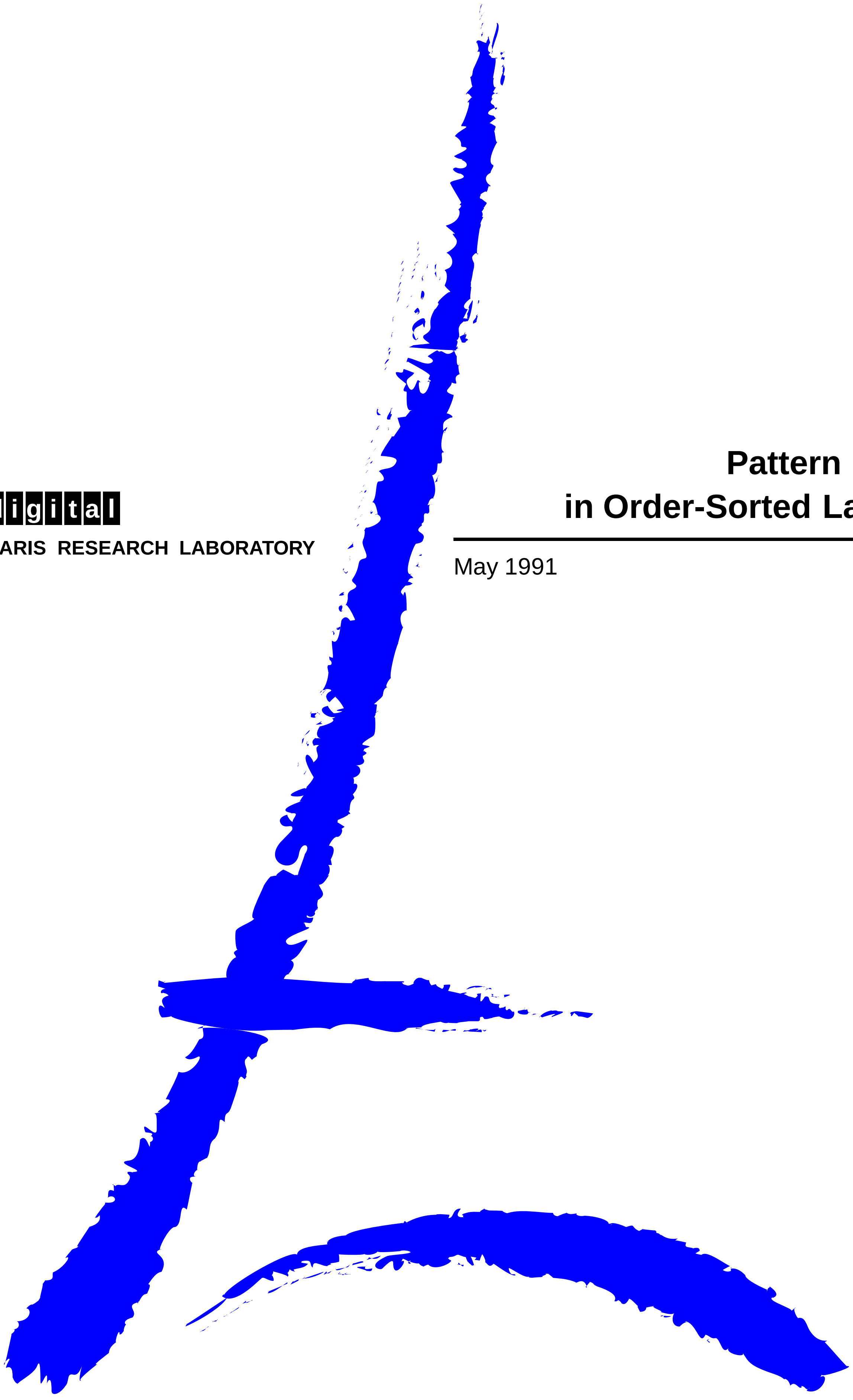
10



digital

PARIS RESEARCH LABORATORY

Pattern Matching in Order-Sorted Languages

May 1991

Delia Kesner

10

Pattern Matching in Order-Sorted Languages

Delia Kesner

May 1991

Publication Notes

This work was initiated by the author while she was a research intern at Digital's Paris Research Laboratory from April to September 1990 working on her *stage de DEA* for the *Université de Paris 7*. It was continued later at INRIA, Rocquencourt, and at the *Laboratoire de Recherche en Informatique de l'Université d'Orsay* (LRI). This document appears simultaneously as LRI Rapport de Recherche N^o 667 (5/1991). A short version of this paper bearing the same title will appear as part of the proceedings of the 16th International Symposium on Mathematical Foundations of Computer Science (MFCS'91) to be held in Warsaw, Poland, in September 1991.

For correspondence, the author may be contacted at the following addresses:

INRIA Rocquencourt	LRI and CNRS UA 410
Domaine de Voluceau, BP 105	Bât 490, Université de Paris-Sud
78153 Le Chesnay Cedex	91405 Orsay Cedex
France	France
`kesner@inria.inria.fr`	`kesner@lri.lri.fr`

Abstract

We study the problem of pattern matching in order-sorted languages whose evaluation strategy is lazy. We propose an extension of the Puel-Suárez compilation scheme to function definitions via order-sorted patterns. Basically, a list of ordered and possibly ambiguous linear patterns is transformed into a set of disjoint order-sorted constrained terms. This set is in turn transformed according to some normalization rules in order to build a pattern matching tree (PMT). Variables of order-sorted constrained terms now have not only structure, but also subsort constraints. Accordingly, discrimination trees are defined to have edges labeled with either structure or subsort constraints. Due to this latter kind of edge, we are not always forced to reduce terms to normal forms during the pattern matching process, taking advantage in this way of the lazy reduction scheme. For example, suppose σ is a sort greater than η, the variable x^η is a pattern and t^σ is a term of sort σ to be matched. If t^σ reduces to a term whose sort is a subsort of η, it is already decidable that the term obtained matches x^η, even if it is not in normal form. We show that the PMT is optimal if a decidable property of sequentiality holds for the sets generated during the compilation process. Our method turns out to be applicable for strict languages as well.

Résumé

Nous étudions le problème du filtrage dans des langages avec sous-sortes et dont la stratégie d'évaluation est paresseuse. Nous proposons une extension d'un schéma dû à Puel et Suárez pour la compilation des définitions de fonctions basées sur des motifs à sortes partiellement ordonnées. En résumé, une séquence ordonnée de motifs linéaires potentiellement ambigus est transformée en un ensemble de termes contraints à sortes ordonnées qui sont mutuellement exclusifs. Cet ensemble est ensuite transformé à l'aide de règles de normalisation permettant de construire un arbre de filtrage. Les variables des termes contraints à sortes ordonnées sont ici soumises à des contraintes non seulement de structure, mais aussi de sous-sortes. Pour refléter cela, un arbre de filtrage est doté d'arêtes étiquetées par des contraintes de structure ou de sous-sorte. Grâce à ce dernier genre d'arêtes, il n'est pas toujours nécessaire de réduire les termes en forme normale pendant le processus de filtrage, et donc de bénéficier de cette manière de la réduction paresseuse. Par exemple, supposons que σ soit une sorte supérieure à η, que la variable x^η soit un motif et que t^σ, un terme de sorte σ, soit à filtrer. Dès que t^σ est réduit à un terme de sorte inférieure ou égale à η, il est d'ores et déjà décidable que le terme ainsi obtenu est filtré par x^η, même s'il n'est pas en forme normale. Nous montrons que l'arbre de filtrage est optimal si une propriété décidable de séquentialité est vérifiée par les ensembles engendrés durant la compilation. Notre méthode s'avère également applicable aux langages stricts.

Keywords

Order-sorted constrained term, pattern matching, discrimination tree, compilation, lazy evaluation, sequentiality

Acknowledgements

I wish to thank my three advisors Hassan Aït-Kaci, Laurence Puel and Ascánder Suárez for proposing this topic for my *DEA* , and for many fruitful discussions. I am also very grateful to Kathleen Milsted for her patient help in the whole preparation of the first version of this paper, including proof reading and English translation.

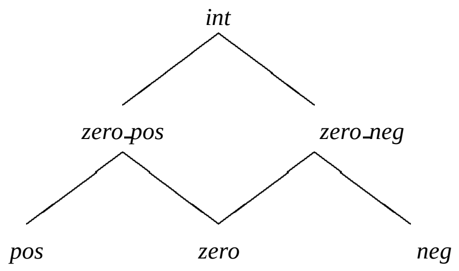
Contents

1 Introduction

Many programming languages use pattern matching in a many-sorted term algebra (such as those in the ML family [6]) or an order-sorted term algebra (such as those in the OBJ family [5]) for function argument-passing. Function definitions consist of an ordered set of rewrite rules. These rules are often ambiguous as some left-hand sides (LHS) of the same function definition may overlap. Thus direct access to the relevant rule based on the LHS's structure is not possible in general. The naïve operational semantics, amounting to sequential lookup until a calling term matches a rule's LHS, obviously leads to poor performance. In addition, since a lazy evaluation strategy allows the manipulation of infinite objects (*i.e.* with no finite constructor normal forms), it is not clear what pattern matching means for lazy languages. For example, for a term to match a LHS term, the reduction scheme should be such that the only part of the term to be evaluated is the one required, in some sense. Recently, Puel and Suárez [10] devised a clever compilation scheme to generate statically a PMT in lazy languages. Such a tree is then used at run-time for fast rule-indexing and takes full advantage of the nature of the LHS terms in a definition. Their work simplified and generalized seminal ideas by Huet and Lévy [7] that were in turn sharpened by Laville [9]. The gist of the Puel-Suárez method rests on generalized notions of constructor terms and sequentiality. They called the new terms **constrained terms**.

Although partially ordered sorts provide a substantially improved expressiveness over many-sorted languages, in an order-sorted system with a lazy reduction strategy, pattern matching is more complex than with non-ordered sorts in that it necessitates two kinds of verifications. The first one, as in the conventional case, is **structure** matching. The other one is to ascertain that the argument's sort is a **subsort** of the formal parameter's sort. Moreover, as functions can have a non-strict semantics, they can yield a result even for some arguments whose evaluation is non-terminating. Therefore, the arguments need only be evaluated just enough so as to make either a structure or subsort verification decidable. However, it is not clear how many steps of reduction must be performed on a given term in order for its sort to become sufficiently precise.

Consider for example the classical subsort order for the integer numbers: where *o* is a constant



$o :\rightarrow zero$
$pred : zero\_neg \rightarrow neg$
$suc : zero\_pos \rightarrow pos$

of sort *zero*, the symbol *pred* is a constructor of sort *neg* and the symbol *succ* is a constructor of sort *pos*.

Let $\mathcal{C}$ be the characteristic function of natural numbers defined by the following rewrite rules.

$\mathcal{C}(x^{zero\_pos}) = 1$

$\mathcal{C}(x^{zero\_neg}) = 0$

Let the order of appearance of these rules be significant as specified. That is, the second rule is to be considered only if the first one is not applicable. Thus, this definition is equivalent to:

$\mathcal{C}(x^{zero\_pos}) = 1$

$\mathcal{C}(x^{neg}) = 0$

Now, suppose a term of sort *int* with non terminating evaluation given by the following sequence of reductions $t^{int} \to t_1^{zero\_pos} \to t_2^{zero\_pos} \to t_3^{zero\_pos} \ldots$ With a strict evaluation strategy, $\mathcal{C}(t^{int})$ is not defined because t^{int} has no denotation. Nevertheless, with a lazy reduction scheme, $\mathcal{C}(t^{int})$ is defined and equal to 1 since t^{int} can be reduced in finitely many steps just so far as necessary to ascertain that it is of sort *zero_pos*. Thus, there is no need always to reduce terms to normal forms during the pattern matching process and pattern matching becomes a non-trivial problem deserving careful attention.

