they are headed with the same function symbol. In such a situation, $\mathcal{A}_{R_\Omega}$ can be made deterministic without adding any state (this is quite straightforward) and $\mathcal{A}_{noind}$ can be computed in polynomial time. We have also seen that, in such a case, the automaton $\mathcal{A}_{NF(R)}$ can be computed in polynomial time, hence:

Corollary 35 *For rewrite systems R such that any two strict subterms of a left hand side of a rule which have the same top symbol are comparable w.r.t. $\sqsubseteq$, strong sequentiality is decidable in polynomial time.*

Note that R can be overlapping here.

Example 36

$$R = \left\{ \begin{array}{rcl} f(f(x,y),z) & \rightarrow & f(x,f(y,z)) \\ f(0,x) & \rightarrow & 0 \\ f(s(x),y) & \rightarrow & s(f(x,y)) \\ f(x,y) & \rightarrow & f(y,x) \end{array} \right.$$

R satisfies the condition of corollary 35: any two strict subterms of the left hand sides which are headed with f (actually there is only one such term here) are comparable w.r.t. $\sqsubseteq$.

On the other hand, there are orthogonal rewrite systems that do not satisfy the conditions of corollary 35.

Example 37

$$R = \left\{ \begin{array}{rcl} g(f(a,x)) & \rightarrow & g(a) \\ h(f(x,a)) & \rightarrow & h(a) \end{array} \right.$$

R is non-overlapping and linear. However, the two strict subterms of left hand sides: $f(a,x)$ and $f(x,a)$ are not comparable w.r.t. $\sqsubseteq$.

In the case of non-left linear rewrite systems, the construction does not work, even if we use automata with constraints [2, 3] instead of bottom-up tree automata.

Example 38 Consider the rewrite system with only one rule whose left side is $f(x, x)$. Then

$$f(f(g^k(f(a,a)),b), f(g^k(b),b)) \xrightarrow[R_\Omega]{} f(f(g^k(\Omega),b), f(g^k(b),b)) \xrightarrow[R_\Omega]{} \Omega$$

However, the replacement for Ω in $f(g^k(\Omega), b)$ is known only when we reach the root of the term, i.e. arbitrary "far" from the first reduction. It is not possible to keep such an information in the finite memory of an automaton with constraints.

5.4 Construction of $\mathcal{A}_{ind}$

Now, instead of constructing $\mathcal{A}_{noind}$, let us construct directly $\mathcal{A}_{ind}$. This will show how to find indexes in a term. We start from a deterministic (completely specified) version of $\mathcal{A}_{R_\Omega}$ and complete it as follows. Each state q of $\mathcal{A}_{R_\Omega}$ is duplicated: we add the state $q^\bullet$ which will intuitively mean that we found an index below. Then we add the following rules:

- $\Omega \to q_x^\bullet$ (this is a guess of an index position; we will express that it has to be applied once in each successful run)
- For each rule $f(q_1, \ldots q_i, \ldots, q_n) \to q$ where q is not a final state, we add the rules $f(q_1, \ldots, q_i^\bullet, \ldots, q_n) \to q^\bullet$

Final states are those which are marked with a $\bullet$.

Example 39 Consider a rewrite system whose left hand sides are $f(a, x), f(f(x, y), a)$ (with 3 function symbols, f, a, b).

The states of $\mathcal{A}_{ind}$ are $\{q_a, q_{f(x,y)}, q_\Omega, q_x, q^\bullet_{f(x,y)}, q_x^\bullet\}$ (states $q_a^\bullet$ and $q_b^\bullet$ have been removed since they are useless). The rules are:

$$\begin{array}{rcl rcl rcl}
\Omega & \to & q_x^\bullet & f(q_x, q_x) & \to & q_{f(x,y)} & f(q_1, q_x^\bullet) & \to & q^\bullet_{f(x,y)} \\
\Omega & \to & q_\Omega & f(q_{f(x,y)}, q_{f(x,y)}) & \to & q_{f(x,y)} & f(q^\bullet_{f(x,y)}, q_1) & \to & q^\bullet_{f(x,y)} \\
a & \to & q_a & f(q_x, q_{f(x,y)}) & \to & q_{f(x,y)} & f(q_1, q^\bullet_{f(x,y)}) & \to & q^\bullet_{f(x,y)} \\
b & \to & q_x & f(q_x, q_a) & \to & q_{f(x,y)} & f(q_x^\bullet, q_2) & \to & q^\bullet_{f(x,y)} \\
f(q_{f(x,y)}, q_a) & \to & q_\Omega & f(q_x, q_\Omega) & \to & q_{f(x,y)} & f(q_a, q_3) & \to & q_\Omega \\
f(q^\bullet_{f(x,y)}, q_a) & \to & q_\Omega & f(q_{f(x,y)}, q_x) & \to & q_{f(x,y)} & f(q_\Omega, q_3) & \to & q_\Omega
\end{array}$$

where q_1 is any state in $\{q_x, q_{f(x,y)}\}$, q_2 is any state in $\{q_x, q_{f(x,y)}, q_a, q_\Omega\}$ and q_3 is any state in $\{q_a, q_\Omega, q_x, q_{f(x,y)}, q_x^\bullet, q^\bullet_{f(x,y)}\}$. The final states are $q_x^\bullet$ and $q^\bullet_{f(x,y)}$.

