Free Sequentiality in Orthogonal Order-Sorted Rewriting Systems with Constructors

Delia Kesner
INRIA Rocquencourt
Domaine de Voluceau, BP 105
78153 Le Chesnay Cedex, France
kesner@inria.inria.fr

Abstract

We introduce the notions of sequentiality and strong sequentiality in order-sorted rewriting systems, both closely related to the subsort order and to the form of declarations of the signature. We define *free sequentiality* for the class of orthogonal systems with constructors, a notion which does not impose conditions over the signature. We provide an effective decision procedure for free sequentiality that gives at the same time a simple construction of a non-deterministic pattern matching tree. These trees describe how the refinement of sorts and structures has to be done along the reduction sequence in such a way that wasteful computations are avoided.

1 Introduction

Term rewriting systems provide an elegant framework for programming in equational logic. *Sequentiality* is a property related to the possibility of *systematically* expanding any term step-by-step in order to get its normal form. This means that for every term not in normal form, there exists a redex occurrence which we necessarily need to reduce in every sequence that normalizes the term and this occurrence can be found without looking ahead. Unfortunately, there is no algorithm to compute these *needed redexes* , and there is also no algorithm to decide if a given system is sequential. These results of undecidability motivate the search for more realistic and effective approaches. Our purpose is to present a notion of sequentiality in orthogonal order-sorted rewriting systems with constructors. This notion is related to the possibility of expanding a term by refining either its sort or its structure. We obtain a characterization of a class of order-sorted systems, for which there is an efficient implementation that avoids useless reductions. We discuss some related work in order to place our results in context.

Orthogonal term rewriting systems are defined to have *linear* and *non-ambiguous* left hand sides of rules (LHS). In this class of systems, the parallel-outermost strategy, a normalization strategy which reduces all outermost redexes in each step, has been shown to be complete. Huet and Levy [HL79] showed that a restricted notion of sequentiality, called *strong sequentiality*, that only considers the LHS's, seems to be much more useful in order to perform efficient implementations of orthogonal rewriting systems. They developed a normalization algorithm which eliminates useless computations by only reducing needed redexes.

In term rewriting systems with *constructors*, the outermost symbols of LHS's are only allowed to appear at a root position. All other symbols are called *constructors* and LHS's are called *patterns*. Thus, subterms of patterns do not have reducible instances, and so reducing a term requires essentially the head normal form of some of its subterms in order to match the reduced term with some LHS rule. A *compilation* scheme will transform a set of LHS rules into a case-expression presented as a *discrimination tree*. However, the resulting tree can fail to terminate for some terms that can be matched by a pattern. This is due to wasteful reductions that try to normalize terms which do not have normal forms. A refined notion of strong sequentiality for term rewriting systems with constructors can be found in [Tha87].

Order-sorted algebras provide a substantially improved expressiveness over many sorted algebras. Sorts are partially ordered and terms are constrained with subsorts. In order-sorted rewriting systems sort information is not only useful for runtime type-checking, but also for pattern matching verifications. *Sort decreasing* rewriting systems refine the sort of the terms during the reduction process. As a consequence, a non-terminating reduction always has a minimal sort, and even if this sort is not always known, we can use some *partial information* in order to perform pattern matching. Thus, terms will need to be evaluated only as far as necessary in order to make either a structure or a subsort verification possible. In general, a few computation steps could be sufficient without the need to reduce terms to full normal forms. For example, suppose σ is a sort greater than η, the variable x is constrained by η, denoted as x^η, f is a constructor and t is a term of sort σ. Suppose that t reduces in one step to a term not in normal form whose sort is a subsort of η. If t has to be matched with the term $f(y)$, its head-normal form will be required, whereas if it has to be matched with the term x^η, only one step of reduction will be sufficient. We really take advantage in this way of a *lazy evaluation* mechanism.

Puel and Suárez [PS90] devised a compilation scheme to statically generate pattern matching algorithms in languages with lazy evaluation. They define a notion of strong sequentiality for sets of terms restricted by *structure constraints*[1]. This sequentiality notion has been generalized in [Kes91] to sets of terms restricted not only by structure, but also by *subsort constraints*. However, only some particular order-sorted signatures, called *unitary signatures*, were considered. Unitary signatures are defined to be regular and to verify some properties as for example minimality of the sort that appears in the codomain of every declaration of a function symbol. However, since Comon [CD91] has noticed that order-sorted signatures can also be seen as *finite bottom-up tree automata*, order-sorted rewriting systems can be studied from a more effective point of view. The main advantage consists of the elimination of restrictions to the function declarations and relation subsort orders of the signature. As an example, unification of order-sorted terms can be expressed in this framework with a finite set of constraints, even if there is an infinite set of most general unifiers. In this paper we will use Comon's formalism in order to study sequentiality in a general case, with no restrictions to the signatures.

[1]For example, a term $f(x)$ restricted by "x different from a" is written as $f(x)|x\Diamond a$.

Section 2 presents order sorted signatures and orthogonal order-sorted rewriting systems with constructors. In section 3 we show how to encode order-sorted signatures with bottom-up tree automata and we give the semantics of terms in the tree language recognized by its automaton. Section 4 introduces sequentiality of monotonic predicates over order-sorted partial terms. Our notion of sequentiality not only takes into account the structure of terms, but also their sorts. The idea is that the expansion of a term consists now of the refinement of either its sort or its structure. In section 5 we define *strong sequentiality*, a notion which is based only on left-hand sides of rules. It turns out that terms are not always forced to be reduced to head normal forms because they are evaluated only as far as required to make either a structure or a subsort verification possible. We provide in section 6 a construction method of non-deterministic *pattern matching trees* (PMT's), also called *pattern matching automata* or *pattern matching algorithms*, having transitions labeled with either constructors or subsort symbols. In section 7 we point out the main problems that arise when using such algorithms with order-sorted systems. Finally, we define *free sequential orthogonal constructor* systems, which allow to construct *optimal* PMT's for the evaluation of *well-sorted* terms, even if they are build with "strange" signatures. Optimal PMT's evaluates only the part of the term which is strictly necessary in order to satisfy the matching predicate, taking advantage in this way of the lazy reduction mechanism. The reduction of a well-sorted term t using an optimal PMT causes either a subsort or a structure verification and avoids wasteful computations. Our restrictions becomes in this way a sufficient condition in order to construct optimal order-sorted pattern matching trees. In reading the paper, familiarity with order-sorted algebras and sequentiality would be helpful, but the treatment is self-contained enough to be meaningful on its own.