We restrict our interest to syntactic pattern matching.[1] Sorts are partially ordered. Minimal sorts are assumed to be pairwise disjoint and non-minimal sorts are assumed to be the union of their subsorts. Functions can have more than one declaration [4]. We will also restrict our attention to linear LHS terms, *i.e.*, without repeated variables. We lose no expressive power, though we lose some notational convenience.

The order of rules defining a function is significant because LHS terms can be **ambiguous**; that is, they can be unifiable. Since we are considering deterministic languages, a list of terms with priority (a **pattern**) must be constructed. Thus, in order to avoid a more complicated syntax and a burden to the programmer, a disambiguating meta-rule will be necessary to construct such a list. Usually, this is according to the appearance of the rules in the text but any other priority will suffice [9]. This must naturally be taken into account when constructing the PMT. We propose here an extension of the Puel-Suárez compilation scheme that accommodates order-sorted constructor-based function definitions. Our compilation method eliminates ambiguous patterns by introducing order-sorted constrained terms. Moreover, as order-sorted pattern matching consists of two kind of verifications, discrimination trees are now quite complex since some edges are now labeled with sort restrictions.

Given a pattern matching problem S, the **strict set** of S is the set of terms for which every PMT associated to S will fail to terminate and an **optimal PMT** is a PMT that will only fail to terminate on the strict set of S. We show that optimality of an order-sorted PMT is a decidable property equivalent to a generalization of the notions of strong **sequentiality** presented in [7] and [10]. Sequentiality of a pattern matching problem S is the possibility of systematically expanding any term step by step until either it matches a pattern of S or it is clear that a positive

[1]See [8] for a discussion of unification and matching in equational theories.

matching is impossible. Our notion of sequentiality takes not only the structure of terms into account, but also the sort system. We present a more general treatment of pattern matching compilation in which the unsorted and many-sorted languages are particular cases.

The paper is organized as follows. Section 2 presents unitary signatures and function definitions by order-sorted equations. Section 3 defines the syntax and semantics of order-sorted constrained terms and patterns. Section 4 describes our compilation method. This consists of three kinds of rules acting on constrained terms. Invariance and completeness of these rules are given. Finally, in Section 5, the new notion of sequentiality for order-sorted constrained terms is presented. We show that sequentiality and optimality of pattern-matching problems are equivalent. A brief description of order-sorted type systems covered by our work can be found in Section 6.

2 Functions Defined by Order-Sorted Patterns

All the conventional notions regarding substitutions, instantiation, and unification of unsorted terms are readily extended to order-sorted terms [14].

A **signature** $\Sigma = \langle \mathcal{S}, \leq, \mathcal{F}, \mathcal{C}, \mathcal{V}, \mathcal{D} \rangle$ consists of a set of sort symbols $\mathcal{S} = \{\sigma, \eta, \delta, \ldots\}$, a partial order $\leq$ on $\mathcal{S}$, a set of function symbols $\mathcal{F} = \{F, G, H, \ldots\}$, a set of constructor symbols $\mathcal{C} = \{f, g, h, \ldots\}$, a set of $\mathcal{S}$-indexed variables $\mathcal{V} = \{x^\sigma, y^\sigma, z^\sigma, \ldots\}$ with a countably infinite number of variables for each sort symbol σ, and a set of declarations $\mathcal{D}$ of the form $q : \sigma_1 \ldots \sigma_n \to \sigma$ where $q \in \mathcal{F} \cup \mathcal{C}$. We will call $\sigma_1 \ldots \sigma_n$ the **domain** of f and σ its **codomain**. The sets $\mathcal{S}$, $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{V}$ are mutually disjoint. For brevity, we will write $s \in \Sigma$ for any symbol s in $\mathcal{S}$, $\mathcal{F}$, $\mathcal{C}$, $\mathcal{V}$ or $\mathcal{D}$. We use $\vec{\sigma}, \vec{\eta}, \ldots$ to denote possibly empty sequences of sorts. The order $\leq$ is extended componentwise to sequences of the same length in $\mathcal{S}^*$ and is also denoted $\leq$.

Σ-terms are constructed in the usual manner with the additional constraint that they be **well-sorted**. Formally, a variable $x^\sigma \in \Sigma$ is a well-sorted Σ-term of sort η if $\sigma \leq \eta$, and $q(t_1 \ldots t_n)$ is a well-sorted Σ-term of sort η if and only if there exists a declaration $q : \sigma_1 \ldots \sigma_n \to \sigma \in \Sigma$ such that $\sigma \leq \eta$ and for all $1 \leq i \leq n$, t_i is a well-sorted Σ-term of sort σ_i. Where Σ is understood, we will refer simply to terms instead of Σ-terms.

A signature Σ is called **regular** if all terms have a least sort. We may emphasize the fact that a term t has least sort σ by writing it as t^σ. A signature Σ is called **unitary** if it is regular and:

- $(\mathcal{S}, \leq)$ is a boolean lattice with least upper bound operation $\sqcup$, greatest lower bound operation $\sqcap$, greatest element $\top$ and least element $\bot$.[2]
- No function or constructor declaration contains the sort symbol $\bot$.
- (**Minimal codomain sort**) If $f \in \mathcal{C}$, then there exists a declaration $f : \vec{\sigma} \to \sigma \in \Sigma$ and σ is a minimal sort (*i.e.*, if $\delta \leq \sigma$ then $\delta = \bot$ or $\delta = \sigma$).
- (**Disjoint domain sort**) If $f \in \mathcal{C}$ and $f : \sigma_1 \ldots \sigma_n \to \sigma \in \Sigma$ and $f : \eta_1 \ldots \eta_m \to \eta \in \Sigma$ are two different declarations of f in Σ, then $n \neq m$ or $n = m \geq 1$ and $\vec{\sigma}$ and $\vec{\eta}$ are

[2]A lattice $(\mathcal{S}, \leq)$ is said to be boolean iff $\forall \sigma \in \mathcal{S}, \exists\, !\, \sigma^c \in \mathcal{S}$ such that $\sigma \sqcap \sigma^c = \bot$ and $\sigma \sqcup \sigma^c = \top$.

disjoint (*i.e.*, $\exists i, 1 \leq i \leq n, \sigma_i \sqcap \eta_i = \bot$).

In the sequel, we assume all signatures to be unitary. Motivations for considering such signatures are discussed in Section 6.

Constructor terms are those terms that do not contain function symbols. A term is called **linear** if its variables occur at most once and **ground** if it contains no variables.

Pattern matching is a prefix ordering, $\sqsubseteq$, induced by instantiation on constructor terms modulo variable renaming. Formally, $t \sqsubseteq t'$ iff $t' = \theta(t)$, where θ is a substitution. We say that t' **matches** t. Note that when only linear terms are on the left-hand side, then $x^{\sigma} \sqsubseteq t^{\eta}$ if $\eta \leq \sigma$, and $f(t_1 \ldots t_n) \sqsubseteq f(h_1 \ldots h_n)$ if and only if for all $(1 \leq i \leq n)$ $t_i \sqsubseteq h_i$.

Unification is a least upper bound operation for $\sqsubseteq$. We will note the least upper bound of two Σ-terms t and t' as $t \sqcup t'$. Two terms are said to **overlap**, or to be **ambiguous**, if they are unifiable. Regular signatures are finitary unifying and they make order-sorted term unification well-behaved (see [14] for a discussion). If in addition the signature is unitary, then a unique unifier is produced.

A **function definition** is specified by a set of **rewrite rules** $\{F(t_i) = p_i\}_{i=1}^{m}$, where $F \in \mathcal{F}$, the t_i's are (possibly mutually ambiguous) linear constructor terms (the **patterns** of F) and each p_i is a term containing no variables not in t_i. A **program** $\mathcal{P}$ is a set of function definitions.

3 Order-Sorted Constrained Terms

3.1 Syntax

For a signature Σ, we define the syntax and semantics of constructor Σ-terms, Σ-constraints, constrained Σ-terms and Σ-patterns. We will drop the prefix Σ where it is understood.

Let t be a term, l a linear term, σ a sort and $\mathcal{T}$ and $\mathcal{F}$ the two logical constants denoting truth and falsehood, respectively. Then $\mathcal{T}, \mathcal{F}, t : \sigma$ and $t \diamond l$ are **atoms**. A **constraint** is recursively defined as an atom or as $C_1 \vee C_2$ or as $C_1 \wedge C_2$ where C_1,C_2 are constraints. When $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma \in \Sigma$ and $x_1 \ldots x_n$ are pairwise distinct variables, we may write an atom $t \diamond f(x_1^{\sigma_1} \ldots x_n^{\sigma_n})$ as $t \diamond f^{\sigma}$ or $t \diamond f$ if the sorts are clear from the context. We will write $t \diamond \{f_1, \ldots, f_n\}$ for a constraint $t \diamond f_1 \wedge \ldots \wedge t \diamond f_n$ and $A \in C$ for an atom A of the constraint C. The intended interpretation of a **sort constraint** $t : \sigma$ is that it is decidable that term t has sort σ, and the interpretation of a **structure constraint** $t \diamond l$ is that it is decidable that t is structurally different from l.

If t is a constructor term and C a constraint, then $t \mid C$ is a **constrained term**. If t is linear (resp. ground), then $t \mid C$ is a linear (resp. ground) constrained term. A **pattern** is a non-empty list of linear constructor terms $p_1 \ldots p_n$. A **constrained pattern** is a non-empty list of linear constrained terms $P_1 \ldots P_n$.

For brevity, we will refer to either a term or a constraint as an object. The set of **free variables** of an object h, denoted $\mathcal{V}(h)$, is defined as expected except that $\mathcal{V}(t \diamond l) = \mathcal{V}(t : \sigma) = \mathcal{V}(t)$. Substitution, denoted $\theta(h)$, is also defined as expected except that $\theta(t \diamond l) = \theta(t) \diamond l$ and $\theta(t : \sigma) = \theta(t) : \sigma$.

We will also refer to the **restriction** of a constraint C to a set of variables V, denoted $C|_V$. For atomic constraints C, $C|_V = C$ if $\mathcal{V}(C) \subseteq V$, otherwise $C|_V = \mathcal{T}$. For constraints formed from $\vee$ and $\wedge$, the restriction distributes through to their arguments.

3.2 Semantics

With a lazy reduction scheme, functions can yield a result even when applied to arguments whose evaluation is non-terminating. A new element is thus necessary to give semantics to such functions. We will introduce a new symbol $\bullet^\sigma$ for each sort σ in the signature. Formally, an **augmented signature** $\Sigma^\bullet$ is a unitary signature Σ without variables and without function symbols $\{F, G, H, \ldots\}$ but with a 0-ary constructor $\bullet^\sigma$ for each sort symbol $\sigma \in \Sigma$ different from $\perp$. $\bullet^\sigma$ denotes those terms of sort σ that cannot be reduced to a term having a constructor symbol at the root (so-called **head-normal form**). Note that all $\Sigma^\bullet$-terms are ground.

