

Natural Narrowing for General Term Rewriting Systems

Santiago Escobar¹, José Meseguer², and Prasanna Thati³

¹ Universidad Politécnica de Valencia, Spain. sescobar@dsic.upv.es

² University of Illinois at Urbana-Champaign, USA. meseguer@cs.uiuc.edu

³ Carnegie Mellon University, USA. thati@cs.cmu.edu

Abstract. For narrowing to be an efficient evaluation mechanism, several *lazy* narrowing strategies have been proposed, although typically for the restricted case of left-linear constructor systems. These assumptions, while reasonable for functional programming applications, are too restrictive for a much broader range of applications to which narrowing can be fruitfully applied, including applications where rules have a non-equational meaning either as *transitions* in a concurrent system or as *inferences* in a logical system. In this paper, we propose an efficient lazy narrowing strategy called *natural narrowing* which can be applied to general term rewriting systems with no restrictions whatsoever. An important consequence of this generalization is the *wide range of applications* that can now be efficiently supported by narrowing. We highlight a few such applications including symbolic model checking, theorem proving, programming languages, and partial evaluation. What thus emerges is a general and efficient *unified mechanism* based on narrowing, that seamlessly integrates a very wide range of applications in programming and proving.

1 Introduction

Rewriting is currently recognized as a very general declarative formalism to specify, program, and reason about computational systems. The more traditional applications have been in the context of equational reasoning and of equational/functional programming, where the rewriting relation is understood as *oriented equality*. But there is an increasing awareness of the usefulness of rewriting in non-equational contexts, where a rewrite is understood as a *transition* or an *inference*: for example to specify and program concurrent systems or logical inference systems. This has led, for example, to theoretical developments such as rewriting logic [33], and to the development of language implementations supporting non-equational rewriting such as ELAN [8] and Maude [9].

A similar widening of the scope is needed for *narrowing*, which generalizes rewriting by performing unification in nonvariable positions instead of the usual matching. Narrowing can in this way endow rewriting languages with new programming and reasoning capabilities in a much wider setting and for a much wider range of applications than those based on equational logic. The traditional understanding of narrowing [18,26] has been as a mechanism for equational *unification*, that is, for solving equational goals $E \vdash (\exists \vec{x}) t = t'$; as

a consequence, properties such as confluence and termination have often been assumed, and many equational reasoning applications, as well as a number of functional/logic programming languages supporting narrowing, have been developed. As proposed for example in [35], in a non-equational setting narrowing can instead be understood as a powerful mechanism for solving *reachability goals* $\mathcal{R} \vdash (\exists \vec{x}) t \rightarrow^* t'$ in a rewrite theory \mathcal{R} . The traditional equational interpretation can still be kept as a special case since, as explained in Section 4.2, solving equations becomes a special case of solving more general reachability goals. However, in this wider setting traditional assumptions such as confluence and termination are in general no longer reasonable (see Section 1.1 and [35,41] for a discussion of completeness issues for narrowing in this setting).

A key challenge for narrowing is the danger of *combinatorial explosion* in exploring the narrowing tree. It becomes in fact essential in practice to use adequate *narrowing strategies* that are as greedy as possible, yet remain complete. One important breakthrough in this direction was the realization that one could extend the work on optimal lazy *reduction* strategies originating with Huet and Levy [25], and extended in different ways by other researchers (see, e.g., [38,3,5,14]) to obtain efficient lazy *narrowing* strategies that only instantiate those positions that are really needed. This was first achieved by Antoy, Echahed and Hanus, who extended their (weakly) outermost-needed rewriting strategy to a (*weakly*) *needed narrowing* strategy [6,5]. Recently, both (weakly) outermost-needed rewriting and (weakly) outermost-needed narrowing have been further improved by Escobar by means of the *natural rewriting* and *natural narrowing* strategies [14,15]. We postpone a more detailed discussion of related work in this area until Section 1.1. For the moment, the main point to bear in mind is that most of the work on lazy rewriting and lazy narrowing strategies [20,30,22,38,3,5,6,4,14,15] has taken place within the context of functional (logic) programming languages, and depends on assumptions such as having *left-linear* and *constructor-based* rules. These assumptions are reasonable for some functional (logic) programming languages, but they substantially limit the expressive power of equational languages such as OBJ [21], CafeOBJ [19], ASF+SDF [13], and the equational subset of Maude [9], where non-linear left-hand sides which need not be constructor based are perfectly acceptable. Such assumptions become even more restrictive and unreasonable for non-equational rewriting languages such as ELAN [8] and Maude [9], where a rewrite $t \rightarrow t'$ is no longer understood as a step of equational simplification but as a transition, and where the rules need not be confluent nor terminating, need not be left-linear, and the constructor-based assumption is utterly unreasonable and almost never holds.