2 Order-Sorted Rewriting Systems

An *order-sorted signature* $\Sigma = \langle \mathcal{S}, \leq, \mathcal{F}, \mathcal{V}, \mathcal{D} \rangle$ consists of a finite set of sort symbols $\mathcal{S} = \{\top, \bot, \sigma, \eta, \delta, \ldots\}$, a partial order $\leq$ on $\mathcal{S}$ such that $\forall \sigma \in \mathcal{S}, \bot \leq \sigma, \sigma \leq \top$, a set of function symbols $\mathcal{F} = \{f, g, h, \ldots\}$, a set of $\mathcal{S}$-indexed variables $\mathcal{V}^\sigma = \{x^\sigma, y^\sigma, z^\sigma, \ldots\}$ with a countably infinite number of variables for each sort symbol σ, and a set of declarations $\mathcal{D}$ of the form $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma$ where $f \in \mathcal{F}$, $\sigma_1 \ldots \sigma_n$ is the *domain* of f, σ its *codomain* and n its *arity*. The sets $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{V}$ are mutually disjoint. For brevity, we will write $s \in \Sigma$ for any symbol s in $\mathcal{S}$, $\mathcal{F}$, $\mathcal{V}$ or $\mathcal{D}$. From now on, we will use $\overline{\sigma_n}$ to denote a string $\sigma_1 \ldots \sigma_n$ and $f : \overline{\sigma_n} \rightarrow \sigma$ to denote a declaration $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma$.

Terms are constructed as usual: a variable $x^\sigma \in \Sigma$ is a term and $f(t_1 \ldots t_n)$ is a term if and only if n is an arity of f and for all $1 \leq i \leq n$, t_i is a term. Σ-terms are *well-sorted* terms constructed in the following way: a variable $x^\sigma \in \Sigma$ is a Σ-term of sort η if $\sigma \leq \eta$, and $f(t_1 \ldots t_n)$ is a Σ-term of sort η if and only if there is $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma \in \Sigma$ such that $\sigma \leq \eta$ and for all $1 \leq i \leq n$, t_i is a Σ-term of sort σ_i. A signature Σ is regular iff every Σ-term has a least sort. In the following, we will drop the prefix Σ when it is clear from the context.

A substitution θ is a function from terms to terms such that $\mathcal{D}\theta = \{x \mid \theta x \neq x\}$ is finite, $\theta f(t_1 \ldots t_n) = f(\theta t_1 \ldots \theta t_n)$ and when s is a term of sort σ, then θs is also of sort σ. A term is called *linear* if its variables occur at most once and *ground* if it contains no variables. We say that t' is an *instance* of t or t' *matches* t if there is a substitution θ such that $\theta t = t'$. Two terms t and t' are said to be *unifiable* or *compatible*, denoted as $t \uparrow t'$ iff there exists a substitution θ such that $\theta t = \theta t'$. Syntactic unification in finite order-sorted signatures is decidable [SNGM89].

A *rewriting rule* is an ordered pair of terms written as $t \rightarrow r$ such that t is not a variable, and every variable occurring in r occurs in t. A rewriting rule $t \rightarrow r$ is *sort decreasing* if, for every substitution θ and every sort symbol σ, θr has sort σ if θt has sort σ. It is decidable whether a finite rewriting system is sort decreasing [SNGM89]. The set of *positions* or *occurrences* of a term t, denoted $\mathcal{O}(t)$, is defined as usual as finite sequences of positive integers such that $\epsilon \in \mathcal{O}(t)$ (the root of t) and $k.u \in \mathcal{O}(f(t_1 \ldots t_n))$ if $u \in t_k$. The subterm of t at position u, denoted t/u, is defined as $t/\epsilon = t$ and $f(t_1 \ldots t_n)/k.u = t_k/u$. The replacement of the subterm of t at position u by a term p is denoted as $t[u \leftarrow p]$ or $t[s]_u$. When we don't care about the term where the replacement was done we will also write $C[s]_u$. The function symbol f appearing in t at position u is denoted $t|_u$.

A *base* $\mathcal{L}$ for a rewriting system is a finite set of linear terms such that when l_1 and l_2 are two distinct terms in $\mathcal{L}$, $u \in \mathcal{O}(l_1)$ and l_1/u is not a variable, then l_1 and l_2 are not unifiable. A *rewriting system* $\mathcal{R}$ is a finite set $\{l_i \rightarrow r_i\}_{i=1}^n$ of rewriting rules, and we say that it is *orthogonal* when $\{l_i\}_{i=1}^n$ is a base.

A term is a *redex* if it is an instance of some $l \in \mathcal{R}$. If t/u is a redex, u is said to be a *redex occurrence* in t. We say that t *reduces to* h, denoted $t \rightarrow_{\mathcal{R}} h$, iff there is a substitution θ and $u \in \mathcal{O}(t)$ such that $t/u = \theta l$ for some $l \rightarrow r \in \mathcal{R}$, where $h = t[u \leftarrow \theta r]$. Note that when $\mathcal{R}$ is a sort decreasing rewriting system the replacement of a subterm of a term t at position u always yields a well-sorted term. We denoted as $\rightarrow_{\mathcal{R}}^*$ the transitive reflexive closure of $\rightarrow_{\mathcal{R}}$. A term t is in *normal form* iff there is no redex occurrence in t. A term t is in *head normal form* if there is no redex s such that $t \rightarrow_{\mathcal{R}}^* s$.

If every $l_i \in \mathcal{L}$ has the form $f_i(t_1^i, \ldots, t_{m_i}^i)$, $\mathcal{F} = \{f_1 \ldots f_n\}$ is called the set of *head* symbols of $\mathcal{L}$ and $\mathcal{C} = \mathcal{F} - \{f_1 \ldots f_n\}$ the set of *constructors* symbols of $\mathcal{L}$. A term $f(t_1 \ldots, t_m)$ is said to be a *pattern* iff f is a head symbol and each t_i is a term without head symbols. A base $\mathcal{L}$ is a *constructor base* iff every $l_i \in \mathcal{L}$ is a pattern and a system based on such $\mathcal{L}$ is called a *constructor system.* For the rest of the paper we will only deal with sort decreasing systems that follow this constructor discipline. Sort decreasing systems guarantee that the sort of a term may only become smaller, and this is precisely what allows an efficient reduction strategy.

3 Signatures as Bottom-Up Tree Automata

Comon [CD91] has noticed that an order-sorted signature Σ is a finite bottom-up tree automaton (Q, A, F, T), where the set of states Q is the set of sorts $\mathcal{S}$, the set of terminal symbols A is $\mathcal{F}$, the set of final states F is also $\mathcal{S}$ and the set of transitions T is defined by the following rules:

- If $\sigma \leq \eta \in \Sigma$, then there is a ϵ-transition rule $\sigma \Rightarrow \eta$ in T
- If $f : \overline{\sigma_n} \rightarrow \sigma \in \Sigma$, then there is an f-transition rule $f(\overline{\sigma_n}) \Rightarrow \sigma$ in T

We will denote by $T(\Sigma)$ the set of ground terms constructed with Σ and by $T(\Sigma|_\sigma)$ the regular tree language recognized by Σ when replacing the set of final states F by $\{\sigma\}$. It turns out that the set of Σ-terms is $T(\Sigma|_\top)$. We define the set of *sort expressions*, denoted by $\mathcal{E_S}$, in the following way:

- $\sigma \in \mathcal{E_S}$, for any σ in Σ
- If $\sigma, \eta \in \mathcal{E_S}$, then $\sigma \vee \eta, \sigma \wedge \eta \in \mathcal{E_S}$
- If $\sigma_1 \dots \sigma_n \in \mathcal{E_S}$ and $f \in \mathcal{F}$, $f(\sigma_1 \dots \sigma_n) \in \mathcal{E_S}$

Given an automaton Σ, the semantics of a given sort expression σ in $T(\Sigma)$, denoted by $[\![\sigma]\!]^\Sigma$ or simply by $[\![\sigma]\!]$, is a set of ground terms defined in the following way:

- $[\![\bot]\!]^\Sigma = \emptyset$
- $[\![\sigma]\!]^\Sigma = T(\Sigma|_\sigma)$
- $[\![\sigma_1 \vee \sigma_2]\!]^\Sigma = [\![\sigma_1]\!]^\Sigma \cup [\![\sigma_2]\!]^\Sigma$
- $[\![\sigma_1 \wedge \sigma_2]\!]^\Sigma = [\![\sigma_1]\!]^\Sigma \cap [\![\sigma_2]\!]^\Sigma$
- $[\![f(\sigma_1 \dots \sigma_n)]\!]^\Sigma = \{f(\eta_1 \dots \eta_n) \in T(\Sigma) \mid \eta_i \in [\![\sigma_i]\!]^\Sigma, \text{for } i = 1 \dots n\}$

Intuitively, there is no well-sorted term of sort $\bot$, t is of sort expression $\sigma_1 \wedge \sigma_2$ iff it is of σ_1 and σ_2, t is of sort expression $\sigma_1 \vee \sigma_2$ iff it is of σ_1 or σ_2 and, t is of sort expression $f(\sigma_1 \dots \sigma_n)$ iff $t = f(h_1 \dots h_n)$ and each h_i is of sort expression σ_i. The meaning of a sort expression of the form $f(\sigma_1 \dots \sigma_n)$ is to approximate the sort of a given term by describing its skeleton form, even if this skeleton is not well-sorted. We refer to [GS84] in order to show that intersection and union of regular tree languages (languages recognized by bottom-up tree automata) are also regular tree languages.

We extend the order $\leq$ over sort expressions in such a way that $\eta_1 \preceq \eta_2$ iff $[\![\eta_1]\!] \subseteq [\![\eta_2]\!]$. Note that $\eta \leq \sigma$ implies $\eta \preceq \sigma$ but the converse is false. It turns out that the greatest lower bound (resp. least upper bound) of σ_1, σ_2 is $\sigma_1 \wedge \sigma_2$ (resp. $\sigma_1 \vee \sigma_2$) and that $\sigma_1 \preceq \sigma_2$ and $\sigma_2 \preceq \sigma_2$ does not necessary imply that σ_1 and σ_2 are the same sort expression. We write $\sigma_1 \uparrow \sigma_2$ if $\sigma_1 \wedge \sigma_2 \neq \bot$ and $\sigma_1 \sharp \sigma_2$ if $\sigma_1 \wedge \sigma_2 = \bot$. Finally $\sigma \uparrow \Gamma$ if $\exists \delta \in \Gamma$ such that $\sigma \uparrow \delta$.

We are now able to decorate variables with sort expressions as for example in $x^{\sigma \vee \eta}$. Analogously, Σ-terms are constructed replacing $\leq$ by $\preceq$ in the definition of section 2 and all the notions presented there are naturally extended. It turns out that every term has a least regular tree language containing it, even if it has many sorts or it has no least sort. Order-sorted signatures as bottom-up tree automata allow to associate a natural notion of "minimality" to each term, and therefore regularity [SNGM89] becomes a redundant property of signatures.

4 Sequentiality of monotonic predicates

New constant symbols Ω^σ will be added to the signature Σ in order denote terms that contain no information about their structures and sorts. Intuitively, we know nothing about a term $\Omega^\top$ and we only know the sort of a term Ω^σ, with $\sigma \in \mathcal{E_S}$ and $\sigma \neq \top, \bot$. Formally, the set of Ω-terms is defined as follows: Ω^σ is an Ω-term if $\sigma \in \mathcal{E_S}$ and $f(t_1 \dots t_n)$ is an Ω-term iff f has arity n and each t_i is an Ω-term.

Well-sorted Ω-terms are constructed as in section 2: Ω^σ is an Ω-term of sort η if and only if $\sigma, \eta \in \mathcal{E_S}$ and $\sigma \preceq \eta$ and $f(t_1 \ldots t_n)$ is an Ω-term of sort η if and only if $\eta \in \mathcal{E_S}$, there is $f : \sigma_1 \ldots \sigma_n \to \sigma \in \Sigma$ such that $\sigma \preceq \eta$ and for all $1 \leq i \leq n$, t_i is a well-sorted Ω-term of sort σ_i. All the conventional notions regarding substitutions, assignments and unification are naturally extended to Ω-terms. An information ordering on Ω-terms is defined by:

- $\Omega^\sigma \sqsubseteq t$ iff t is of sort η and $\eta \preceq \sigma$.
- $f(t_1 \ldots t_n) \sqsubseteq f(u_1 \ldots u_n)$ iff $\forall i = 1 \ldots n,\ t_i \sqsubseteq u_i$.

From now on, $t \sqcup t'$ denotes the least upper bound of the Ω-terms t and t'. It is important to recall that the formalism of order-sorted signatures as bottom-up tree automata assures that $t \sqcup t'$ is also an Ω-term. For example, the least upper bound of Ω^σ and Ω^η is the Ω-term $\Omega^{\sigma \wedge \eta}$. See [CD91] for details. We will also use $max(S)$ and $min(S)$ to denote the maximal and minimal (w.r.t $\sqsubseteq$) Ω-terms of S respectively.

Let $\mathcal{L}$ be a base and t a term. t_Ω indicates the Ω-term obtaining from t by replacing each variable in t with Ω and $\mathcal{L}_\Omega = \{l_\Omega \mid l \in \mathcal{L}\}$. From now on, we will abuse the notation by writing $\mathcal{L}$ instead of $\mathcal{L}_\Omega$ in order to make the paper more readable. This is possible because $\mathcal{L}$ has linear terms. An Ω-*normal form* is an Ω-term without redexes, containing at least one occurrence of Ω. We reserve the phrase *normal form* for terms containing neither redexes nor Ω's. Every ground term can be translated to an Ω-term by the following *context* Ω-*rule*:

$$C[s] \to_\Omega C[\Omega^\top], \text{ where } s \uparrow \mathcal{L} \text{ and } s \neq \Omega$$

The $\to_\Omega$ reduction is confluent and terminating as it is shown in [KM91], and the generalization of these results for order-sorted systems is straightforward.

Let the truth values be ordered by $false \ll true$. Let P be a monotonic predicate on Ω-terms. Increasing information about a given term (in the sense of $\sqsubseteq$) can only change the value of the predicate to a more favorable one.