The free order-sorted term algebra on the signature Σ is denoted by $\mathcal{T}_\Sigma$. An interpretation of a signature Σ over its $\Sigma^\bullet$-term algebra $\mathcal{T}_{\Sigma^\bullet}$ satisfies:

- $\sigma^{\mathcal{T}_{\Sigma^\bullet}} := \{s \mid s \text{ is a } \Sigma^\bullet\text{-term of sort } \sigma\}$
- $\perp^{\mathcal{T}_{\Sigma^\bullet}}$ is the empty set and $\top^{\mathcal{T}_{\Sigma^\bullet}}$ the universe of $\mathcal{T}_{\Sigma^\bullet}$
- $\sigma \leq \eta$ implies $\sigma^{\mathcal{T}_{\Sigma^\bullet}} \subseteq \eta^{\mathcal{T}_{\Sigma^\bullet}}$
- If f is a constructor and $f : \sigma_1 \ldots \sigma_n \to \sigma \in \Sigma$, then $f^{\mathcal{T}_{\Sigma^\bullet}}$ is a function $\sigma_1^{\mathcal{T}_{\Sigma^\bullet}} \times \ldots \times \sigma_n^{\mathcal{T}_{\Sigma^\bullet}} \to \sigma^{\mathcal{T}_{\Sigma^\bullet}}$ such that $f^{\mathcal{T}_{\Sigma^\bullet}}(s_1 \ldots s_n) = f(s_1 \ldots s_n)$

Sorts and subsorts sometimes allow us to decide if a term matches a pattern even if its evaluation is non-terminating. We show how to characterize such nice terms.

We associate to each constructor term t three disjoint sets of $\Sigma^\bullet$-terms such that the union of these three sets is the $\mathcal{T}_{\Sigma^\bullet}$-algebra. The first one, denoted by $[\![t]\!]_\mathcal{T}$ or simply by $[\![t]\!]$ and called the **denotation** or **solution** of t, is the set of $\Sigma^\bullet$-terms that are instances of t. The second one, denoted by $[\![t]\!]_\mathcal{U}$ and called the **uncertain** or **strict** set of t, is the set of all $\Sigma^\bullet$-terms for which we cannot decide if they are instances of t. The last, $[\![t]\!]_\mathcal{F}$, is the set of $\Sigma^\bullet$-terms that are not instances of t. Formally,

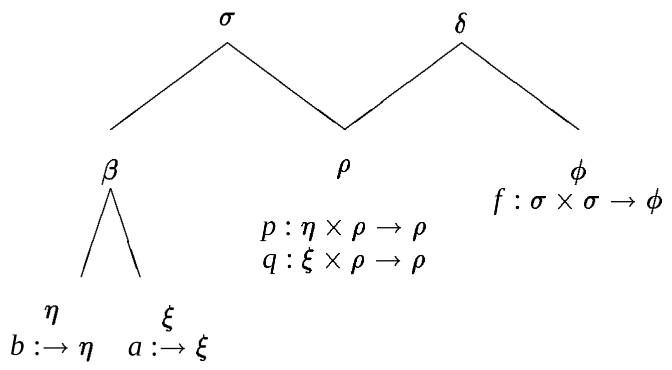
- $[\![t]\!]_\mathcal{T} = \{\theta(t) | \theta \text{ is a } (\mathcal{V}, \Sigma^\bullet)\text{-assignment}\}$
- $[\![t]\!]_\mathcal{U}$ is defined by recursion as:

$$\begin{aligned} [\![x^\sigma]\!]_\mathcal{U} &= \{\bullet^\eta | \eta \sqcap \sigma \neq \perp \text{ and } \eta \not\leq \sigma\} \\ [\![f(t_1 \ldots t_n)^\sigma]\!]_\mathcal{U} &= \{\bullet^\eta | \eta \geq \sigma\} \cup \{f(a_1 \ldots a_n) \mid \exists i\ a_i \in [\![t_i]\!]_\mathcal{U} \text{ and } \forall j\ a_j \notin [\![t_j]\!]_\mathcal{F}\} \end{aligned}$$

In both cases we can decide that $\bullet^\eta$ does not match t^σ when η and σ are disjoint. In the first case, we can decide also that $\bullet^\eta$ matches x^σ when $\eta \leq \sigma$. Under the opposite conditions, we cannot decide and then $\bullet^{\eta} \in [\![x^{\sigma}]\!]_{\mathcal{U}}$. In the second case, σ is a minimal sort (we are dealing with unitary signatures) and so, every $\bullet^{\eta}$ with η comparable with σ (*i.e.* $\eta \geq \sigma$) belongs to the strict set of t. Note also that we can decide that a term with the same constructor symbol does not match if at least one of its arguments does not match.

- $[\![t]\!]_{\mathcal{F}} = \mathcal{T}_{\Sigma^{\bullet}} - [\![t]\!]_{\mathcal{T}} - [\![t]\!]_{\mathcal{U}}$

Example 3.1 Consider the following subsort order:


σ
δ
β
ρ
ϕ
$f : \sigma \times \sigma \to \phi$
$p : \eta \times \rho \to \rho$
$q : \xi \times \rho \to \rho$
η
ξ
$b :\to \eta$
$a :\to \xi$

$$[\![f(x^{\sigma}, b)]\!]_{\mathcal{T}} = \{f(b,b), f(\bullet^{\sigma}, b), f(\bullet^{\eta}, b), f(a,b), \ldots\}$$
$$[\![f(x^{\sigma}, b)]\!]_{\mathcal{U}} = \{f(b,\bullet^{\sigma}), f(\bullet^{\sigma}, \bullet^{\sigma}), f(a, \bullet^{\eta}), \bullet^{\phi}, \ldots\}$$
$$[\![f(x^{\sigma}, b)]\!]_{\mathcal{F}} = \{b, a, f(\bullet^{\sigma}, \bullet^{\xi}), f(a,a), \ldots\}$$

Proposition 1 *$[\![t]\!]_{\mathcal{T}}$ and $[\![t]\!]_{\mathcal{U}}$ are disjoint, $[\![x^{\sigma}]\!] = \sigma^{\mathcal{T}_{\Sigma^{\bullet}}}$, and $[\![t_1 \sqcup t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$*

To each constraint C, we associate a three-valued **truth value** — *true*, *false* or *uncertain* — under a $(\mathcal{V}(C), \Sigma^{\bullet})$-assignment θ, denoted $[\![C]\!]_{\theta}$. The important cases are defined as follows. The other cases follow standard three-valued logic with $\wedge$ being the minimum of its arguments and $\vee$ the maximum (*false* $<$ *uncertain* $<$ *true*).

$$[\![t \diamond l]\!]_{\theta} = \begin{cases} \textit{true} & \text{if } \theta(t) \in [\![l]\!]_{\mathcal{F}} \\ \textit{false} & \text{if } \theta(t) \in [\![l]\!]_{\mathcal{T}} \\ \textit{uncertain} & \text{if } \theta(t) \in [\![l]\!]_{\mathcal{U}} \end{cases} \qquad [\![t : \sigma]\!]_{\theta} = \begin{cases} \textit{true} & \text{if } \theta(t) \in \sigma^{\mathcal{T}_{\Sigma^{\bullet}}} \\ \textit{false} & \text{if } \theta(t) \text{ is of sort } \eta \\ & \text{ and } \eta \sqcap \sigma = \perp \\ \textit{uncertain} & \text{otherwise} \end{cases}$$

Example 3.2 With the subsort order of Example 3.1:

$$[\![x^{\delta} : \sigma]\!]_{[x^{\delta} \longleftarrow \bullet^{\delta}]} = \textit{uncertain} \qquad [\![x^{\delta} : \delta]\!]_{[x^{\delta} \longleftarrow \bullet^{\delta}]} = \textit{true}$$
$$[\![f(b,b) \diamond a]\!]_{\theta} = \textit{true} \qquad [\![x^{\delta} \diamond f(b,b)]\!]_{[x^{\delta} \longleftarrow \bullet^{\delta}]} = \textit{uncertain}$$

To each constrained term $t|C$, we associate three disjoint sets of $\Sigma^{\bullet}$-terms. The first one, denoted by $[\![t|C]\!]_{\mathcal{T}}$ or simply by $[\![t|C]\!]$ and called the **denotation** or **solution** of $t|C$, is the set of $\Sigma^{\bullet}$-terms that are instances of t and satisfy C. The second one, denoted $[\![t|C]\!]_{\mathcal{U}}$ and called the **uncertain** or **strict** set, is the set of $\Sigma^{\bullet}$-terms for which we cannot decide if they are instances of t or if they satisfy C. The third one, denoted $[\![t|C]\!]_{\mathcal{F}}$, is the set of $\Sigma^{\bullet}$-terms that are not instances of t or do not satisfy C. Formally,

- $[\![t|C]\!]_{\mathcal{T}} = \{\theta(t) \mid \theta$ is a $(\mathcal{V}(t), \Sigma^{\bullet})$-assignment and $[\![C|_{\mathcal{V}(t)}]\!]_{\theta} = \mathit{true}\}$
- $[\![t|C]\!]_{\mathcal{U}} = \{\theta(t) \mid \theta$ is a $(\mathcal{V}(t), \Sigma^{\bullet})$-assignment and $[\![C|_{\mathcal{V}(t)}]\!]_{\theta} = \mathit{uncertain}\} \cup [\![t]\!]_{\mathcal{U}}$
- $[\![t|C]\!]_{\mathcal{F}} = \mathcal{T}_{\Sigma^{\bullet}} - [\![t|C]\!]_{\mathcal{U}} - [\![t|C]\!]_{\mathcal{T}}$

A constrained term is **consistent** if it denotes a nonempty set. If $C_1 \vee \ldots \vee C_n$ is the disjunctive normal form of C, the denotation of $t|C$ is the union of the denotations of $t \mid C_1 \ldots t \mid C_n$.