The goal of this paper is to propose an efficient lazy narrowing strategy called *generalized natural narrowing* that greatly extends the natural narrowing strategy of [14] and keeps all the good properties while overcoming all the above limitations. In fact our strategy can be applied to *completely general* rewrite systems with no restrictions whatsoever: even rewrite rules with extra variables in their righthand sides are allowed; furthermore, we allow rewritings to be *context-sensitive* [31] according to a function ϕ specifying which argument positions in

each function symbol are *frozen*, so that rewriting in the subterms at those positions is forbidden. In this way, we obtain a general lazy narrowing strategy applicable in the broader setting of solving reachability goals for rewrite systems whose rules can have a non-equational semantics. Furthermore, this generalization is obtained together with *actual gains in efficiency*, in the sense that our natural narrowing strategy, besides being efficiently implementable, performs strictly better than previously proposed lazy strategies when specialized to their setting (see Section 1.1). A further efficiency advantage, generalizing that of [6,5,14,15], is that, as explained in Section 3, natural narrowing computes substitutions in an *incremental* way, without explicit use of a unification algorithm. Perhaps the most important consequence of the generality of our natural narrowing strategy is the *wide range of applications* that can now be supported. To give the reader a better feeling for some of these application areas, including symbolic model checking, programming, theorem proving, and partial evaluation, we discuss those areas, and the respective benefits of using natural narrowing for each of them, in Section 4. What emerges, in summary, is a general and efficient *unified mechanism*, seamlessly integrating rewriting and narrowing, and making it available for a very wide range of programming and proving applications.

To give the reader a first intuitive feeling for how our generalized natural narrowing works, and for the difficulties that it resolves, we illustrate some of the key issues by means of a simple example.

Example 1. Consider the following rewrite system for proving equality of arithmetic expressions built using division (\div), modulus or remainder (%), subtraction ($-$), and minimum (`min`) operations on natural numbers.

- | | |
|--|--|
| (1) $M \% s(N) \rightarrow (M-s(N)) \% s(N)$ | (5) $\text{min}(0, N) \rightarrow 0$ |
| (2) $(0 - s(M)) \% s(N) \rightarrow N - M$ | (6) $\text{min}(s(N), 0) \rightarrow 0$ |
| (3) $M - 0 \rightarrow M$ | (7) $\text{min}(s(N), s(M)) \rightarrow s(\text{min}(M, N))$ |
| (4) $s(M) - s(N) \rightarrow M - N$ | (8) $X \approx X \rightarrow \text{True}$ |

Note that this rewrite system is not left-linear because of rule (8) and it is not constructor-based because of rule (2). Furthermore, note that it is neither terminating nor confluent due to rule (1).

The aim of natural rewriting and narrowing strategies [14] is to lazily compute head-normal forms of a given term t . Specifically, given a term that is not a head-normal form, the strategy reduces (narrows) to the extent possible, only those redexes (narroxes) that are necessary for a rule to be applied at the root. We would like to generalize natural rewriting and narrowing to a version that enjoys the good optimality properties of natural rewriting and natural narrowing (see [14]) and that can also handle non-left-linear and non-constructor-based rules such as (2) and (8). This is accomplished for rewriting in the *generalized natural rewriting strategy* of [17]. In this paper, we are interested in a narrowing strategy which, besides performing the minimal number of evaluations, instantiates variables only as much as necessary. For example, consider the term⁴

⁴ The subterm `10!` represents factorial of 10 but we do not include the rules for `!` because we are only interested in the fact that it has a remarkable computational cost, and therefore we would like to avoid its reduction whenever possible.