In order to define the notion of sequentiality of a monotonic predicate, let's introduce some definitions that we will use for that purpose. The *acceptable set* of Ω-terms associated to a predicate P and a term t, denoted by $Accept(P, t)$ is defined to be the set of Ω-terms h such that $t \sqsubseteq h$ and $P(h) = true$. For each $u \in \mathcal{O}(t)$, we identify two subsets of $Accept(P, t)$ in such a way that the first one has only those terms with constructors at position u and the second one has only those terms with Ω at position u:

$$StrucAccept(P, t, u) = Accept(P, t) \cap \{h[f(t_1 \ldots t_n)]_u\}$$
$$SortAccept(P, t, u) = Accept(P, t) \cap \{h[\Omega^\eta]_u\}$$

We characterized positions of a term t where expansions may be performed, *i.e,* occurrences where we need to expand either the structure or the sort of t in order for the predicate P to become true on the expanded term. Formally, a position $u \in \mathcal{O}(t)$ is said to be a *potential direction* of P in $t[\Omega^\sigma]_u$ if and only if $\forall h \in Accept(P, t)$, $t/u \sqsubset h/u$. A position $u \in \mathcal{O}(t)$ is a *structure direction* of P in $t[\Omega^\sigma]_u$ if and only if $Accept(P, t) = StrucAccept(P, t, u)$ (*i.e*, $SortAccept(P, t, u) = \emptyset$). In the untyped case, a position is a potential direction if and only if it is a structure one and the definition coincides with that of direction in [HL79]. Finally, a position

$u \in \mathcal{O}(t)$ is a *sort direction* when it is a potential but not a structure direction, *i.e,* $t/u = \Omega^\sigma$, u is a potential direction and $SortAccept(P, t, u) \neq \emptyset$.

> *(Sequentiality)* The predicate P is said to be *sequential at* t if and only if, whenever $P(t) = false$ and there exists h such that $t \sqsubseteq h$ and $P(h) = true$, then there is either a structure or a sort direction of P in t. Finally, P is *sequential* (resp. *sort sequential*) iff it is sequential at every Ω-normal form (resp. at every well-sorted Ω-normal form).

This notion turns out to be the same as that in [HL79] when the subsort order is empty, and sort directions do not make sense.

Consider for example the predicate $\mathcal{NF}_\mathcal{R}$ over Ω-terms, where $\mathcal{R}$ is a term rewriting system and $\mathcal{NF}_\mathcal{R}(t) = true$ iff $t \rightarrow^*_\mathcal{R} u$ and u is in normal form (without redexes nor Ω's). The *system* $\mathcal{R}$ is said to be *sequential* iff $\mathcal{NF}_\mathcal{R}$ is a sequential predicate. This kind of sequentiality is undecidable [HL79], as the corresponding directions cannot in general be found effectively even if a given system is known to be sequential. Sequentiality is therefore a too general property for practical applications.

5 Strong Sequentiality

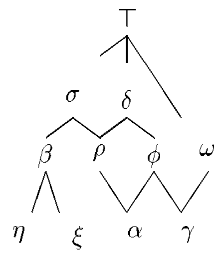
Huet-Levy showed in [HL79] that a restricted notion of sequentiality, called *strong sequentiality* was more appropriate. The technical formulation is based only on the left-hand sides of rules by using the idea of arbitrary reduction sequences, *i.e,* sequences in which a redex is replaced at each step by an arbitrary ground term.

Let t be an Ω-term, $\mathcal{L}$ a constructor base and $Match_\mathcal{L}$ a monotonic predicate defined as $Match_\mathcal{L}(t) = true$ if and only if there exists $l \in \mathcal{L}$ such that $l \sqsubseteq t$. Huet-Levy define a *base* $\mathcal{L}$ *to be match sequential* iff $Match_\mathcal{L}$ is sort-sequential at every Ω-normal form. Strong sequentiality implies match sequentiality (as defined in [HL79]) in the general case of orthogonal systems, although it coincides with the latter in the case of constructor systems as it is shown in [Tha87]. Since in this paper we restrict our attention to constructor systems and the formulation of match sequentiality is much simpler than that of strong sequentiality, we define directly a *base* $\mathcal{L}$ *to be strongly sequential* (resp. *strongly sort sequential*) iff $Match_\mathcal{L}$ is sort-sequential at every Ω-normal form (resp. at every well-sorted Ω-normal form).

We define also $Comp_\mathcal{L}(t) = \{l \sqcup t \mid l \in \mathcal{L} \text{ and } l \uparrow t\}$. For every $u \in \mathcal{O}(t)$, the set $Comp_\mathcal{L}(t, u)$ denotes the subset of $Comp_\mathcal{L}(t)$ such that $k \in Comp_\mathcal{L}(t, u)$ if $t/u \sqsubset k/u$. In order to distinguish the terms appearing in $Comp_\mathcal{L}(t, u)$ we define:

$$StrucComp_\mathcal{L}(t, u) = Comp_\mathcal{L}(t, u) \cap \{l[f(t_1 \ldots t_n)]_u\}$$
$$SortComp_\mathcal{L}(t, u) = Comp_\mathcal{L}(t, u) \cap \{l[\Omega^\eta]_u\}$$

Example 1 *Let* $\mathcal{L} = \{f(\Omega^\eta, \Omega^\sigma)\ f(\Omega^\rho, a)\ f(\Omega^\phi, b)\ f(a, \Omega^\sigma)\ f(\Omega^\omega, a)\}$*,* t *the* Ω*-term* $f(\Omega^\top, \Omega^\sigma)$*, and* Σ *the following signature:*



$c :\to \gamma$
$b :\to \eta$
$a :\to \xi$

$p : \eta \times \rho \to \rho$
$q : \xi \times \rho \to \rho$
$f : \sigma \times \sigma \to \phi$
$f : \beta \times \rho \to \alpha$
$h : \top \times \top \times \top \to \gamma$

$Comp_{\mathcal{L}}(t) = Comp_{\mathcal{L}}(t,1) = \mathcal{L}$
$StrucComp_{\mathcal{L}}(t,1) = \{f(a,\Omega^{\sigma})\}$
$SortComp_{\mathcal{L}}(t,1) = \{f(\Omega^{\eta},\Omega^{\sigma}),\ f(\Omega^{\rho},a),\ f(\Omega^{\phi},b)\ f(\Omega^{\omega},a)\}$
$Comp_{\mathcal{L}}(t,2) = StrucComp_{\mathcal{L}}(t,2) = \{f(\Omega^{\rho},a),\ f(\Omega^{\phi},b),\ f(\Omega^{\omega},a)\}$
$SortComp_{\mathcal{L}}(t,2) = \emptyset$

Remark 1 *$Comp_{\mathcal{L}}(t) \subseteq Accept(Match_{\mathcal{L}},t)$.*

Lemma 1 *$\forall h \in Accept(Match_{\mathcal{L}},t)$, $\exists k \in Comp_{\mathcal{L}}(t)$ such that $k \sqsubseteq h$.*

Proof. Take any $h \in Accept(Match_{\mathcal{L}},t)$. Then, $Match_{\mathcal{L}}(h) = true$ (*i.e.* $\exists l \in \mathcal{L}$ such that $l \sqsubseteq h$) and since $t \sqsubseteq h$ we have $t \sqcup l \in Comp_{\mathcal{L}}(t)$ and $t \sqcup l \sqsubseteq h$.