Example 3.3 Let $T = f(x^{\sigma}, y^{\sigma}) \mid x^{\sigma} : \eta \;\wedge\; y^{\sigma} \diamond q(a, z^{\rho})$. With the subsort order of Example 3.1:

$$[\![T]\!]_{\mathcal{T}} = \{f(b,b), f(\bullet^{\eta}, b), \ldots\}$$
$$[\![T]\!]_{\mathcal{U}} = \{f(\bullet^{\beta}, b), f(b, \bullet^{\rho}), \ldots\}$$
$$[\![T]\!]_{\mathcal{F}} = \{f(a, q(a, \bullet^{\rho})), b, a, \ldots\}$$

Proposition 2 *The following equivalences will be used where required:*

- $[\![x^{\eta}]\!] = [\![x^{\sigma}|x^{\sigma} : \eta]\!]$ *if* $\sigma \geq \eta$
- $[\![t]\!]_{\mathcal{F}} = [\![x^{\top}|x^{\top} \diamond t]\!]_{\mathcal{T}}$
- $[\![t \mid C_1 \vee C_2]\!] = [\![t \mid C_1]\!] \cup [\![t \mid C_2]\!]$ *and* $[\![t \mid C_1 \wedge C_2]\!] = [\![t \mid C_1]\!] \cap [\![t \mid C_2]\!]$
- $[\![t \mid \mathcal{F}]\!] = \{\}$, $[\![t \mid \mathcal{T}]\!] = [\![t]\!]$ *and, if t is a ground constructor term, then* $[\![t]\!] = \{t\}$

To each constrained pattern $P_1 \ldots P_n$ (and thus in particular to each pattern), we associate three sets of $\Sigma^{\bullet}$-terms. The **solution** or **denotation** of $P_1 \ldots P_n$, denoted by $[\![P_1 \ldots P_n]\!]_{\mathcal{T}}$ or simply by $[\![P_1 \ldots P_n]\!]$, is the set of $\Sigma^{\bullet}$-terms t for which there exists a P_i such that t matches P_i and it is decidable that t does not match $P_k, k < i$. The **uncertain** or **strict** set of a constrained pattern $P_1 \ldots P_n$, denoted $[\![P_1 \ldots P_n]\!]_{\mathcal{U}}$, is the set of $\Sigma^{\bullet}$-terms t such that there exists a P_i for which we cannot decide whether t matches P_i and t is not in the denotation of any preceding prefix $P_1 \ldots P_k$, $k < i$ of the pattern. The last one, denoted $[\![P_1 \ldots P_n]\!]_{\mathcal{F}}$, is the set of $\Sigma^{\bullet}$-terms that, decidably, are not solutions of $P_1 \ldots P_n$. Formally:

- $[\![P_1 \ldots P_n]\!]_{\mathcal{T}} = \{t \mid \exists i \; (1 \leq i \leq n) \; t \in [\![P_i]\!]_{\mathcal{T}}$ and $\forall k \; (k < i) \; t \in [\![P_k]\!]_{\mathcal{F}}\}$
- $[\![P_1 \ldots P_n]\!]_{\mathcal{U}} = \{t \mid \exists i \; (1 \leq i \leq n) \; t \in [\![P_i]\!]_{\mathcal{U}}$ and $\forall k \; (k < i) \; t \notin [\![P_1 \ldots P_k]\!]_{\mathcal{T}}\}$
- $[\![P_1 \ldots P_n]\!]_{\mathcal{F}} = \mathcal{T}_{\Sigma^{\bullet}} - [\![P_1 \ldots P_n]\!]_{\mathcal{T}} - [\![P_1 \ldots P_n]\!]_{\mathcal{U}}$

Example 3.4 Consider the constrained pattern:

$$P_1, P_2 = f(x^{\beta}, b) \mid x^{\beta} : \xi \;,\; f(y^{\sigma}, z^{\sigma}) \mid z^{\sigma} : \beta \;\wedge\; z^{\sigma} \diamond a$$

With the subsort order of Example 3.1:
$f(\bullet^{\sigma}, b) \in [\![P_2]\!]_{\mathcal{T}}$ and $f(\bullet^{\sigma}, b) \notin [\![P_1]\!]_{\mathcal{F}}$, since $f(\bullet^{\sigma}, b) \in [\![P_1]\!]_{\mathcal{U}}$. Hence, $f(\bullet^{\sigma}, b) \notin [\![P_1, P_2]\!]$

4 Compilation Rules

In this section, we describe our compilation method. This consists of three kinds of rules acting on constrained terms. The **simplification** rules transform restrictions on terms into restrictions on variables. **Partitioning** transforms an ambiguous order-sorted pattern into an equivalent set (modulo simplification) of disjoint order-sorted constrained terms. The **normalization** rules transform a set of disjoint order-sorted constrained terms into a set of simpler ones that facilitates the construction of the pattern discrimination tree.

4.1 Simplification Rules

The simplification rules define a reduction relation $\longrightarrow_s$ on constraints that transforms a structure or sort constraint on terms into an equivalent constraint on variables, that is, either $\mathcal{T}$, $\mathcal{F}$, or of the form $x \diamond t$ or $x : \sigma$. Figure 1 presents the simplification rules. Most are derived from [10, 3, 11] and are self-explanatory. The interesting ones are rules 14 and 15.

The **complete sort** rule allows us to simplify several structural constraints to a single sort constraint. It states that a term does not match any of the constructors of a sort δ if and only if it is not of sort δ. Note that this rule is only applicable when the constructors of δ are finite. The **negative sort** rule states that any term t, which is of sort σ but not of sort η, is of sort $\sigma - \eta$, where $\sigma - \eta = \sigma \sqcap \eta^c$.[3] We will write as $\sigma - \{\eta_1, \ldots, \eta_k\}$ the sort $(\ldots(\sigma - \eta_1) - \ldots) - \eta_k$.

If $x^{\sigma} : \eta$ appears in a constraint C, then the variable x^{σ} is said to be **restricted** by η in C, otherwise it is restricted by σ. We shall say that a constraint C is in **simplified form** (irreducible by $\longrightarrow_s$), denoted $C\downarrow_s$, if and only if it is either $\mathcal{T}$ or $\mathcal{F}$ or

- If $x^{\sigma} : \eta$ is in C then $\sigma > \eta$ and $\eta \neq \bot$
- If $x^{\sigma} \diamond t^{\eta}$ is in C then t is not a variable and $\sigma \geq \eta$
- If $x \diamond \{f_1, \ldots, f_n\} \in C$ and $f_i : \vec{\rho_i} \to \delta_i \in \Sigma$ $(i = 1 \ldots n)$, then $f_1, \ldots, f_n$ are not all the constructors of $\{\delta_1 \ldots \delta_n\}$

Example 4.1 The constraint $x^{\sigma} : \eta \wedge y^{\delta} \diamond f(a, a)$ is in simplified form while $x^{\sigma} \diamond z^{\xi}$ and $y^{\delta} \diamond \{p, q\}$ are not.

Theorem 1 (Simplification) *Let $t|C$ be a constrained term.*

[3]Note that $(\sigma - \eta) \leq \sigma$ and $(\sigma - \eta) \sqcap \eta = \bot$.

Structures

1. $f(t_1 \ldots t_n)^\sigma \diamond f(h_1 \ldots h_n)^\sigma$
   $\longrightarrow_s \bigvee_{1 \le i \le n} t_i \diamond h_i$
2. $f(t_1 \ldots t_n) \diamond g(h_1 \ldots h_m) \longrightarrow_s \mathcal{T}$
3. $f() \diamond f() \longrightarrow_s \mathcal{F}$
4. $t^\sigma \diamond h^\eta \longrightarrow_s \mathcal{T}$
   if $\eta \sqcap \sigma = \bot$

Conjunction and Disjunction

5. $t \diamond l \wedge t \diamond l' \longrightarrow_s t \diamond l$
   if $l \sqsubseteq l'$
6. $t \diamond l \vee t \diamond f(\vec{h}) \longrightarrow_s t \diamond l \sqcup f(\vec{h})$
   if $l \sqcup f(\vec{h})$ exists
7. $x : \sigma \wedge x \diamond t^\eta \longrightarrow_s x : \sigma$
   if $\sigma \sqcap \eta = \bot$
8. $x : \sigma \wedge x : \eta \longrightarrow_s x : \sigma \sqcap \eta$
9. $x : \sigma \vee x : \eta \longrightarrow_s x : \eta \sqcup \sigma$
   if σ and η are comparable

Positive Sorts

10. $t^\sigma : \bot \longrightarrow_s \mathcal{F}$
11. $t^\sigma : \top \longrightarrow_s \mathcal{T}$
12. $t^\sigma : \eta \longrightarrow_s \mathcal{T}$
    if $\sigma \le \eta$
13. $t^\sigma : \eta \longrightarrow_s t^\sigma : \sigma \sqcap \eta$
    if σ and η are not comparable

Negative Sorts

14. $t^\sigma \diamond y^\eta \longrightarrow_s t^\sigma : \sigma - \eta$

Complete Sort

15. $\bigwedge_{i=1}^{n} x^\sigma \diamond f_i \longrightarrow_s x^\sigma : \sigma - \{\delta_i\}_{i=1}^{n}$
    if $f_i : \vec{\rho} \to \delta_i \in \Sigma$ and the f_i's
    are all the constructors of $\{\delta_i\}_{i=1}^{n}$

Figure 1: Simplification Rules

(Invariance)	*If $C \longrightarrow_s C'$ then $[\![t\|C]\!] = [\![t\|C']\!]$*
(Termination)	*There are no infinite chains $C_1 \longrightarrow_s C_2 \longrightarrow_s \ldots$*
(Completeness)	*For each constrained term $t\|C$ where C is not in simplified form, there exists a C' such that $C \longrightarrow_s C'$*

Proof: (Sketch) The invariance claim must be verified for each rule—a tedious task, since there are so many cases, but straightforward. Completeness can be easily shown by considering a constrained term whose constraint is not in simplified form. To prove the termination claim, we use the lexical ordering over (C_1, C_2, C_3, C_4), where:

- $C_1 = \sum_{(t \diamond h) \in C} size(t)$, where $size(t)$ is defined as one would expect,
- C_2 is the number of structure atoms in C,
- C_3 is the number of sort atoms in C and,
- $C_4 = \sum_{(t:\sigma) \in C} path(\sigma)$, where $path(\sigma)$ is the length of the maximal path from $\perp$ to σ in the sort lattice.

Rules $1, 2, 3, 4$ decrease C_1; rules $5, 6, 7, 14, 15$ decrease C_2; rules $8, 9, 10, 11, 12$ decrease C_3; and rule 13 decreases C_4. When one rule decreases C_i, $C_k (k < i)$ does not change. Thus, the complexity with respect to the lexical ordering is always reduced and the length of a $\longrightarrow_s$ derivation is bounded. ▮

4.2 Partitioning

The definition of a pattern's denotation (Section 3.2) suggests splitting a pattern into an equivalent set of constrained terms, whose set denotations are disjoint, and whose union is the set denotation of the pattern.

Let $T_1 = t_1|C_1$ and $T_2 = t_2|C_2$ be two constrained terms. We say T_2 **matches** T_1, denoted $T_1 \sqsubseteq T_2$, iff there exists a substitution θ such that $t_1 \sqsubseteq t_2$ (*i.e.* $t_2 = \theta(t_1)$) and $C_2 \Rightarrow \theta(C_1)$, where $\Rightarrow$ is logical implication. A substitution θ unifies two constrained terms T_1 and T_2 if and only if θ unifies t_1 and t_2 and the constrained term $\theta(t_1) \mid \theta(C_1 \wedge C_2)$ is consistent. If θ unifies T_1 and T_2, then $T_1 \sqcup T_2 = t_1 \sqcup t_2 | \theta(C_1 \wedge C_2)$ is the least upper bound with respect to $\sqsubseteq$ and we say that T_1 and T_2 are compatible constrained terms.

Let $T = t|C$ be a constrained term. The **restriction** of T under a substitution θ, denoted $T|_\theta$, is defined to be $t|C'$ where $C \wedge t \diamond \theta(t) \longrightarrow_s^* C'$.

Proposition 3 *$[\![T_1 \sqcup T_2]\!] = [\![t_1|C_1]\!] \cap [\![t_2|C_2]\!]$ and $[\![\theta(T)]\!] \cap [\![T|_\theta]\!] = \emptyset$*

The recursive function, $\mathcal{PART}$, takes a constrained term T and a pattern $p_1 \ldots p_n$ as arguments, and partitions T according to $p_1 \ldots p_n$ into a set of constrained terms whose denotations are disjoint.[4] To illustrate, suppose we wish to partition $x^\top|\mathcal{T}$ according to the pattern $p_1 \ldots p_n$. The first set generated is $\{p_1|\mathcal{T}\}$ and we go on to recursively partition the decidable $x^\top$-complement of p_1, that is, $x^\top|x^\top \diamond p_1$, according to the rest of the pattern $p_2 \ldots p_n$. Note that the order of the pattern is respected.