$t_2 = 10! \% \min(X, X-0) \approx 10! \% 0$ and the following two narrowing sequences we are interested in amongst all possible. First, the following sequence leading to **True**, that starts by unifying subterm $t_2|_{1.2}$ with left-hand side (lhs) l_5 :

$$10! \% \min(X, X-0) \approx 10! \% 0 \rightsquigarrow_{[X \mapsto 0]} 10! \% 0 \approx 10! \% 0 \rightsquigarrow_{id} \text{True}$$

Second, the following sequence not leading to **True**, that starts by reducing subterm $t_2|_{1.2.2}$ with lhs l_3 and that early instantiates variable X :

$$\begin{aligned} 10! \% \min(X, X-0) &\approx 10! \% 0 \\ \rightsquigarrow_{[X \mapsto s(X')]} 10! \% \min(s(X'), s(X')) &\approx 10! \% 0 \\ \rightsquigarrow_{id} 10! \% s(\min(X', X')) &\approx 10! \% 0 \end{aligned}$$

Note that although it is possible to further narrow the last term, we are not interested in doing so, since such term is already a head-normal form. In the following, we informally introduce the key points of our strategy:

1. (*Demanded positions*). This notion is relative to a lhs l and determines which positions in a term t should be narrowed in order to be able to apply lhs l at root position. For the term $t_2 = 10! \% \min(X, X-0) \approx 10! \% 0$ and lhs $l_8 = X \approx X$, only subterm $\min(X, X-0)$ is demanded, since it is the only disagreeing part in $t_2|_1$ w.r.t. $t_2|_2$.
2. (*Failing term*). This notion is relative to a lhs l and stops further wasteful narrowing steps. Specifically, the last term $10! \% s(\min(X', X')) \approx 10! \% 0$ of the second former sequence fails w.r.t. l_8 , since the subterm $s(\min(X', X'))$ is demanded by l_8 but there is no possible narrowing step above it that would convert it into term 0.
3. (*Most frequently demanded positions*). This notion determines those demanded positions w.r.t. non-failing lhs's that are demanded by the maximum number of rules and that cover all such non-failing lhs's. It provides the optimality properties of our natural rewriting and narrowing strategies, since it substantially reduces the set of positions to be considered. If we look closely at lhs's l_5 , l_6 , and l_7 defining \min , we can see that position 1 in the term $\min(X, X-0)$ is more demanded than position 2, i.e., position 1 is disagreeing w.r.t. l_5 , l_6 , and l_7 , whereas position 2 is disagreeing only w.r.t. l_6 and l_7 . Thus, position 1 is the most frequently demanded position for all rules defining \min that also covers such rules. Note that position 1 is rooted by a variable and this motivates the following point.
4. (*Lazy instantiation*). This notion relates to an incremental construction of unifiers without the explicit use of an unification algorithm. This is necessary in the previous example, since subterm $\min(X, X-0)$ does not unify with lhs l_6 and l_7 . However, we can deduce that narrowing at subterm $X-0$ is only necessary when substitution $[X \mapsto s(X')]$, inferred from l_6 and l_7 , have been applied. Thus, we early construct the appropriate substitutions $[X \mapsto 0]$ and $[X \mapsto s(X')]$ in order to reduce the search space.

Following is the outline of the rest of the paper. In Section 2, we present the preliminary background. In Section 3 we define our generalized natural narrowing strategy. In Section 4, we motivate our work by illustrating various applications of the generalized narrowing strategy. We conclude in Section 5. Proofs of all

results are given in Appendix A. Now, we further motivate the paper with a comparison with the related literature.

1.1 Related work

The rewriting and narrowing strategies known as *lazy* avoid the combinatorial explosion by computing (whenever possible) only those evaluation steps that are *needed* for obtaining head-normal forms. Lazy rewriting strategies are based on the original *strongly needed reduction* strategy of Huet and Levy [25]. This strategy is optimal (computes only needed steps), correct, and complete for *strongly sequential rewrite systems* (SS), a subclass of orthogonal rewrite systems. Note that this strategy does not apply to Example 1, since orthogonality implies left-linearity. Sekar and Ramakrishnan proposed the *parallel needed reduction* strategy [38] as an extension of Huet and Levy’s strongly needed reduction to make it correct and complete for a larger rewrite system class, though still optimal for the former class. This larger rewrite system class is *constructor-based weakly orthogonal systems* (CB-WO), and thus it is not applicable to Example 1. Antoy proposed the *outermost-needed rewriting* [3] as an efficient implementation of strongly needed reduction for *inductively sequential rewrite systems* (IS), which are equivalent to SS’s when instantiated to left-linear constructor systems [24]. In [3], Antoy also provides the *weakly outermost-needed rewriting* to make outermost-needed rewriting correct and complete for CB-WO’s. Thus, both are not applicable to Example 1.