Lemma 2 *A position u is a potential direction of $Match_{\mathcal{L}}$ in $t[\Omega^{\sigma}]_u$ if and only if $Comp_{\mathcal{L}}(t,u) = Comp_{\mathcal{L}}(t)$.*

Proof. Let u be a potential direction of $Match_{\mathcal{L}}$ in $t[\Omega^{\sigma}]_u$ and let $l \in \mathcal{L}$ such that $l \uparrow t$. Then $l \sqcup t$ is in $Comp_{\mathcal{L}}(t)$ and so in $Accept(Match_{\mathcal{L}},t)$. Since u is a potential direction, $t/u \sqsubset (l \sqcup t)/u$ and then $l \sqcup t$ is also in $Comp_{\mathcal{L}}(t,u)$.

Conversely, let $h \in Accept(Match_{\mathcal{L}},t)$. By lemma 1, there is $k \in Comp_{\mathcal{L}}(t)$ such that $k \sqsubseteq h$ and hypothesis k is also in $Comp_{\mathcal{L}}(t,u)$. Therefore $t/u \sqsubset k/u$, by transitivity $t/u \sqsubset h/u$, and thus u is a potential direction.

Lemma 3 *A position u is a structure direction of $Match_{\mathcal{L}}$ in $t[\Omega^{\sigma}]_u$ if and only if $StrucComp_{\mathcal{L}}(t,u) = Comp_{\mathcal{L}}(t)$.*

Proof. Let u be a structure direction and let $l \in \mathcal{L}$ such that $l \uparrow t$. Since $l \sqcup t$ belongs to $Comp_{\mathcal{L}}(t)$ it also belongs to $Accept(Match_{\mathcal{L}},t)$, by hypothesis $Accept(Match_{\mathcal{L}},t) = StrucAccept(Match_{\mathcal{L}},t,u)$ and therefore $(l \sqcup t)|_u = f$ and $l \sqcup t$ belongs also to $StrucComp_{\mathcal{L}}(t,u)$.

Conversely, let $h \in Accept(Match_{\mathcal{L}},t)$. By lemma 1, $\exists k \in Comp_{\mathcal{L}}(t)$ such that $k \sqsubseteq h$ and by hypothesis k is also in $StrucComp_{\mathcal{L}}(t,u)$ and therefore $(l \sqcup t)|_u = f$. As $k \sqsubseteq h$, $h|_u = f$, and we conclude that $h \in StrucAccept(Match_{\mathcal{L}},t,u)$.

Lemma 4 *A position u is a subsort direction of $Match_{\mathcal{L}}$ in t if and only if $t/u = \Omega^{\sigma}$, $Comp_{\mathcal{L}}(t,u) = Comp_{\mathcal{L}}(t)$ and $SortComp_{\mathcal{L}}(t,u) \neq \emptyset$.*

Proof. Let u be a subsort direction of $Match_{\mathcal{L}}$ in t. By hypothesis $t/u = \Omega^{\sigma}$, and by lemma 2, $Comp_{\mathcal{L}}(t,u) = Comp_{\mathcal{L}}(t)$. Since u is a subosrt direction, $\exists h \in SortAccept(Match_{\mathcal{L}},t,u)$ and in particular h is in $Accept(Match_{\mathcal{L}},t)$. By lemma 1, there is $k \in Comp_{\mathcal{L}}(t)$ such that $k \sqsubseteq h$. Since u is a potential direction, k is also in $Comp_{\mathcal{L}}(t, u)$ (*i.e.* $t/u \sqsubset k/u$) and since h/u is of the form Ω, k/u is also an Ω.

Conversely, by hypothesis $t/u = \Omega^{\sigma}$ and by lemma 2, u is a potential direction of $Match_{\mathcal{L}}$ in t. By hypothesis $\exists k \in SortComp_{\mathcal{L}}(t, u)$. In particular k belongs to $Comp_{\mathcal{L}}(t)$ and so it belongs to $Accept(Match_{\mathcal{L}}, t)$. Since k/u is an Ω and $t/u \sqsubset k/u$, then $k \in SortAccept(Match_{\mathcal{L}}, t, u)$.

Example 2 *Let $\mathcal{L}$ be the base of example 1. We have that 1 is a sort direction of $Match_{\mathcal{L}}$ in $f(\Omega^{\sigma}, \Omega^{\sigma})$, $f(\Omega^{\rho}, b)$ and $f(\Omega^{\phi}, a)$, while 2 is a structure direction of $Match_{\mathcal{L}}$ in $f(\Omega^{\rho}, \Omega^{\sigma})$, $f(\Omega^{\phi}, \Omega^{\sigma})$ and $f(\Omega^{\omega}, \Omega^{\sigma})$.*

On the other hand, 2 is not a direction of $Match_{\mathcal{L}}$ in $f(\Omega^{\sigma}, \Omega^{\sigma})$ and 1 is neither a direction of $Match_{\mathcal{L}}$ in $f(\Omega^{\rho}, \Omega^{\sigma})$ nor in $f(\Omega^{\phi}, \Omega^{\sigma})$ nor in $f(\Omega^{\omega}, \Omega^{\sigma})$.

6 Pattern Matching Trees

The decision procedure of strong sequentiality in untyped constructor systems can also be used to build pattern matching algorithms [Tha87]. However, this is not the case in the order-sorted framework: the difficulty lies in the definition of the order-signature. In this paper we do not discuss the properties that a signature has to verify in order to allow a really sequential pattern matching process; this notion can be found in [Kes91]. Our interest is to discuss strongly sorted sequential systems, and to characterize those algorithms that do not take into account any property of the signature. We assume there is an orthogonal mechanism of type checking that takes place before the pattern matching process does and therefore, we study those algorithms that allow to perform sequential searching of directions at every *well-typed* Ω-normal form. We first define pattern matching trees and we give an intuitive outline of its behavior.

A *pattern matching tree (PMT)* associated to a base $\mathcal{L}$ is defined as:

- Each node is an Ω-pattern.
- If h is an internal node, it has a distinguished position u (denoted $h\ \{u\}$).
- $\Omega^{\top}\{\epsilon\}$ is the root
- If $\mathcal{L}'$ is the set of leaves, $\mathcal{L} \subseteq \mathcal{L}'$ and $\forall l' \in \mathcal{L}'\ \exists l \in \mathcal{L}$ such that $l \sqsubseteq l'$.
- If p_i is a child of $p\ \{u\}$, then p_i and p only differs at position u and $p/u \sqsubset p_i/u$.