[4]In fact, we mean "partitions the set denotation of T" but we shall say simply "partitions T".

$$\mathcal{PART}(T, []) = \emptyset;$$

$$\mathcal{PART}(T, p_1 \ldots p_n) = \begin{cases} \mathcal{PART}(T, p_2 \ldots p_n) \\ \text{if } T \text{ and } p_1|\mathcal{T} \text{ are not unifiable;} \\ \{T \sqcup (p_1|\mathcal{T})\} \cup \mathcal{PART}(T|_\theta, p_2 \ldots p_n), \\ \text{otherwise; where } \theta \text{ is the mgu of } T \text{ and } p_1|\mathcal{T}. \end{cases}$$

Example 4.2 With the subsort order of Example 3.1, let g be a constructor whose domain sort is $(\sigma \sqcup \delta) \times nat$, where $\sigma \sqcup \delta$ and nat are disjoint sorts. Partitioning $x^\top|\mathcal{T}$ according to the pattern $g(x^\sigma, 0), g(y^\delta, z^{nat}), x^\top$ yields:

$$\{g(x^\sigma, 0),\ g(y^\delta, z^{nat})|y^\delta : \phi,\ g(y^\delta, z^{nat})|z^{nat}\diamond 0,\ x^\top|x^\top \diamond g(x^\sigma, 0) \wedge x^\top \diamond g(y^\delta, z^{nat})\}$$

Proposition 4 *If $\mathcal{PART}(x^\top|\mathcal{T}, p_1 \ldots p_n) = \{P_1, \ldots, P_n\}$, then $P_i = \{p_i| \bigwedge_{j<i} p_i \diamond p_j\}$ and $[\![P_i]\!] \cap [\![P_j]\!] = \emptyset$ for $i \neq j$.*

Proof: (Sketch) By induction on i, using the fact that if the substitution θ_i is defined by $\theta_i(x^\top) = p_i$ for all $1 \leq i \leq n$, then $(\ldots(x^\top|\mathcal{T})|_{\theta_1} \ldots)|_{\theta_n} = x^\top|x^\top \diamond p_1 \wedge \ldots \wedge x^\top \diamond p_n$. ∎

Theorem 2 (Partitioning) *Let $p_1 \ldots p_n$ be a pattern such that $\mathcal{PART}(x^\top|\mathcal{T}, p_1 \ldots p_n) = \{P_1, \ldots, P_n\}$, then*

- $[\![p_1 \ldots p_n]\!]_\mathcal{T} \subseteq \bigcup_{1\leq i\leq n} [\![P_i]\!]_\mathcal{T}$
- *If $t \in \bigcup_{1\leq i\leq n} [\![P_i]\!]_\mathcal{T}$, then $\exists j$ such that $t \in [\![p_j]\!]_\mathcal{T}$ and $t \in [\![p_k]\!]_\mathcal{F}$, for $k < j$*
- $[\![p_1 \ldots p_n]\!]_\mathcal{F} = [\![P_1 \ldots P_n]\!]_\mathcal{F}$

Proof: (Sketch) The first claim is shown by induction on n. The second one is shown by Proposition 4 and the second item of Proposition 2. To show the last claim, it suffices to prove that both $[\![p_1 \ldots p_n]\!]_\mathcal{F}$ and $[\![P_1 \ldots P_n]\!]_\mathcal{F}$ are equal to $\bigcap_{1\leq i\leq n} [\![p_i]\!]_\mathcal{F}$. ∎

Thus, partitioning transforms an ambiguous list of order-sorted terms (a pattern) into an unambiguous set of order-sorted constrained terms. It changes neither the decidable sets associated to each pattern nor the strict one. We say that $\{S_1, \ldots, S_n\}$ is a **complete decomposition** if and only if there exists a pattern $p_1 \ldots p_n$ such that partitioning $x^\top|\mathcal{T}$ according to the list $[p_1 \ldots p_n\ x^\top]$ yields $\{S_1, \ldots, S_n\}$. When $x^\top$ is appended to the list of patterns, both the success and the failure of the matching are considered. This does not change the original problem because the discrimination tree covers all the cases that may appear during the pattern matching process. It turns out that when $\{S_1, \ldots, S_n\}$ is a complete decomposition and $t \in [\![S_j]\!]_\mathcal{F}$, there exists $1 \leq i \leq n$ such that $t \in [\![S_i]\!]_\mathcal{T}$.

4.3 Normalization

In order to simplify the construction of the PMT, we use four normalization rules (**flattening**, **sort**, **structure**, **empty**) that operate over sets of disjoint constrained terms. Since the formalization of these rules is quite complex, while the underlying idea is rather intuitive, we first show three typical examples where such normalizations are to be performed.

The first rule decomposes a complex structural constraint into several simpler ones with only one constructor in the right hand side of each structure atom. Normalizing $x^{\top}|x^{\top} \diamond cons(1,z)$ with the **flattening** rule yields $x^{\top}|x^{\top} \diamond cons(\_,\_)$ and $cons(y,z)|y \diamond 1$.

Now, consider the constrained term $x^{\top}|x^{\top}\diamond cons(y,z)$. Recall that, with a lazy evaluation mechanism the sort of a well-sorted expression can be known before its structure. Thus, if $x^{\top}$ is substituted by a term whose sort is incompatible with that of $cons(y,z)$, it is not necessary to reduce the term. On the other hand, if $x^{\top}$ is substituted by a *list* expression, it is necessary to continue reduction in order to decide whether it is a "*cons*" term. Thus, assuming that *list* and *int* are the only sorts in the system, the **sort** rule transforms $x^{\top}|x^{\top}\diamond cons(y,z)$ into $x^{\top}|x^{\top} : int$, $x^{\top}|x^{\top} : list \wedge x^{\top}\diamond cons(y,z)$.

If a variable has a common occurrence in more than one constrained term and is restricted by different structural constraints, then the **structure** rule can be applied. For example, $f(a,x^{int}) \mid x^{int} \diamond 1$ and $f(b,x^{int}) \mid x^{int} \diamond 2$, is transformed into $f(a,2)|\mathcal{T}$, $f(b,1)|\mathcal{T}$, $f(a,x^{int}) \mid x^{int} \diamond 1 \ \wedge \ x^{int} \diamond 2$ and $f(b,x^{int}) \mid x^{int} \diamond 1 \ \wedge \ x^{int} \diamond 2$. Now, the subterm at position 2 is a variable restricted by the same set of symbols or it is just one of such symbols.

We shall say that $t|C$ is in **normalized form**, if and only if C is in simplified form but is different from $\mathcal{F}$ and whenever $x\diamond f(t_1 \ldots t_n)^{\sigma}$ appears in C and $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma \in \Sigma$, then x is restricted by σ in C and $t_1 \ldots t_n$ are mutually distinct variables of sort $\sigma_1 \ldots \sigma_n$ respectively. We shall say that a set of constrained terms $\{t_1|C_1, \ldots, t_n|C_n\}$ is in normalized form (irreducible by $\longrightarrow_n$), if and only if every $t_i|C_i$ is in normalized form and, whenever there exist two terms t_i and t_j and there exists a position u such that for all $v <_{pos} u$, t_i and t_j have the same structure symbol at position v, t_i/u and t_j/u are variables restricted by the same sort and by nonempty sets s_i and s_j of structure symbols in C_i and C_j respectively, then $s_i = s_j$.

Example 4.3 The constrained term $h(x^{\sigma},y^{\phi})|y^{\phi}\diamond f(v^{\sigma},w^{\sigma})$ is in normalized form while $f(x^{\sigma},y^{\sigma})|x^{\sigma}\diamond a$ and $h(x^{\sigma},y^{\phi})|y^{\phi}\diamond f(v^{\eta},w^{\xi})$ are not.

Figure 2 presents the normalization rules. There, we assume $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma \in \Sigma$ and $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ are pairwise distinct variables. When θ is a substitution, $(t|C)\langle\langle\theta\rangle\rangle$ denotes the term $\theta(t) \mid \theta(C) \downarrow_s$. When normalizing, rules are applied in the order that they appear in Figure 2. They satisfy the properties of Theorem 1; namely, termination, invariance and completeness.

Theorem 3 (Normalization) *Let S be a set of constrained terms.*

Flattening: If $\exists i\ \big(1 \leq i \leq n\big)$ such that t_i is not a variable of sort σ_i, then

$$S \cup \{\ t \mid x\Diamond f\big(t_1 \dots t_n\big)^\sigma \wedge C\ \} \longrightarrow_n$$

$$S \cup \{\ t \mid x\Diamond f \wedge C,\ \big(t \mid x\Diamond f\big(t_1 \dots t_n\big)^\sigma \wedge C\big)\langle\!\langle x \leftarrow f\big(x_1 \dots x_n\big)\rangle\!\rangle\ \}$$

Sort: If x is restricted by η in C and $\eta > \sigma$, then

$$S \cup \{\ t \mid x\Diamond f^\sigma \wedge C\ \} \longrightarrow_n S \cup \{\ t \mid \big(x : \eta - \sigma \wedge C\big)\downarrow_s,\ t \mid \big(x : \sigma \wedge x\Diamond f^\sigma \wedge C\big)\downarrow_s\ \}$$

Structure: If $\exists u$ such that $t_1/u = \{x^\sigma | x^\sigma \Diamond s_1\}$ and $t_2/u = \{y^\sigma | y^\sigma \Diamond s_2\}, f^\sigma \in s_1$, but $f^\sigma \notin s_2$ and $\forall v, v <_{pos} u$, t_1 and t_2 have the same constructor symbol at position v, then

$$S \cup \{\ t_1 \mid P_1,\ t_2 \mid P_2\ \} \longrightarrow_n$$

$$S \cup \{\ t_1 \mid P_1,\ t_2 \mid \big(P_2 \wedge y^\sigma \Diamond f^\sigma\big)\downarrow_s,\ \big(t_2 \mid P_2\big)\langle\!\langle y^\sigma \longleftarrow f\big(x_1 \dots x_n\big)\rangle\!\rangle\ \}$$

Empty: $S \cup \{t | \mathcal{F}\} \longrightarrow_n S$

Figure 2: Normalization Rules

(Invariance)	*If $S \longrightarrow_n S'$, then $[\![S]\!] = [\![S']\!]$*
(Termination)	*There are no infinite chains $S \longrightarrow_n S' \longrightarrow_n \ldots$*
(Completeness)	*If S is not in normalized form, $\exists S'$ such that $S \left(\longrightarrow_s \cup \longrightarrow_n\right) S'$*