The continuous interest in the unification of functional and logic programming in a seamless way attracted much attention (see [23] for a survey) and Antoy, Echahed and Hanus extended Antoy’s outermost-needed rewriting strategy to the *needed narrowing* strategy [6], becoming the best narrowing strategy for functional logic programming languages over other narrowing strategies such as [20,30,22] (see [6] for a detailed comparison). Antoy, Echahed, and Hanus extended their needed narrowing strategy to the *weakly needed narrowing strategy* [5] in order to cope with CB-WO’s. Note that these strategies are not applicable to Example 1.

In recent work [14], we have proposed refinements of (weakly) outermost-needed rewriting and (weakly) needed narrowing, called *natural rewriting* and *natural narrowing*. These strategies compute less than (or exactly the same) steps than outermost-needed rewriting and needed narrowing for IS’s. Optimality is identified for specific terms rather than classes of rewrite systems. In the case of weakly outermost-needed rewriting and narrowing, they also compute less than (or exactly the same) steps. However, they are not applicable to Example 1 because of left-linearity and constructor-based conditions.

It is worthy to mention that an exception about left-linearity condition is [4]. In [4], a non-left-linear rule “ $l \rightarrow r$ ” is transformed into a left-linear conditional rule “ $l' \rightarrow r$ if $\dots, X \downarrow X_1, \dots, X \downarrow X_n, \dots$ ” by renaming, in the linear term l' , extra occurrences of a variable X to X_1, \dots, X_n . However, $t \downarrow s$ succeeds only if there exists a constructor term w such that $t \rightarrow^* w$ and $s \rightarrow^* w$ and this is an unreasonable condition for the kind of rewrite systems with a non-equational

semantics considered in this paper. Anyway, the strategy is not applicable to Example 1, since it imposes the constructor-based condition.

This work is part of a broader joint effort to generalize narrowing from equational logic to rewriting logic, so as to make possible a much wider range of applications. It builds on our previous work extending natural rewriting to general term rewriting systems [17], and also on work by Meseguer and Thati on narrowing for rewrite theories [35,41]. As shown in [35,41], for general rewrite theories which need not be confluent and need not be terminating, the issue of *completeness*, that is, of narrowing being a complete semi-decision procedure for solving reachability goals, is nontrivial and does not always hold, essentially because rewrites can take place in the substitutions. The paper [35] characterizes several classes of rewrite theories for which narrowing is complete; and [41] gives a generalized “back-and-forth” version of narrowing that is guaranteed to always be complete. In contrast to [35,41], this paper offers a narrowing strategy, which could be the basis of [35,41], to make narrowing efficient, and proves that this strategy is sound and complete in a different sense of “completeness,” namely that any solution found by unrestricted narrowing (outside substitutions) will also be reachable using the strategy.

2 Basic Concepts

We assume some familiarity with term rewriting and narrowing (see [40,23] for missing definitions). We assume a *signature* Σ and an infinite set of variables X . We write T_Σ for ground terms, and $T_\Sigma(X)$ for terms with variables. $\mathcal{V}ar(t)$ denotes the set of variables occurring in t . A variable $x \in \mathcal{V}ar(t)$ is called linear in t if it has only one occurrence in t . A term t is said to be linear if every variable $x \in \mathcal{V}ar(t)$ is linear in t .

Positions are defined as strings of natural numbers, with Λ the root (or empty) position, $p.q$ position concatenation, and $p < q$ the usual prefix ordering. We extend position concatenation to sets $P.Q = \{p.q \mid p \in P \wedge q \in Q\}$ and use $P.q$ and $p.Q$ as shorthands for $P.\{q\}$ and $\{p\}.Q$. We say $p \in P$ is an *outermost* position in P if there is no $q \in P$ such that $q < p$. If $p = q.q'$, we define $p/q = q'$. $root(t)$ denotes the symbol labeling the root position, $t|_p$ the subterm at position p , $t|_P$ its extension to a set of positions, and $t[s]_p$ the replacement of subterm $t|_p$ in t by s . $\mathcal{P}os_S(t)$ denotes all positions in t with a symbol from $S \subseteq \Sigma \cup X$; we use $\mathcal{P}os_f(t)$ and $\mathcal{P}os(t)$ as shorthands for $\mathcal{P}os_{\{f\}}(t)$ and $\mathcal{P}os_{\Sigma \cup X}(t)$. A *substitution* is a mapping from variables to terms, its application to a term is denoted by $\sigma(t)$, its restriction to a set of variables V is denoted by $\sigma|_V$, and substitution concatenation is denoted as $\sigma \circ \theta$, i.e., $\sigma(\theta(x))$ for a variable x . We write $\sigma^{-1}(x) = \{y \mid \sigma(y) = x\}$. We write $t \leq t'$ if $\exists \sigma : \sigma(t) = t'$, and in that case we say t *matches* t' .