For example, $f(\Omega, \Omega)$ is accepted since $f(q_x^\bullet, q_\Omega) \to q_{f(x,y)}^\bullet$. But $f(f(\Omega,\Omega),\Omega)$ is not accepted. Moreover, this term being irreducible, the system is not strongly sequential.

Lemma 40 *The automaton $\mathcal{A}_{ind}$ accepts all terms in $\mathcal{T}_\Omega$ that have an index.*

<u>**Proof:**</u>
For convenience, we confuse here $\mathcal{A}_{R_\Omega}$ and its deterministic version. We have to prove two inclusions:

Every term accepted by $\mathcal{A}_{ind}$ has an index Let $t \xrightarrow[\mathcal{A}_{ind}]{n} q^\bullet$. We show that t has an index by induction on n: if $n = 0$ then $t = \Omega$ and Λ is an index. If $n > 0$, by definition of the rules, $t \xrightarrow[\mathcal{A}_{ind}]{n-1} f(q_1, \ldots, q_i^\bullet, \ldots, q_n) \to q^\bullet$. By induction hypothesis, $t|_i$ has an index p (since it is accepted). Now, by determinacy of $\mathcal{A}_{R_\Omega}$ and by lemma 30, t can not be reduced to Ω by R_Ω. Hence, there is no redex w.r.t R_Ω in t, along the path $i \cdot p$. Which means that $i \cdot p$ is an index in t by theorem 29.

Every term in $\mathcal{T}_\Omega$ that has an index is accepted by $\mathcal{A}_{ind}$ Let p be an index in $t \in \mathcal{T}_\Omega$. We show by induction on the size of p that t is accepted by $\mathcal{A}_{ind}$. If $p = \Lambda$, then $t = \Omega$ is indeed accepted. Assume now that $p = 1 \cdot q$ and $t = f(t_1, \ldots, t_n)$. Then q is an index in $t|_1$ and hence, by induction hypothesis, there is a state $q^\bullet$ such that $t|_1 \xrightarrow[\mathcal{A}_{ind}]{*} q^\bullet$. Let, for $i > 1$ q_i be the state in which $t|_i$ is accepted by $\mathcal{A}_{R_\Omega}$. p being an index and by lemma 30, $f(q, q_2, \ldots, q_n)$ cannot be rewritten into a final state of $\mathcal{A}_{R_\Omega}$. Hence there is a rule $f(q^\bullet, q_2, \ldots, q_n) \to q'^\bullet$ in $\mathcal{A}_{ind}$, which shows that t can be reduced to a final state $q'^\bullet$.

□

Now, if we come back to the problem of finding an index in a term, we can use $\mathcal{A}_{ind}$: all successful runs on t contain a $\bullet$-marked path from the root to an index. Consider for example the term $f(f(\Omega,\Omega), f(\Omega,\Omega))$, the only successful run is $q_{f(x,y)}^\bullet(q_{f(x,y)}^\bullet(q_x^\bullet, q_\Omega), q_{f(x,y)}(q_\Omega, q_\Omega))$, which shows the path 11, which is the only index.

There is still some work to do w.r.t. index search. Indeed, so far, we have to apply the automaton on the whole term after each reduction step. Huet and Lévy, on the other hand, give a deterministic algorithm which never visits twice a node in the input term. To do something similar, we would have first to consider our automaton top-down (instead of bottom-up as we did through all the paper). Such an automaton is in general non-deterministic (and cannot be determinized). In order to avoid backtracking on the input term we would have to keep a stack of choice points and derive simultaneously the possible runs on the branch which is explored. This requires some additional implementation machinery, which is out of the scope of this paper.

Concerning reduction strategies, we should also note that index reduction is a normalizing strategy *for sequential orthogonal term rewriting systems*, as shown by Huet and Lévy. For overlapping systems, or for non-left linear systems this is no longer true. Hence the extension of decidability results to these cases are only interesting for subclasses of rewrite systems (as e.g. described in [24]).

6 Further applications

We believe that the tree automata approach can be used successfully to other works on reduction strategies. For example the *strong root stability* of [12] is also expressible in WSkS for left linear rewrite systems. The use of automata should also be investigated in parallel reduction strategies, such as in [21]: a run of the automaton not only gives an index position, but all index positions. Our approach could also be used for related notions of rewriting, provided the data structure (terms in our case) can be expressed in SkS.