Proof: (Sketch) The invariance claim is straightforward. Now, suppose S is not in normalized form. If there exists a constrained term $t|\mathcal{F}$, the empty rule can be applied. If there is $t|C$ such that $x\Diamond f^{\sigma}$ is in C and x is restricted by η with $\eta > \sigma$, the sort rule can be applied. If there exists $x\Diamond f(t_1 \ldots t_n)^{\sigma}$, where $f : \sigma_1 \ldots \sigma_n \to \sigma \in \Sigma$ and t_i not a variable of sort σ_i, then the flattening rule can be applied. Suppose there exist two terms t_i and t_j and there exists a position u such that $t_i/u = \{x^{\sigma}|x^{\sigma}\Diamond s_i\}$, $t_j/u = \{y^{\sigma}|y^{\sigma}\Diamond s_j\}$, for all $v <_{pos} u$, t_i and t_j have the same structure symbol at position v, but $s_i \neq s_j$. If $f^{\eta} \in s_i$ or s_j with $\sigma > \eta$, then the sort rule can be applied. Otherwise, the structure rule can be applied. To prove the termination claim over a set $\{t_1|C_1, \ldots, t_n|C_n\}$ of constrained terms, note that rules are applied in order. Flattening decreases the complexity of right-hand sides of structure atoms of C_i's; sort decreases $\sum_{i=1\ldots n} \sum_{(x\Diamond f^{\sigma})\in C_i} distance(x\Diamond f^{\sigma}, C_i)$, where $distance(x\Diamond f^{\sigma}, C_i)$ is defined as 0 if x is restricted by σ in C_i, 1 otherwise; structure decreases $|s_i - s_j| + |s_j - s_i|$, and empty decreases the number of constrained terms. ▮

5 Sequentiality and Optimality

Compiling pattern matching consists of transforming a function defined by order-sorted patterns into a case-expression presented as a discrimination tree. The tree obtained is not always optimal, that is, it could fail to terminate on some terms that are not in the strict set of the pattern. As the evaluation mechanism is sequential, we must choose an order of verification running the risk of losing some solutions. In a many-sorted framework, consider Berry's example [1] formed by the patterns $f\left(true, true, z\right)$, $f\left(false, y, true\right)$, $f\left(x, false, false\right)$. Given a term $f\left(\_, \_, \_\right)$, we must choose an argument position in order to start the matching. If we start at position three, the term $f\left(true, true, \bullet^{Bool}\right)$ will not be matched, even though it belongs to the denotation of the first pattern. The same happens with the terms $f\left(false, \bullet^{Bool}, true\right)$, $f\left(\bullet^{Bool}, false, false\right)$ if we start at the second or third positions, respectively.

With a strict evaluation mechanism, an optimal PMT will be faster but the solutions (that is, those terms that match or not) will remain the same as that of a non-optimal tree. On the other hand, in a lazy evaluation framework, some non-optimal trees may fail to terminate due to unnecessary verifications that try to reduce subterms that do not have a head-normal form.

In our framework, the sort of each term is examined before its structure, because the sort can be refined after usually only a few reduction steps whereas to examine the structure, more reduction steps are required to obtain head-normal form. The construction method for PMT's that we present here chooses a direction (intuitively, a position in a term at which to start reduction) and thus decides whether a subsort or structure verification is required. At each level of the tree, the structures and sorts are more precise than those of preceding levels.

We propose a notion of **sequentiality** of the pattern matching predicate that takes not only the structure of terms into account, but also the sort system. Intuitively, a disjoint set S of patterns is sequential in the sense of [7, 10] if it is possible to decide the matching property without doing some look-ahead. That is, for any constrained term T not matching any pattern of S, there exists a position (so-called **direction**) where a reduction must be performed in order to decide the matching property. Furthermore, this position can be determined without looking at the subterms of T which are not computed yet. Our definition of sequentiality requires the set S of patterns to have also the **sort property**: intuitively, S has the sort property if whenever u is a position of a constrained term T to be evaluated and two different patterns of S that are compatible with T have variables x^σ, y^ρ at position u respectively, then σ and ρ are either disjoint or comparable sorts. In fact, if σ and ρ have a common subsort δ different from $\perp, \sigma$ and ρ, the position u cannot be taken as a direction because unnecessary reductions of T may be performed in order to distinguish between σ and ρ.

For example, consider the three unambiguous patterns $f(true, y^\sigma, z)$, $f(false, y^\delta, true)$, $f(x, y^\phi, false)$, where the subsort order is that of Example 3.1. If the PMT associated with this problem needs to know whether a term is of sort δ (resp. of sort σ) as in the case of $f(false, \bullet^\delta, true)$ (resp. $f(true, \bullet^\sigma, true)$), it will fail to terminate even though the term is in the denotation of the second (resp. first) pattern.

Optimal PMT's will only fail to terminate on the strict set of the problem. It turns out that sequentiality of a pattern matching problem is equivalent to optimality of its tree. Thus, sequentiality becomes a necessary and sufficient condition for the construction of an optimal tree. We shall next give an effective decision procedure for sequentiality on disjoint sets of patterns.

In reading the following section, familiarity with the work of [7] would be helpful, but the treatment is self-contained enough to be meaningful on its own.

5.1 Sequentiality

The set of **positions** or **occurrences** of a constrained term $t|C$, denoted $\mathcal{O}(t|C)$, is defined as the set of positions of t, which is recursively defined as usual as finite sequences of positive integers such that $\epsilon \in \mathcal{O}(t)$ and $k.u \in \mathcal{O}(f(t_1 \dots t_n))$ if $u \in t_k$. We use $<_{pos}$ to denote the lexical ordering between positions. The subterm of t at position u, denoted t/u, is defined as $t/\epsilon = t$ and $f(t_1 \dots t_n)/k.u = t_k/u$. We use $(t|C)/u$ to denote the constrained term $(t/u)|D$, where D is the constraint of all the atoms in C restricting variables of t/u. For example, $(f(g(x^\sigma, a), y^\rho)|x^\sigma : \eta \wedge x^\sigma \Diamond f \wedge y^\rho \Diamond g)/1 = g(x^\sigma, a)|x^\sigma : \eta \wedge x^\sigma \Diamond f$. If the replacement of the subterm of t at position u by a term p is a well-sorted term, we define $t[u \leftarrow p]$ to be that term. If $T = t|C$ and $P = p|D$ are two constrained terms and $t[u \leftarrow p]$ is a well-sorted term, $T[u \leftarrow P]$ is defined as $t[u \leftarrow p]|(C \wedge D|_{\mathcal{V}(t[u\leftarrow p])})$. For example, if $g(x^\sigma, a)$ is a term of sort ξ, $(f(g(x^\sigma, a), b)|x^\sigma : \eta \wedge x^\sigma \Diamond f)[1 \leftarrow y^\xi | y^\xi \Diamond g] = f(y^\xi, b)|y^\xi \Diamond g$.

We can think of a set of disjoint patterns S as a predicate on constrained terms such that $S(M)$ is true if and only if $\exists P \in S, P \sqsubseteq M$. If truth values are considered to be ordered by $false \sqsubseteq true$, S is a monotonic predicate on constrained terms. Increasing information about the term (in the sense of $\sqsubseteq$) can only change the value of the predicate to a favorable one.

A position $u \in \mathcal{O}(t|C)$ is said to be a **direction** of $T = t|C$ in a set of disjoint constrained patterns $S = \{t_1|C_1, \ldots, t_n|C_n\}$ if and only if

- T/u has the form $x^\sigma|P$
- For every constrained term M such that $T \sqsubseteq M$ and $S(M)$ is true, $M/u \not\sqsubseteq T/u$
- **(Sort property)** If $\exists i, j$ such that $\forall v, v <_{pos} u$, t_i and t_j have the same constructor symbol at position v; t_i/u is a variable restricted by η_i in C_i and t_j/u is a variable restricted by η_j in C_j, then $\eta_i \sqcap \eta_j = \perp$ or $\eta_i \leq \eta_j$ or $\eta_j \leq \eta_i$.

Lemma 1 *Let T be a constrained term $t|C$. A position u is a direction of T in $S = \{S_1, \ldots, S_n\}$ if and only if T/u has the form $x^\sigma|P$ and for every constrained pattern $S_i \in S$ compatible with T, we have that u is an occurrence of S_i, $S_i/u \not\sqsubseteq T/u$ and the sort property holds.*

Proof: Let u be a direction of T in $\{S_1, \ldots, S_n\}$ and S_i be a constrained pattern compatible with T. Then T/u has the form $x^\sigma|P$ with x^σ restricted by ϕ in P. Suppose that $u \notin \mathcal{O}(S_i)$. Then, there exists a position v such that $u = v.w$, $w \neq \epsilon$ and S_i/v is $y^\rho|D$. Since S_i is compatible with T, there exists a constrained term $M = m|F$ such that $S_i \sqsubseteq M$, $T \sqsubseteq M$; that is, M/v is an instance of $y^\rho|D$. Now, $(M/v)[w \leftarrow z^\phi|P] = M[u \leftarrow z^\phi|P]$ is a well-sorted constrained term that is also an instance of S_i and obviously of T. Therefore $M/u \sqsubseteq T/u$ by construction, which contradicts the hypothesis.

Conversely, let M be a constrained term such that $T \sqsubseteq M$ and $S_i \sqsubseteq M$ and suppose $M/u \sqsubseteq T/u$. Then, $S_i/u \sqsubseteq M/u \sqsubseteq T/u$, which contradicts our hypothesis. ▮

By normalization, a complete decomposition S reduces to another complete decomposition S' and the set of directions of any term T in S is the set of directions of T in S'.

We say that a constrained term T is **compatible** with a set of disjoint constrained patterns S if and only if there exists M such that $T \sqsubseteq M$ and $S(M)$ is true. In particular, if S has only an element $\{P\}$, T is compatible with S if and only if T and P are unifiable, *i.e.*, $\exists M$ such that $T \sqsubseteq M$ and $P \sqsubseteq M$.

A set of disjoint constrained patterns S is **sequential in a constrained term** T if and only if, whenever $S(T)$ is false but it is compatible with S, then there exists a direction of T in S. We say that S is **sequential** if and only if it is sequential in all constrained terms in normalized form. Sequentiality of a predicate S is the possibility of systematically expanding any constrained term step by step until either the predicate is true or it is clear that a positive answer is impossible.

The sort property enriches the known notions of sequentiality by taking into account the sort system. When a variable's position is restricted by two sorts with nonempty and nontrivial intersection, some solutions are lost, as illustrated by the following example.

Example 5.1 With the subsort order of Example 3.1, the following set of disjoint constrained patterns is not sequential:
$\{\ h(p, y^\top, z^\rho)|y^\top : \sigma,\ h(x^\rho, y^\top, p)|x^\rho \diamond p \wedge y^\top : \delta,\ h(x^\rho, y^\top, q)|y^\top : \phi\ \}$

If pattern matching starts at the first position (resp. at the third), it will fail to terminate on the term $h(\bullet^{\rho}, f(a,a), q)$ (resp. $h(p, b, \bullet^{\rho})$) even though it belongs to the denotation of the third (resp. first) pattern. Now, note that the first and second constrained terms have variables at position 2 which are restricted by sorts with nonempty and nontrivial intersection. If matching starts at this position asking whether or not a term is of sort σ (resp. of sort δ), it will fail to terminate on $h(q, \bullet^{\delta}, p)$ (resp. $h(p, \bullet^{\sigma}, p)$), even though it is in the denotation of the second (resp. first) pattern.