A rewrite rule is written $l \rightarrow r$, where $l, r \in T_\Sigma(X)$ and $l \notin X$. A *rewrite system*⁵ is a triple $\mathcal{R} = (\Sigma, \phi, R)$ with Σ a signature, R a set of rewrite rules, and

⁵ What we call here a rewrite system is a special case of a *rewrite theory* [34], which is a 4-tuple (Σ, ϕ, E, R) with E a set of equations.

$\phi : \Sigma \rightarrow \mathcal{P}(\mathbb{N})$ specifies the *frozen* arguments of each symbol $\phi(f) \subseteq \{1, \dots, k\}$ for the arity k of f . We say position p in t is *frozen* if $\exists q < p$ such that $p = q.i.q'$ and $i \in \phi(\text{root}(t|_q))$. $L(\mathcal{R})$ denotes the set of *left-hand-sides* (lhs) of rules in R . \mathcal{R} is called *left-linear* if all left-hand-sides are linear terms. Given $\mathcal{R} = (\Sigma, \phi, R)$, we can decompose Σ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined symbols*, where $\mathcal{D} = \{\text{root}(l) \mid l \in L(R)\}$ and $\mathcal{C} = \Sigma - \mathcal{D}$. A *pattern* is a term with only one defined symbol, at root position. $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, \phi, R)$ is called a *constructor system* (CS) if all left-hand-sides are patterns.

A *rewrite step* is defined as $t \rightarrow_{\langle p, l \rightarrow r \rangle} s$ (or simply $t \rightarrow s$) if $p \in \text{Pos}_{\mathcal{D}}(t)$ is not frozen in t , $l \rightarrow r \in R$, $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$. The pair $\langle p, l \rightarrow r \rangle$ or the subterm $t|_p$ are called a *redex*. $\xrightarrow{>A}$ denotes a rewrite step at a position $p > A$. A term t is called a *normal form* if there is no rewrite step from t and a *head-normal form* (or *root-stable*) if there is no rewrite sequence starting from t with a step at root position. A substitution σ is called *normalized* if $\sigma(x)$ is a normal form for all variables x . Similarly, a *narrowing step* is defined as $t \rightsquigarrow_{\langle p, l \rightarrow r, \sigma \rangle} s$ (or simply $t \rightsquigarrow_{\sigma} s$) if $\sigma(t) \rightarrow_{\langle p, l \rightarrow r \rangle} s$. The triple $\langle p, l \rightarrow r, \sigma \rangle$ is called a *narrox*.

3 Generalizing Natural Narrowing

In [17], we have generalized the natural rewriting strategy of [14] to the larger class of rewrite systems that need not be left-linear and constructor-based. A noteworthy feature of the generalization is that it is conservative, i.e., the generalized strategy coincides with the original one for the class of left-linear constructor-based rewrite systems.

In this paper, we define, again for this larger class of systems, a generalized natural narrowing strategy that, to the extent possible, performs only those narrowing steps that are essential for applying some rule at root position. That is, if a term t is not a head-normal form (or some substitution making the term not a head-normal form exists), then we know that after a (possibly empty) sequence of narrowing steps at positions other than the root, a narrowing step at the root position with a rule $l \rightarrow r$ is possible. Besides, narrowing can be simply understood as an extension of rewriting, where in addition to rewriting at non-variable positions, one can also instantiate variables; indeed, we can see any narrowing step as a sequence of several unitary instantiations and a reduction step. According to the previous two points, we adopt the approach⁶ of computing: (i) a demanded set of narroxes for non-variable positions in t such that *at least* one of them has to be reduced before *any* rule can be applied at the root position; and (ii) a *demanded* set of variables in t such that *at least* one of them has to be instantiated before any rule can be applied at the root position.

We use the notions of demanded positions, failing terms, and most frequently demanded positions, as in [17]. First, we recall the notion of demanded positions.

⁶ This idea of demandedness for reductions and instantiations is common in lazy narrowing strategies for programming languages, such as needed narrowing [6]; see [7] for a survey about demandedness in programming languages.