PMT's as defined here may be non-deterministic, since different children of a given node are not necessary incompatible. This non-determinism only happens because of sort directions, and it is one of the main differences with classical pattern matching trees [PS90]. Since we are interested in strong sort sequentiality, nodes of a PMT are also required to be *well-sorted.* A PMT associated to a base $\mathcal{L}$ can also be seen as a non-deterministic case-expression, where each step of the tree is seen as a constraint to be satisfied by the term to be matched. The evaluation of a term m is performed by $Match_{\mathcal{L}}(m, \Omega^{\top}, \epsilon)$, where the procedure $Match_{\mathcal{L}}$ is defined as follows:

When p is a leaf, the procedure terminates with a succesful result:

$Match_{\mathcal{L}}(t, p, \epsilon) = true$

When $p_1 \{u_1\}, \ldots, p_k \{u_k\}$ are the children of a node $p \{u\}$, the reduction of the subterm at position u of the term to be matched is produced:

$Match_{\mathcal{L}}(t, p, u) =$
  case t/u **of:**
  $p_1/u \sqsubseteq t/u :\Rightarrow Match_{\mathcal{L}}(t, p_1, u_1)$
  $\vdots$
  $p_k/u \sqsubseteq t/u :\Rightarrow Match_{\mathcal{L}}(t, p_k, u_k)$
  otherwise: **if** $Normal\_Form(t/u)$
    then $false$
    else $Match_{\mathcal{L}}(t[u \leftarrow Reduction(t, u)], p, u)$

$Reduction(t, u) =$
  case u **of:**
  u is a sort direction: One_Step_Reduction(t/u)
  u is a structure direction: Head_Normal_Form(t/u)

In sort decreasing systems, if a term t reduces to another term t', the sort of the second is smaller or equal than that of the first. Therefore, a one-step reduction may possibly refine the sort information of a term in order to satisfy a subsort constraint (the sort direction case). On the other hand, since only constructor function symbols are allowed inside left-hand sides of rules, the head-normal form will be always required in order to satisfy a structure constraint (the structure direction case).

In the untyped case, the well known example of Berry, consisting of the non-strongly sequential base $\mathcal{L} = \{f(a, a, \Omega), f(b, \Omega, a), f(\Omega, b, b)\}$, shows how unnecessary verifications lead to non optimal PMT's. Let t_1, t_2 and t_3 be three redexes in $t = f(t_1, t_2, t_3)$. The choice of normalizing t_1 before t_2 and t_3 fails to reach a matching for the term t, if t_1 has no normal form and t_2 and t_3 have both normal forms b. The same reasoning applies if t_2 or t_3 are chosen to be normalized first. As [HL79] have shown, there is no optimal pattern matching algorithm associated to $\mathcal{L}$.

In the order-sorted case, the following set illustrates a non strongly sequential base for which there is no optimal pattern matching algorithm. Let us consider the set $\mathcal{L}_1 = \{h(\Omega^\top, \Omega^\eta, \Omega^\xi), h(\Omega^\xi, \Omega^\top, \Omega^\eta), h(\Omega^\eta, \Omega^\xi, \Omega^\top)\}$, a slight modification of Berry's example, where the subsort order is the same of example 1. Let t_1, t_2 and t_3 be three redexes in $t = f(t_1, t_2, t_3)$. The choice of normalizing t_1 before t_2 and t_3 fails to reach a matching for the term t, if t_1 has no normal form and t_2 and t_3 have normal forms b and a respectively. The same reasoning applies if t_2 or t_3 are chosen to be normalized first.

The algorithm below can be seen as an implicit method of construction of PMT's associated to $\mathcal{L}$. The main idea is to find directions at some well-sorted terms that are prefixes of $\mathcal{L}$. $SortSeq(\mathcal{L}, t)$ proceeds as follows: if there is $l \in \mathcal{L}$ such that $l \sqsubseteq t$, return true. Otherwise, try to find a direction u of $\mathcal{L}$ in t. If such a direction cannot be found, $\mathcal{L}$ is not strongly sequential at t and then fail. Otherwise, a set of terms called STEPS is computed and strong sequentiality is verified recursively at the well-sorted terms expanded by substituting the subterm at position u of t by a well-sorted term in $\textsc{steps}$. We denote as $g(\overline{\eta_k})$ the term $g(\Omega^{\eta_1}\ldots\Omega^{\eta_k})$.

$SortSeq(\mathcal{L},t) =$
if $\exists l \in \mathcal{L}$ such that $l \sqsubseteq t$,
then $true$
else if $\exists u \in \mathcal{O}(t)$ such that
$StrucDirec(\mathcal{L},t,u) = [true, \textsc{steps}]$ **or** $SortDirec(\mathcal{L},t,u) = [true, \textsc{steps}]$
then $\bigwedge_{p \in STEPS} SortSeq(Comp_{\mathcal{L}}(t[u \leftarrow p]), t[u \leftarrow p])$
else $false$

$StrucDirec(\mathcal{M},t,u) =$
if $(t/u = \Omega^{\sigma})$ **and** $(\forall l \in \mathcal{M},\ l|_u$ is a constructor$)$
then $[true, min\{f(\overline{\eta_k}) \mid f : \overline{\eta_k} \rightarrow \eta \in \Sigma$ and $\eta \preceq \sigma$ and $\exists l \in \mathcal{M},\ l|_u = f\}]$
else $[false, \emptyset]$

$SortDirec(\mathcal{M},t,u) =$
if $(t/u = \Omega^{\sigma})$ **and** $(\forall l \in \mathcal{M},\ t/u \sqsubset l/u)$ **and** $(\exists l \in \mathcal{M},\ l/u = \Omega^{\eta})$
then $[true, min\{l/u \mid l \in \mathcal{M}\}]$
else $[false, \emptyset]$

Example 3 *Let* $\mathcal{L}_1 = \{f(\Omega^{\beta}, p(b,\Omega^{\rho})),\ f(a, q(a,\Omega^{\rho})),\ f(\Omega^{\rho},\Omega^{\sigma})\}$, *where* Σ_1 *is the signature of example 1. Figure 1 shows a PMT for* $\mathcal{L}_1$.