5.2 Construction of pattern matching trees

A **pattern matching tree (PMT)** for a constrained term T and a complete decomposition $S = \{S_1, \ldots, S_n\}$ is defined as:

- T is the root and each node is a constrained linear pattern in simplified form.
- If u is the direction of P in S and $T_1, \ldots, T_k$ are the children of P, then:
  - $T_1 \ldots T_k$ are pairwise incompatible constrained terms;
  - P/u has the form $x^{\sigma}|\mathcal{T}$, T_i and P only differ on u and $P/u \sqsubset T_i/u$;
  - for every T_i, there exists a pattern S_j such that $T_i \sqsubseteq S_j$.
- If $H_1 \ldots H_m$ are the leaves of the tree, then $\{S_1, \ldots, S_n\} \longrightarrow_n^* \{H_1, \ldots, H_m\}$

A PMT of a complete decomposition $\{S_1, \ldots, S_n\}$ is a pattern matching tree for the constrained term $x^{\top}|\mathcal{T}$ and $\{S_1, \ldots, S_n\}$. A PMT of $\{S_1, \ldots, S_n\}$ is **optimal** if and only if it fails to terminate only for the strict set of $S_1 \ldots S_n$.

We now describe an algorithm $\mathcal{TREE}$ that constructs a PMT for a constrained term $T = t|C$ and a complete decomposition $\{S_1, \ldots, S_n\}$. If C is non-consistent, return the empty tree. Otherwise, if T is an S_i, return the single-node tree T. Otherwise, normalize $\{S_1, \ldots, S_n\}$ into $\{H_1, \ldots, H_m\}$ and search a direction u of T in $\{H_1, \ldots, H_m\}$. If such a direction cannot be found, $\{H_1, \ldots, H_m\}$ is not sequential in T so fail. Otherwise, proceed with $\mathcal{DIR}(T, \{H_1, \ldots, H_m\}, u)$ where $T = t|C$; $T/u = x^{\sigma}|\mathcal{T}$; $H_i = h_i|C_i$ and $\mathcal{DIR}$ is defined by:

```
$\mathcal{DIR}\ (T, \{H_1, \ldots, H_m\}, u) =$
    Let $Sorts$ be the maximal sorts of $\{\ \eta \mid T \sqsubseteq H_i,\ h_i/u \in \mathcal{V}$ is restricted by $\eta\ \}$ and
    let $Forms$ be $\{\ f_i \mid T \sqsubseteq H_i, h_i/u = f_i(\ldots)\ \}$ in
    if $Sorts = \emptyset$
    then (a structure step) build a tree rooted at $T$ with children :
        $\mathcal{TREE}(T[u \leftarrow f(\ldots)] \downarrow_s, \{H_1, \ldots, H_m\})$ for each $f$ in $Forms$ and
        $\mathcal{TREE}(t \mid (C \wedge x^{\sigma} \diamond Forms) \downarrow_s, \{H_1, \ldots, H_m\}))$
    else (a sort step) build a tree rooted at $T$ with children:
        $\mathcal{TREE}(t \mid (C \wedge x^{\sigma} : \eta) \downarrow_s, \{H_1, \ldots, H_m\})$ for each $\eta \in Sorts$ and
        $\mathcal{TREE}(t \mid (C \wedge x^{\sigma} : \sigma - Sorts) \downarrow_s, \{H_1, \ldots, H_m\}))$
```

Example 5.2 Let us consider the subsort order of Example 3.1. Let α be the sort $\sigma \sqcup \delta$ and g be the constructor of ϕ whose domain sort is $\alpha \times nat$, with nat and α disjoint sorts. Consider the pattern $g\big(x^{\sigma}, y^{nat}\big),\ g\big(x^{\alpha}, 0\big),\ g\big(x^{\delta}, 1\big)$. Normalization yields:

$g\big(x^{\sigma}, y^{nat}\big)$
$g\big(x^{\alpha}, 0\big)|x^{\alpha} : \phi$
$g\big(x^{\delta}, 1\big)|x^{\delta} : \phi$
$y^{\top}|y^{\top} : nat \sqcup \sigma$
$g\big(x^{\alpha}, y^{nat}\big)|x^{\alpha} : \phi \wedge y^{nat} \diamond 0 \wedge y^{nat} \diamond 1$

Figure 3 shows the PMT. The directions are between square brackets.

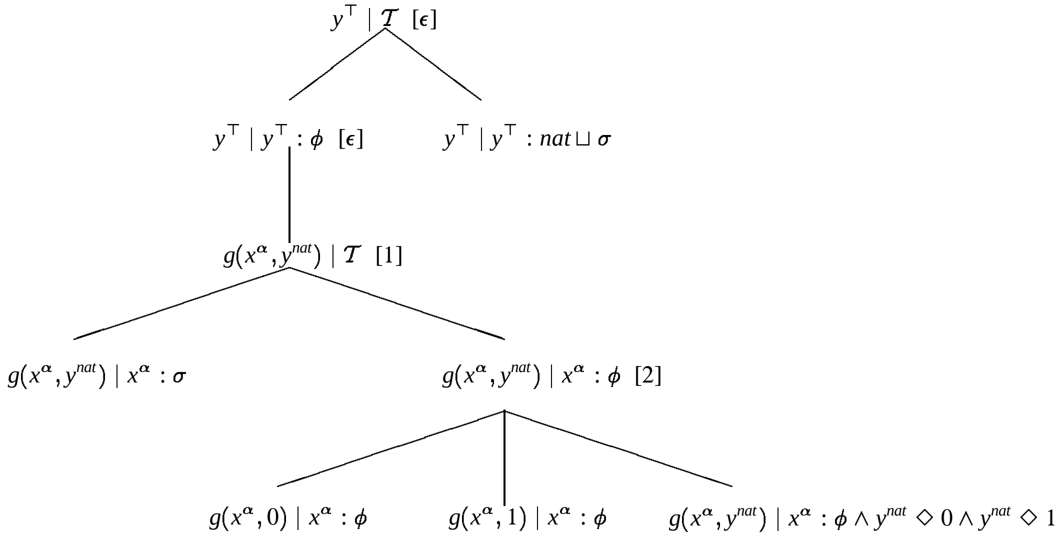

$y^{\top} \mid \mathcal{T}$ [ϵ]
$y^{\top} \mid y^{\top} : \phi$ [ϵ]
$y^{\top} \mid y^{\top} : nat \sqcup \sigma$
$g(x^{\alpha}, y^{nat}) \mid \mathcal{T}$ [1]
$g(x^{\alpha}, y^{nat}) \mid x^{\alpha} : \sigma$
$g(x^{\alpha}, y^{nat}) \mid x^{\alpha} : \phi$ [2]
$g(x^{\alpha}, 0) \mid x^{\alpha} : \phi$
$g(x^{\alpha}, 1) \mid x^{\alpha} : \phi$
$g(x^{\alpha}, y^{nat}) \mid x^{\alpha} : \phi \wedge y^{nat} \diamond 0 \wedge y^{nat} \diamond 1$

Figure 3: Pattern Matching Tree

Theorem 4 *A finite complete decomposition in normalized form $S = \{S_1, \ldots, S_n\}$ is sequential in every normalized term if and only if it is sequential in every node of its associated PMT.*

Proof: Since nodes of the PMT are in normalized form the left to right implication is evident. Conversely, let M be a constrained term in normalized form such that $S(M)$ is false and M is compatible with S. As $x^{\top}|\mathcal{T}$ is the root of the pattern matching tree but M does not match any pattern of S, there exists a node $T = t|C$ whose children are $T_1 \ldots T_m$ and whose direction in S is u such that $T \sqsubseteq M$ and for all $1 \leq i \leq m$, $T_i \not\sqsubseteq M$. We will prove that u is a direction of M in S, that is (by Lemma 1) the sort property holds, (1) M/u has the form $x^{\rho}|P$ and for all $S_i \in S$ compatible with M, (2) u is an occurrence of S_i and (3) $S_i/u \not\sqsubseteq M/u$.

By definition of discrimination tree T/u is $x^\sigma|\mathcal{T}$ and so (1) holds. By construction each level is either a sort or a structure step and since u is a direction of T in S the sort property holds.

In the sort step case, $T_1/u \dots T_m/u$ are variables $x_1^{\rho_1} \dots x_m^{\rho_m}$ and (1) immediately holds. Since $T \sqsubseteq M$ and nodes at the same level of T are defined to be incompatible, if S_i is a pattern compatible with M, then $T \sqsubseteq S_i$ and (2) holds. By construction, for every S_j compatible with T we have $S_j/u \not\sqsubseteq T/u$. In particular we have $S_i/u \not\sqsubseteq T/u$. Now, if $T/u = M/u$ (3) holds. Otherwise, suppose $S_i/u \sqsubseteq M/u$. As M is in normalized form and σ is not a minimal sort, M/u does not have any structure atom and then M/u is of the form $x^\rho|\mathcal{T}$ and $\sigma > \rho$. Since $S_i/u \sqsubseteq M/u$, S_i/u is $x^{\xi_i}|\mathcal{T}$ and $\xi_i \geq \rho$. By hypothesis for all $1 \leq j \leq m$, $T_j/u \not\sqsubseteq M/u$ and then $\rho_j \not\geq \rho$. As $T \sqsubseteq S_i$, there exists T_j such that $T_j \sqsubseteq S_i$ and then $\rho_j = \xi_i$. Then $\rho_j \geq \rho$ which contradicts the hypothesis.

In the structure step case $T_1/u = f_1, \dots, T_{m-1}/u = f_{m-1}$ and $T_m/u = x^\sigma|x^\sigma\Diamond\{f_1 \dots f_{m-1}\}$. Since $T \sqsubseteq M$ and for all $1 \leq i \leq m$, $T_i \not\sqsubseteq M$, $T/u = M/u$ or $M/u = x^\sigma|x^\sigma\Diamond S$. In the first case, we have u is a direction of M in S. In the second case (1) immediately holds. Let S_i be a constrained pattern compatible with M. As in the sort step case, $T \sqsubseteq S_i$ and (2) holds too. Now, suppose $S_i/u \sqsubseteq M/u$. Then S_i/u is a variable and there must be a child T_j of T such that $T_j \sqsubseteq S_i$. We have T_j/u is also a variable and thus necessarily $j = m$ and $\{f_1 \dots f_{m-1}\} \subseteq S$. By construction, T and T_j only differ on u and $T \sqsubseteq M$. Therefore $T_j \sqsubseteq M$ which contradicts the hypothesis and thus (3) holds. ∎

Theorem 5 *A PMT of a complete decomposition S in normalized form is optimal iff S is sequential.*

Proof: By Theorem 4, $\{S_1, \dots, S_n\}$ is sequential if and only if there exists a discrimination tree in which each node $t|C$ has a direction in the set $\{S_1, \dots, S_n\}$ and the sort property holds. The set of terms for which the algorithm does not terminate is generated at each node $t|C$ of the PMT by some terms of the form $(t|C)[u \leftarrow \bullet^\rho]$, where u is the chosen direction of $t|C$ in $\{S_1, \dots, S_n\}$. By definition $\{S_1, \dots, S_n\}$ is optimal if and only if it fails to terminate only for the strict set of $\{S_1, \dots, S_n\}$. We must verify that the algorithm fails to terminate in $(t|C)[u \leftarrow \bullet^\rho]$ if and only if it is in the strict set of $\{S_1, \dots, S_n\}$. The right to left implication is evident. Conversely, by construction each level is either a sort or a structure step. If $T_1, \dots, T_m$ are the children of the node $t|C$, two cases are to be considered:

In the structure level case, $T_i = (t|C)[u \leftarrow f_i^\eta]$, $i = 1 \dots m-1$ and $T_m = t \mid (C \wedge x^\sigma\Diamond\{f_1^\eta, \dots, f_{m-1}^\eta\})$. By normalization T/u is a variable restricted by η and then $\rho = \eta$. Thus, $(t|C)[u \leftarrow \bullet^\eta]$ is in the strict set of each T_i and then in the strict set of each leaf S_j such that $T_i \sqsubseteq S_j$. Then $(t|C)[u \leftarrow \bullet^\eta]$ is in the strict set of $\{S_1, \dots, S_n\}$.