Definition 1. For a term s and a set of terms $T = \{t_1, \dots, t_n\}$ we say that s is a context of the terms in T if $s \leq t_i$ for all $1 \leq i \leq n$. There is always a least general context s of T , i.e., one such that for any other context s' we have $s' \leq s$; furthermore s is unique up to renaming of variables. For $1 \leq i \leq n$, let the substitution σ_i be such that $\sigma_i(s) = t_i$ and $\text{Dom}(\sigma_i) \subseteq \text{Var}(s)$. We define the set $\text{Pos}_{\neq}(T)$ of disagreeing positions between the terms in T as those $p \in \text{Pos}_X(s)$ such that there is an i with $\sigma_i(s|_p) \neq s|_p$.

Definition 2 (Demanded positions). For terms l and t , let s be the least general context of l and t , and let σ be the substitution such that $\sigma(s) = l$. We define the set of demanded positions in t w.r.t. to l as

$$DP_l(t) = \bigcup_{x \in \text{Var}(s)} \begin{array}{l} \text{if } \sigma(x) \notin X \text{ then } \text{Pos}_x(s) \text{ else } Q.\text{Pos}_{\neq}(t|_Q) \\ \text{where } Q = \text{Pos}_{\sigma^{-1}(\sigma(x))}(s) \end{array}$$

Intuitively, the set $DP_l(t)$ returns a set of positions in t at which t necessarily has to be “changed” (either by a narrowing step if it is a non-variable position, or by an instantiation if it is a variable position) before the rule $l \rightarrow r$ can be applied at the root position, i.e., before l can match the term under consideration.

Example 2. Consider the left-hand side $l_8 = X \approx X$ and the term $t_2 = 10! \% \min(X, X-0) \approx 10! \% 0$ of Example 1. The least general context of l_8 and t_2 is $s = W \approx Z$. Now, for $\sigma = \{W \mapsto X, Z \mapsto X\}$, we have $\sigma(s) = l_8$. Then, while computing $DP_{l_8}(t_2)$, we compute the set of disagreeing positions between the subterms in t_2 corresponding to the non-linear variable X in l_8 , i.e., the set $\text{Pos}_{\neq}(T)$ for $T = t_2|_{\{1,2\}} = \{10! \% \min(X, X-0), 10! \% 0\}$. According to Definition 1, the least general context of T is the term $10! \% Y$ and the set of disagreeing positions between terms in T is then $\text{Pos}_{\neq}(T) = \{2\}$. Thus, we obtain that $DP_{l_8}(t_2) = \{1, 2\}.\{2\} = \{1.2, 2.2\}$.

Now, four points have to be addressed. First, note that the symbol at a position $p \in DP_l(t)$ in t can be changed not only by a narrowing step or by instantiation at p , but also by a narrowing step or instantiation at a position $q < p$. Thus, besides considering the positions in $DP_l(t)$ as candidates for evaluation, we also need to consider the positions q in t that are above some position in $DP_l(t)$. However, we only need to consider those positions q at which t has a defined symbol that is not frozen, because otherwise an evaluation at that position in t is not possible. Thus, for a position q in a term t , we define $D_t^\uparrow(q) = \{p \mid p \leq q \wedge p \in \text{Pos}_{\mathcal{D}}(t) \wedge p \text{ is not frozen}\}$. We lift this to sets of positions as $D_t^\uparrow(Q) = \bigcup_{q \in Q} D_t^\uparrow(q)$. This gives us the following useful result that shows how demandedness captures the neededness of positions in rewrite sequences for the application of a rule at root position.

Lemma 1. [17] Consider a rewrite sequence $t \rightarrow_{\langle p_1, l_1 \rightarrow r_1 \rangle} t_1 \cdots \rightarrow_{\langle p_n, l_n \rightarrow r_n \rangle} t_n$ such that $p_n = \Lambda$. Then, either $l_n \leq t$ or there is a k , $1 \leq k < n$, such that $p_k \in D_t^\uparrow(DP_{l_n}(t))$.

Example 3. Continuing Example 2, we have $D_{t_2}^\uparrow(DP_{l_8}(t_2)) = D_{t_2}^\uparrow(\{1.2, 2.2\}) = \{\Lambda, 1, 1.2, 2\}$, since position 2.2 is rooted by a constructor symbol, and thus removed.

Second, we only compute $DP_l(t)$ for those left-hand sides l such that t does not *fail* w.r.t. l . Roughly, we say t fails w.r.t. l if no sequence of narrowing steps or instantiations in t will help to produce a term to which the rule $l \rightarrow r$ can be applied at the root. Thus, we can safely ignore the positions demanded by such l . Note that this behavior is proved to be complete in Theorem 3 below.