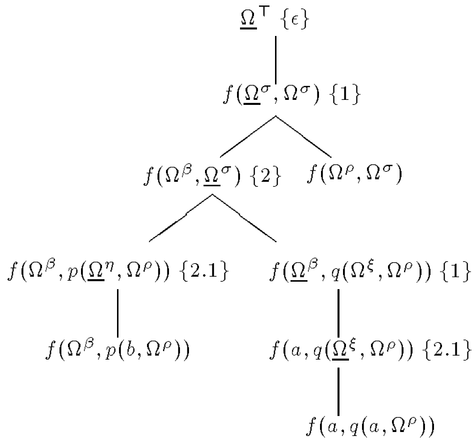


Figure 1: Pattern Matching Tree for $\mathcal{L}_1$

Example 4 *Let* Σ_2 *be signature containing the subsort order of example 1 and the following set of declarations:*

$$
\begin{array}{llll}
f : \delta \times \sigma \times \top \rightarrow \delta & f : \beta \times \sigma \times \top \rightarrow \delta & m :\rightarrow \gamma & k :\rightarrow \eta \\
f : \delta \times \sigma \times \top \rightarrow \phi & f : \beta \times \sigma \times \top \rightarrow \rho & l :\rightarrow \xi &
\end{array}
$$

Let $\mathcal{L}_2 = \{f(k, \Omega^\sigma, \Omega^\top),\ f(l, \Omega^\sigma, \Omega^\top),\ f(\Omega^\rho, k, \Omega^\top)\ f(\Omega^\rho, l, m)\ f(\Omega^\phi, l, k)\}$. We show a PMT for $\mathcal{L}_2$ in figure 2.

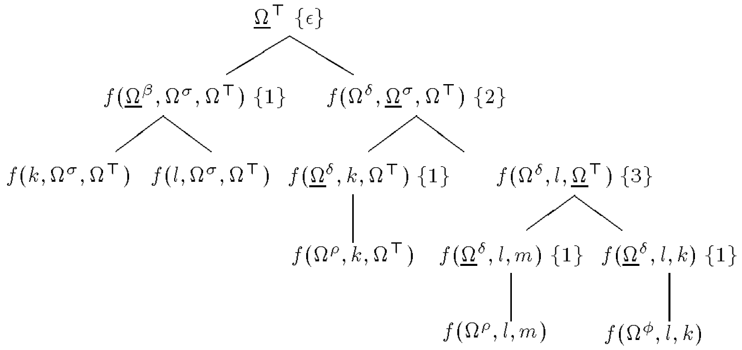


Figure 2: Pattern Matching Tree for $\mathcal{L}_2$

Remark 2 *If t is a node of a PMT associated to $\mathcal{L}$ and $t_1 \dots t_m$ its children, then $\forall l \in \mathcal{L}$ such that $l \uparrow t$, $\exists 1 \le i \le m$, $t_i \sqsubseteq l$.*

Theorem 1 *If $\mathcal{L}$ is strongly sequential, then $SortSeq(\mathcal{L}, \Omega^\top) = true$.*

Proof. If $\mathcal{L}$ is strongly sequential, $Match_\mathcal{L}$ has a structure or a sort direction at every term in Ω-normal form. In particular, every node constructed by $SortSeq(\mathcal{L}, \Omega^\top)$ is a well-sorted Ω-normal form and thus $SortSeq(\mathcal{L}, \Omega^\top) = true$.

7 Free Sequentiality

In order-sorted rewriting systems, sort sequentiality is related to the definition of order-sorted signatures, which introduces complex situations that are well illustrated by the following example. Let $\sigma, \eta, \delta, \beta$ be four pairwise disjoint sorts and g a constructor symbol declared as:

$$\begin{array}{ll} g : \top \times \sigma \times \eta \to \delta & \quad g : \top \times \top \times \top \to \top \\ g : \eta \times \top \times \sigma \to \delta & \quad h : \top \to \top \\ g : \sigma \times \eta \times \top \to \delta & \quad h : \delta \to \top \end{array}$$

Let $\mathcal{L}_3 = \{h(\Omega^\delta)\}$ be a base and $t = h(g(\Omega^\top, \Omega^\top, \Omega^\top))$. We have $Match_{\mathcal{L}_3}(t) = false$ but there exists an instance of t such that $Match_{\mathcal{L}_3}$ becomes true. If $\mathcal{L}_3$ is strongly sequential a direction of $Match_{\mathcal{L}_3}$ in t must be found, and this direction does not depend only on the predicate, but on the signature Σ. In the example, choosing a direction of $Match_{\mathcal{L}_3}$ at $h(g(\Omega^\top, \Omega^\top, \Omega^\top))$ is equivalent to choosing a direction of $WellSorted_\delta$ at $g(\Omega^\top, \Omega^\top, \Omega^\top)$, where the last predicate is defined as $WellSorted_\delta(t) = true$ if and only if t is of sort δ. But the set of domains of the three last declarations of g is just Berry's example on signatures and this set is known to be not strongly sequential [Ber78], so we cannot find a direction. Thus, we need a notion of "sequential order-sorted signatures" and sequential type-checking, and moreover, such notion must be independent of any predicate. In this paper we want to get rid of the property of sequentiality of signatures, and so instead of restricting the class of signatures to sequential ones, we forbid the declarations of constructors that allow the expansion of terms that do not lead to an immediate realization of the match predicate. A reader interested in sequential signatures can be referred to [Kes92].

We need a stronger notion of sort direction in some terms of the PMT in order to avoid such problems. Let us recall that $g(\overline{\eta_k})$ denotes $g(\Omega^{\eta_1}\dots\Omega^{\eta_k})$. We say that t has the *free sort expression (FSE)* property at position u if and only if $\forall g : \overline{\eta_k} \rightarrow \eta \in \Sigma$ such that $t/u \sqsubseteq g(\overline{\eta_k})$, $t[g(\overline{\eta_k})]_u$ is well-sorted and $t[g(\overline{\eta_k})]_u \uparrow \mathcal{L}$, $\exists h \in Comp_{\mathcal{L}}(t)$ such that $h/u \sqsubseteq g(\overline{\eta_k})$. By construction of the tree, every internal node with a structure direction as distinguished position has the FSE property, but this is not true in general for every internal node of the tree. The FSE property is clearly decidable because the set of declarations of the signature is finite.

> *(Free Sequentiality)* A base $\mathcal{L}$ is *free sequential* iff $SortSeq(\mathcal{L}, \Omega^{\top}) = true$ and every node of the PMT constructed by $SortSeq(\mathcal{L}, \Omega^{\top})$ has the FSE property.

Note that $\mathcal{L}_1$ of example 3 is not free sequential since for example $f(\Omega^{\sigma}, \Omega^{\sigma})$ has not the FSE property because of the declaration $f : \sigma \times \sigma \rightarrow \phi$. If we replaced $f : \sigma \times \sigma \rightarrow \phi$ by $f : \sigma \times \sigma \rightarrow \alpha$ in the signature Σ of example 1, every node has the FSE property and thus $\mathcal{L}_1$ is free sequential. The base $\mathcal{L}_2$ of example 4 is free sequential and the base $\mathcal{L}_3$ presented at the beginning of this section does not: the node $h(\Omega^{\delta})$ has not the FSE property because of the declaration $g : \top \times \top \times \top \rightarrow \top$.