Acknowledgments

I thank Aart Middeldorp, Takahashi Nagoya, Masahito Sakai, Yoshihito Toyama and Ralf Treinen for their careful reading of a former version of this paper: they found several mistakes. This helped me to improve the paper.

References

[1] Sergio Antoy and Aart Middeldorp, *A sequential reduction strategy*, 4th International Conference on Algebraic and Logic Programming (Madrid, Spain) (Giorgio Levi and Mario Rodríguez-Artalejo, eds.), Lecture Notes in Computer Science, vol. 850, Springer-Verlag, September 1994, pp. 168–185.

[2] B. Bogaert and Sophie Tison, *Equality and disequality constraints on brother terms in tree automata*, Proc. 9th Symp. on Theoretical Aspects of Computer Science (Paris) (A. Finkel, ed.), Springer-Verlag, 1992.

[3] A-C. Caron, J.-L. Coquidé, and M. Dauchet, *Encompassment properties and automata with constraints*, 5th International Conference on Rewriting Techniques and Applications (Montreal, Canada) (Claude Kirchner, ed.), Lecture Notes in Computer Science, vol. 690, Springer-Verlag, June 1993.

[4] Hubert Comon and Catherine Delor, *Equational formulae with membership constraints*, Information and Computation **112** (1994), no. 2, 167–216.

[5] Max Dauchet, Thierry Heuillard, Pierre Lescanne, and Sophie Tison, *The confluence of ground term rewriting systems is decidable*, Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh, 1988.

[6] Max Dauchet and Sophie Tison, *The theory of ground rewrite systems is decidable*, Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, 1990.

[7] Nachum Dershowitz and Jean-Pierre Jouannaud, *Rewrite systems*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), vol. B, North-Holland, 1990, pp. 243–309.

[8] M. Gécseg and M. Steinby, *Tree automata*, Akademia Kiadó, Budapest, 1984.

[9] Gérard Huet and Jean-Jacques Lévy, *Computations in orthogonal rewriting systems II*, Computational Logic: Essays in Honor of Alan Robinson (J.-L. Lassez and G. Plotkin, eds.), MIT Press, 1991, This paper was written in 1979, pp. 415–443.

[10] Jean-Pierre Jouannaud and Walid Sadfi, *Strong sequentiality of left-linear overlapping rewrite systems*, Workshop on Conditional Term Rewriting systems (Jerusalem) (N. Dershowitz, ed.), Lecture Notes in Computer Science, vol. 968, Springer-Verlag, July 1994, pp. 235–246.

[11] J.R. Kennaway, *Sequential evaluation strategies for parallel-or and related reduction systems*, Annals of Pure and Applied Logic **43** (1989), 31–56.

[12] Richard Kennaway, *A conflict between call-by-need computation and parallelism*, Workshop on Conditional Term Rewriting Systems (Jerusalem) (N. Dershowitz, ed.), 1994.

[13] Delia Kesner, *La définition de fonctions par cas à l'aide de motifs dans des langages applicatifs*, Thèse de doctorat, Université Paris-Sud, Orsay, France, December 1993.

[14] Jan Willem Klop and Aart Middeldorp, *Sequentiality in orthogonal term rewriting systems*, Journal of Symbolic Computation **12** (1991), 161–195.

[15] Takashi Nagaya, Masahito Sakai, and Yoshihito Toyama, *NVNF-sequentiality of left-linear term rewriting systems*, Proc. Japanese Workhop on Term Rewriting (Kyoto), July 1995.

[16] Michael J. O'Donnell, *Computing in systems described by equations*, Lecture Notes in Computer Science, vol. 58, Springer, Berlin, West Germany, 1977.

[17] M. Oyamaguchi, *NV-sequentiality: a decidable condition for call-by-need computations in term rewriting systems*, SIAM Journal on Computing **22** (1993), no. 1, 114–135.

[18] M. Rabin, *Decidable theories*, Handbook of Mathematical Logic (J. Barwise, ed.), North-Holland, 1977, pp. 595–629.

[19] M.O. Rabin, *Decidability of second-order theories and automata on infinite trees*, Trans. Amer. Math. Soc. **141** (1969), 1–35.

[20] Helmut Seidl, *Deciding equivalence of finite tree automata*, Siam Journal of Computing **19** (1990), no. 3, 424–437.

[21] R.C. Sekar and I.V. Ramakrishnan, *Programming in equational logic : beyond strong sequentiality*, Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, 1990.

[22] J.W. Thatcher and J.B. Wright, *Generalized finite automata with an application to a decision problem of second-order logic*, Math. Systems Theory **2** (1968), 57–82.

[23] W. Thomas, *Automata on infinite objects*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), Elsevier, 1990, pp. 134–191.

[24] Yoshihito Toyama, *Strong sequentiality of left linear overlapping term rewriting systems*, Proc. 7th IEEE Symp. on Logic in Computer Science (Santa Cruz, CA), 1992.