For a position p and a term t , we define the set $R_t(p)$ of *reflections* of p w.r.t. t as follows: if p is under a variable position in t , i.e., $p = q.q'$ for some q such that $t|_q = x$, then $R_t(p) = \mathcal{P}os_x(t).q'$, else $R_t(p) = \{p\}$. We say that the path to p in t is *stable* (or simply p is stable) if $root(t|_p) \notin X$ and $D_t^\uparrow(p) \setminus \{\Lambda\} = \emptyset$.

Definition 3 (Failing term). *Given terms l, t , we say t fails w.r.t. l , denoted by $l \blacktriangleleft t$, if there is $p \in DP_l(t)$ such that p is stable, and one of the following holds: (i) $R_l(p) \cap DP_l(t) = \{p\}$; or (ii) there is $q \in R_l(p) \cap DP_l(t)$ with $root(t|_p) \neq root(t|_q)$, and q is also stable. We denote by $l \blacktriangleleft t$ that t is not failing w.r.t. l .*

Example 4. Consider the subterm $t_2|_2 = 10! \% 0$ and the lhs's $l_1 = \mathbf{M} \% \mathbf{s}(\mathbf{N})$ and $l_2 = (\mathbf{O} - \mathbf{s}(\mathbf{M})) \% \mathbf{s}(\mathbf{N})$ in Example 1. We have that $l_1 \blacktriangleleft t_2|_2$ and $l_2 \blacktriangleleft t_2|_2$, because position 2 in $t_2|_2$ belongs to $DP_{l_1}(t_2|_2)$ and $DP_{l_2}(t_2|_2)$, is stable, and $R_{l_1}(2) = R_{l_2}(2) = \{2\}$. Similarly, the term $t' = 10! \% \mathbf{s}(\min(\mathbf{X}', \mathbf{X}')) \approx 10! \% 0$ fails w.r.t. l_8 since $1.2, 2.2 \in DP_{l_8}(t')$, $1.2, 2.2$ are stable, and $R_{l_8}(1.2) = R_{l_8}(2.2) = \{1.2, 2.2\}$.

Third, we do not consider all positions in each $DP_l(t)$ but a subset called the *most frequently demanded positions*. The idea behind this is that narrowing or instantiating at those positions is enough for being able to reduce at root position. In this way, we can substantially reduce (or optimize) the set of positions to be considered. Specifically, for each $l \in L(\mathcal{R})$ such that $l \blacktriangleleft t$, instead of considering (the defined symbols above) every position in $DP_l(t)$, it is sufficient to consider only the positions $R_l(q)$ for at least one $q \in DP_l(t)$. Note that this behavior is proved to be complete in Theorem 3 below.

Definition 4 (Most frequently demanded positions). *Given the sets of positions $\{Q_1, \dots, Q_n\} = \bigcup_{l \in L(\mathcal{R}) \wedge l \blacktriangleleft t} \{DP_l(t)\}$, we define the filtered set of demanded positions of a term t , denoted by $FP(t)$, as one of the minimal sets of positions P such that (i) for each $1 \leq j \leq n$, $Q_j \cap P \neq \emptyset$ and (ii) for each $p \in P$ and $l \in L(\mathcal{R})$ such that $p \in DP_l(t)$, $R_l(p) \subseteq P$.*

Example 5. Consider the subterm $t_2|_1 = 10! \% \min(\mathbf{X}, \mathbf{X}-0)$ and the lhs's l_1 and l_2 in Example 1. The reader can check that $DP_{l_1}(t_2|_1) = \{2\}$, $DP_{l_2}(t_2|_1) = \{1, 2\}$, and $DP_l(t_2|_1) = \{\Lambda\}$ for any other lhs l . Then, we have $\bigcup_{l \in L(\mathcal{R}) \wedge l \blacktriangleleft t} \{DP_l(t)\} = \{\{2\}, \{1, 2\}, \{\Lambda\}\}$ and $FP(t_2|_1) = \{\Lambda, 2\}$, since this set covers the three previous sets obtained from the rules. On the other hand, consider the term t_2 and the lhs l_8 in Example 1. From Example 2, we have $DP_{l_8}(t_2) = \{1.2, 2.2\}$ and $DP_l(t_2) = \{\Lambda\}$ for any other lhs l . Thus, $FP(t_2) = \{\Lambda, 1.2, 2.2\}$, since this set is closed by reflection.