In the sort level case, T/u has the form $x^\sigma|\mathcal{T}$ and by construction each T_i is of the form $t \mid (C \wedge x^\sigma : \eta_i)$ with $\eta_i < \sigma$. Since the decomposition is complete, $\bigsqcup_{i=1\dots m} \eta_i = \sigma$ and by the sort property $\eta_1 \dots \eta_m$ are pairwise disjoint sorts. If the algorithm fails to terminate in $(t|C)[u \leftarrow \bullet^\rho]$, there exists at least one i such that $\rho \not\leq \eta_i$ and $\rho \sqcap \eta_i \neq \perp$. We have $(t|C)[u \leftarrow \bullet^\rho]$ in the strict set of T_i. Now, for all S_j such that $T_i \sqsubseteq S_j$, we have either $T_i/u = S_j/u$ or $S_j/u = f^{\eta_i}$ or $S_j/u = x^{\eta_i}|x^{\eta_i}\Diamond\{f^{\eta_i}, g^{\eta_i}, \dots\}$ and then $(t|C)[u \leftarrow \bullet^\rho]$ is in the strict set of S_j. Then $(t|C)[u \leftarrow \bullet^\rho]$ is in the strict set of $\{S_1, \dots, S_n\}$. ∎

6 Discussion

Unitary signatures have been defined in Section 2 to be regular and to verify some constraints over the set of sort symbols and the set of function and constructor declarations. Since complements of sorts are often used during the compilation scheme we require the lattice of sorts to be boolean. On the other hand, regularity is a sufficient condition for signatures to be finitary unifying. Nevertheless, it is not very clear why we restrict our interest to unitarity ones. We present two simple transformation rules acting on signatures. A signature obtained through these rules is constructed to verify the minimal codomain sort and the disjoint domain sort properties. The transformation preserves the set of well-sorted ground constructor terms (*i.e.* the free order-sorted term algebra) and so, we can think of our compilation scheme as one that not only transforms patterns, but also signatures. We show in what follows how to obtain this "compiled" version.

First, let us justify the need for constructors to verify the hypothesis of minimal codomain sort (the third condition of unitarity). Let $\Sigma = \langle \mathcal{S}, \leq, \mathcal{F}, \mathcal{C}, \mathcal{V}, \mathcal{D} \rangle$ be a regular signature which does not satisfy this condition and $(S, \leq)$ be a boolean lattice. Then, there exists a constructor $f \in \mathcal{C}$ such that $f : \vec{\eta} \rightarrow \sigma \in \mathcal{D}$ and σ is not a minimal sort. The **downward** signature Σ' obtained from Σ is

$$\langle \mathcal{S} \cup \{\lambda\} \cup \{\lambda^c\}, \leq \cup \{\sigma > \lambda\} \cup \{\lambda^c > \eta \mid \eta \in \Gamma\}, \mathcal{F}, \mathcal{C}, \mathcal{V}, (\mathcal{D} - \{f : \vec{\eta} \rightarrow \sigma\}) \cup \{f : \vec{\eta} \rightarrow \lambda\} \rangle$$

where λ and λ^c are new sort symbols and Γ is the set of maximal sorts of $\{\eta \mid \eta \sqcap \lambda = \perp\}$.

Note that σ is strictly greater than the new symbol λ which is now a minimal sort. Intuitively, Σ' has the same structure, but constructors are in a "lower level" of the lattice. When the number of non-minimal codomain constructors is finite, this transformation terminates and the same set of ground constructor terms can be built in Σ'. The new partial order set of sorts is also a boolean lattice.

Now, suppose the minimal codomain sort condition is verified whereas the disjoint domain sort is not. Let $\Sigma = \langle \mathcal{S}, \leq, \mathcal{F}, \mathcal{C}, \mathcal{V}, \mathcal{D} \rangle$ and f a constructor with two different declarations $f : \sigma_1 \ldots \sigma_n \rightarrow \sigma$ and $f : \eta_1 \ldots \eta_m \rightarrow \eta$. If $n = m = 0$, then $f :\rightarrow \sigma$ and $f :\rightarrow \eta$ implies $\sigma = \eta$, because σ and η are minimal and whenever Σ is regular they must be comparable. Then $n = m \geq 1$ and $\vec{\sigma}, \vec{\eta}$ are not disjoint sorts. In this case there exists a term (not necessarily a ground term) having η and σ as sorts. Since Σ is regular and η, σ are minimal we also have $\sigma = \eta$. The **disjoint domain** signature obtained from Σ is $\Sigma' = \langle \mathcal{S}, \leq, \mathcal{F}, \mathcal{C}, \mathcal{V}, \mathcal{D}' \rangle$, where the new set of declarations is defined in this way:

- If $\vec{\sigma} \leq \vec{\eta}, f : \vec{\sigma} \rightarrow \sigma$ is redundant and we can remove it. $\mathcal{D}'$ is $\mathcal{D} - \{f : \vec{\sigma} \rightarrow \sigma\}$
- Otherwise,
  - $\mathcal{I}_\sigma = \{i \in [1 \ldots n] \mid \sigma_i - \eta_i \neq \perp\}$
  - $\forall i \in \mathcal{I}_\sigma, \forall j \in [1 \ldots n], \xi^i_j$ is $\sigma_i - \eta_i$ if $i = j$, σ_i otherwise
  - $\mathcal{D}_\sigma = \{f : \xi^i_1 \ldots \xi^i_n \rightarrow \sigma \mid i \in \mathcal{I}_\sigma\}$
  - $\mathcal{I}_\eta = \{i \in [1 \ldots n] \mid \eta_i - \sigma_i \neq \perp\}$
  - $\forall i \in \mathcal{I}_\eta, \forall j \in [1 \ldots n], \phi^i_j$ is $\eta_i - \sigma_i$ if $i = j$, η_i otherwise
  - $\mathcal{D}_\eta = \{f : \phi^i_1 \ldots \phi^i_n \rightarrow \sigma \mid i \in \mathcal{I}_\eta\}$
  - $\mathcal{D}'$ is $\mathcal{D} - \{f : \vec{\eta} \rightarrow \sigma, f : \vec{\sigma} \rightarrow \sigma\} \cup \{f : \vec{\eta} \sqcap \vec{\sigma} \rightarrow \sigma\} \cup \mathcal{D}_\sigma \cup \mathcal{D}_\eta$

This transformation always terminates and the obtained declarations have incompatible domains by definition of "$-$" (set difference). Even now, we can build the same set of ground constructor symbols.

Comon [2] noticed that an order-sorted signature Σ is a finite bottom-up tree automaton where the set of final states is the set of sorts $\mathcal{S}$. It turns out that the set of well-sorted terms of sort σ is the set of trees recognized by a tree automaton at the final state corresponding to σ. The transformations of signatures we have proposed above are simply transformations of their tree automata. When restricting the set of final states of the new tree automaton Σ' to that of Σ, the same set of well-sorted terms is recognized.

Unsorted and many-sorted signatures are particular cases of unitary ones, and therefore our work also remains applicable to them. There are two other interesting order-sorted type systems to be considered. The work of Aït-Kaci and Smolka [13] has shown that **features types** and constructor types are dual concepts. In this kind of system every constructor symbol has exactly one declaration and is a constructor of a minimal sort. In addition, the set of feature terms is a prelattice, provided the sort symbols are ordered as a lattice. On the other hand, Smolka [12] proposes a discipline with polymorphic order-sorted types restricted to free constructors. Specification of the inclusion order between types is defined via special classes of terminating rewriting systems and no function symbol contains more that one declaration. He shows that the set of sort terms equipped with the order specified by the rewriting rules is a well-founded quasi-lattice having $\perp$ as its least element. Reasonable algorithms to compute the greatest common subsort and least common subsort of two sort terms are given. Our order-sorted framework also allows us to accommodate pattern matching in languages with such a type system.

7 Conclusion

The method of treating ambiguous linear order-sorted pattern matching presented in this paper generalizes previous work on non-ambiguous linear patterns [7], ambiguous linear patterns [9] and ambiguous linear patterns using constrained terms [10]. We extend several notions introduced in [10], such as constrained terms, non-reducible $\bullet$ terms, strict sets of patterns, sequentiality and pattern-matching trees, to the order-sorted case. We define discrimination trees to have not only edges labeled with structure constraints, but also with subsort restrictions. This feature allows to decide pattern matching without reducing terms to normal forms, taking advantage in this way of the lazy evaluation strategy. It turns out that our method constructs optimal order-sorted PMT's for sequential order-sorted pattern matching problems and can be used either with a lazy or strict evaluation strategy. As in [10], our method can also be used for non-sequential problems.

Our general order-sorted framework accommodates lazy pattern matching on all the regular systems described in Section 6. Compilation of non-linear and higher-order patterns remains as further research work.

References

1. Gérard Berry. Séquentialité de l'évaluation formelle des lambdas-expressions. Proc. 3rd International Colloquium on Programming, Paris, France (1978).

2. Hubert Comon. Inductive Proofs by Specifications Transformation. In *Rewriting techniques and applications, LNCS 355,* pages 76–91. Springer-Verlag (1989).

3. Hubert Comon. Disunification: a Survey. Technical Report 540, Laboratoire de Recherche en Informatique. Université Paris-Sud, Centre d'Orsay (1990).

4. Joseph Goguen and José Meseguer. Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. Technical report, SRI International-CSL (1989).

5. Joseph Goguen and Timothy Winkler. Introducing OBJ3. Technical Report SRI-CSL-88-9, SRI International, Computer Science Lab (1988).

6. Robert Harper, David MacQueen, and Robin Milner. Standard ML. Technical Report ECS-LFCS-86-2, Department of Computer Science, University of Edinburgh (1986).

7. Gérard Huet and Jean-Jacques Lévy. Call by need computations in non ambiguous linear term rewriting systems. Rapport INRIA Laboria 359, INRIA, Le Chesnay, France (1979).

8. Jean-Pierre Jouannaud and Claude Kirchner. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. Technical Report 561, Laboratoire de Recherche en Informatique. Université Paris-Sud, Centre d'Orsay (1990).

9. Alain Laville. Evaluation des Filtrages avec Priorité. Application au Langage ML. Thèse de Doctorat. Université Paris VII (1988).

10. 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).

11. Gert Smolka. A Feature Logic with Subsorts. Technical Report 33, WT LILOG, IBM Deutschland (1988).

12. Gert Smolka. Logic Programming over Polymorphically Order-Sorted Types. PhD thesis. Universität Kaiserslautern, Germany (1988).

13. Gert Smolka and Hassan Aït-Kaci. Inheritance Hierarchies: Semantics and Unification. Volume 7. In *Journal of Symbolic Computation*. Academic Press (1989).

14. 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).

10

Pattern Matching in Order-Sorted Languages

Delia Kesner