Theorem 2 *If $\mathcal{L}$ is free sequential, then $\mathcal{L}$ is strongly sort sequential.*

Proof. Let m be a well-sorted Ω-normal form such that $Match_{\mathcal{L}}(m) = false$ and $m \uparrow \mathcal{L}$. As $\Omega^{\top}$ is the root of the PMT but m does not match any pattern of $\mathcal{L}$, there exists a node t with children $t_1 \dots t_k$ and distinguished position u such that $t \sqsubseteq m$ and for each $1 \leq i \leq k$, $t_i \not\sqsubseteq m$. Since t differs only at position u which each t_i, then

$$\forall 1 \leq i \leq k,\ t_i/u \not\sqsubseteq m/u \tag{1}$$

We will first prove that u is a potential direction of $Match_{\mathcal{L}}$ in m:

a) Suppose $m/u = g(l_1 \dots l_n)$. Since m/u is well-sorted and $t/u = \Omega^{\sigma} \sqsubseteq m/u$, there is a declaration $g : \overline{\eta_k} \rightarrow \eta$ such that $g(\overline{\eta_k}) \sqsubseteq m/u$ and $\eta \preceq \sigma$. Then $\Omega^{\sigma} \sqsubseteq g(\overline{\eta_k})$ and $t[g(\overline{\eta_k})]_u$ is well-sorted. Therefore, $t[g(\overline{\eta_k})]_u \sqsubseteq m[g(\overline{\eta_k})]_u \sqsubseteq m$ and since $m \uparrow \mathcal{L}$, also $t[g(\overline{\eta_k})]_u \uparrow \mathcal{L}$. By hypothesis, t has the FSE property at position u and so $\exists l \in Comp_{\mathcal{L}}(t)$ such that $l/u \sqsubseteq g(\overline{\eta_k})$. By construction of the tree, $\exists i$ such that $t_i \sqsubseteq l$, and thus $t_i/u \sqsubseteq g(l_1 \dots l_n) = m/u$ holds by transitivity which leads to a contradiction with equation 1. Then m/u has the form Ω^{δ} with $\delta \preceq \sigma$.

b) Let $l \uparrow m$ and suppose $m/u = (l \sqcup m)/u$. As $t \sqsubseteq m$, also $l \uparrow t$ and by construction of the tree, $\exists i$ such that $t_i/u \sqsubseteq (l \sqcup t)/u$. Therefore $t_i/u \sqsubseteq$

$(l \sqcup t)/u \sqsubseteq (l \sqcup m)/u = m/u$ which leads to a contradiction with equation 1.

Thus $m/u \sqsubset (l \sqcup m)/u$ and u is a potential direction of $Match_{\mathcal{L}}$ in m.

By definition, if u is a potential direction, it is either a structure or a sort direction and this is sufficient to show that $\mathcal{L}$ is strongly sort sequential. But in particular we can add that, whenever u is a structure direction of $Match_{\mathcal{L}}$ in t, it is also a structure direction of $Match_{\mathcal{L}}$ in m, and whenever u is a sort direction of $Match_{\mathcal{L}}$ in t, it may be either a sort or a structure direction of $Match_{\mathcal{L}}$ in m.

The intuitive idea of this result is that whenever a system is free sequential and we supply an orthogonal mechanism of type checking, pattern matching can be efficiently performed, by avoiding unnecessary reductions to head normal forms.

As we have noticed above, the FSE property always holds in nodes having a structure direction and, if the system is untyped, sort directions do not make sense. Therefore our result of theorem 2 can be seen as a generalization of that in [Tha87], that in terms of our formalism can be stated as follows:

Corollary 1 *For every untyped base such that $SortSeq(\mathcal{L}, \Omega^{\top}) = true$, $\mathcal{L}$ is strongly sequential.*

The problem we have encountered with order-sorted systems is that there is a "hidden" concept of sequentiality in the presentation of their signatures. Each time there is a sort constraint to be satisfied, the type checker needs a sequential mechanism to verify such constraint and this leads to the problem of defining some kind of "sequential type checking" which is also related to the declarations of function symbols. Thus, we can have a sequential signature but a non strongly sequential set of rules, or a strongly sequential set of rules but a sequential signature. While strong sequentiality is equivalent to optimal computations in orthogonal untyped rewriting systems with constructors, a combination of sequential matching (strongly sequential bases) and sequential type checking will be necessary in the order-sorted case in order to have a complete result of effective computations.

Conclusions

We have introduced the notion of sequentiality and strong sequentiality in order-sorted rewriting systems and we have characterized via a decidable property those orthogonal constructor systems which admit an easy and simple implementation. Thanks to the formalism of order-sorted signatures as bottom-up tree automata, we simplify and avoid many restrictions that often appear in the treatment of order-sorted signatures, such as regularity, downward completeness and coregularity. We think that this is a first step towards efficient implementation of order-sorted rewriting systems. In this paper we suppose that an orthogonal mechanism of type verification is given, *i.e*, terms to be evaluated are always well-sorted. Future work will focus on sequential order-sorted signatures and sequential type checking.

Acknowledgements

I would like to thank Jean-Pierre Jouannaud and Huber Comon for stimulating suggestions that motivate many extensions of my previous work. I am very grateful to Laurence Puel for fruitful discussions and helpful improvements in the final version of the paper. I would also like to thank Roberto Di Cosmo and Chet Murthy for their help in the presentation of the work.

References

[Ber78] Gérard Berry. Séquentialité de l'évaluation formelle des lambdas-expressions. *Proc. 3rd International Colloquium on Programming*, Paris, France, 1978.

[CD91] Hubert Comon and Catherine Delor. Equational Formulas with Membership Constraints. Technical Report 649, Laboratoire de Recherche en Informatique, 1991. To appear in Information and Computation.

[GS84] Ferenc Gécseg and Magnus Steinby. Tree Automata. Akademiai Kiado, Budapest, 1984.

[HL79] Gérard Huet and Jean-Jacques Lévy. Call by need computations in non ambiguous linear term rewriting systems. Technical Report IRIA Laboria 359, INRIA, Le Chesnay, France, 1979.

[Kes91] Delia Kesner. Pattern Matching in Order-Sorted Languages. In *Mathematical Foundations of Computer Science*. LNCS 520, Springer-Verlag. Also in PRL Research Report Number 10, Digital Equipment Corporation, Paris Research Laboratory, 1991.

[Kes92] Delia Kesner. Sequential signatures. Draft, 1992.

[KM91] Jan Willem Klop and Aart Middeldorp. Sequentiality in orthogonal term rewriting Systems. In *J. Symbolic Computation*. Academic Press, 1991.

[PS90] Laurence Puel and Ascánder Suárez. Compiling Pattern Matching by Term Decomposition. In *1990 ACM Conference on Lisp and Functional Programming*. ACM Press. Also in PRL Research Report Number 4, Digital Equipment Corporation, Paris Research Laboratory, 1990.

[SNGM89] Gert Smolka, Werner Nutt, Joseph Goguen, and José Meseguer. Order Sorted Equational Computation. In Hassan Aït-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures. Volume 2: Rewriting Techniques*, pages 297–367. Academic Press, 1989.

[Tha87] Satish Thatte. A Refinement of Strong Sequentiality for Term Rewriting systems with Constructors. *Information and Computation*, Number 72, 1987.