Fourth, whenever t matches l for a rule $l \rightarrow r$, in addition to reducing t with $l \rightarrow r$ we have to consider as candidates for evaluation those positions q in t that have a defined symbol and that are above a variable position in l . This is necessary for completeness of the strategy, see [17]. Now, we are able to define the set of *sufficient demanded positions* that collects the four previous ideas.

Definition 5 (Sufficient demanded positions). *We define the sufficient set of demanded positions of a term t as*

$$SP(t) = \cup_{l \in L(\mathcal{R}) \wedge l \leq t} D_l^\dagger(\mathcal{P}os_X(l)) \cup D_t^\dagger(FP(t)).$$

Example 6. Consider term t_2 in Example 1. Since t_2 is not a redex, we have $SP(t_2) = D_{t_2}^\dagger(FP(t_2))$. By Example 5, we have $SP(t_2) = D_{t_2}^\dagger(\{A, 1.2, 2.2\})$. And by Example 3, we have $SP(t_2) = \{A, 1, 1.2, 2\}$.

At this point, we can recall⁷ the *generalized natural rewriting* strategy [17].

Definition 6 (Generalized Natural Rewriting). *We define the set of demanded redexes of a term t as*

$$DR(t) = \{\langle A, l \rightarrow r \rangle \mid l \in L(\mathcal{R}) \wedge l \leq t\} \cup \cup_{q \in SP(t) \setminus \{A\}} q.DR(t|_q)$$

where for a set of redexes S we define $q.S = \{\langle q.p, l \rightarrow r \rangle \mid \langle p, l \rightarrow r \rangle \in S\}$. We say term t reduces by natural rewriting to term s , denoted by $t \xrightarrow{m}_{\langle p, l \rightarrow r \rangle} s$ (or simply $t \xrightarrow{m} s$) if $t \rightarrow_{\langle p, l \rightarrow r \rangle} s$, $\langle p, l \rightarrow r \rangle \in DR(t)$.

Theorem 1 (Correctness & Completeness). [17] *If a term t is a \xrightarrow{m} -normal form, then t is root-stable. If $t \rightarrow^* s$, then there is an s' such that $t \xrightarrow{m^*} s'$, $root(s') = root(s)$ and $s' \xrightarrow{\geq A} s$.*

Before defining generalized natural narrowing, we have to define a set of demanded substitutions for narrowing, to address the question of what substitutions the demanded variable positions are to be instantiated with.

Definition 7 (Demanded substitutions). *We define the set $DSub(t)$ of demanded substitutions for narrowing t at the root position as follows. For each pair of $p \in FP(t) \cap \mathcal{P}os_X(t)$ and $l \in L(\mathcal{R})$ such that $p \in DP_l(t)$ and $l \blacktriangleleft t$, we construct the substitution σ as explained below, and stipulate that $\sigma \in DSub(t)$.*

If $p \in \mathcal{P}os_\Sigma(l)$, then $\sigma = [t|_p \mapsto root(l|_p)(\bar{w})]$ for fresh variables \bar{w} . On the other hand, if $p \notin \mathcal{P}os_\Sigma(l)$, then we know that p is under a non-linear variable position in l , i.e., $|R_l(p) \cap DP_l(t)| > 1$, and let $Q = R_l(p) \cap DP_l(t)$. There are two cases: (i) if all the terms in $t|_Q$ are variables, then we define σ to be such that for every $q \in Q$ we have $\sigma(t|_q) = w$ for a fresh variable w , (ii) if every non-variable term in $t|_Q$ is rooted by the same symbol f , then we define $\sigma = [t|_p \mapsto f(\bar{w})]$.

⁷ Generalized natural rewriting is formalized in [17] using a stronger *failing* notion instantiated for rewriting, where condition $root(t|_p) \notin X$ in the stable definition is removed.

Note that, in the previous definition, if there are two non-variable positions $p_1, p_2 \in Q$ such that $t|_{p_1}$ and $t|_{p_2}$ are rooted with different symbols, then no substitution can resolve the conflict between the disagreeing positions p_1 and p_2 and they are demanded for evaluation.

We can deduce the following useful result that ensures that appropriate substitutions are inferred from left-hand sides for demanded variable positions.

Lemma 2. *Let l, t, σ , and $p \in FP(t) \cap Pos_X(t)$. If $l \blacktriangleleft t$, $p \in DP_l(t)$, and $p \notin DP_l(\sigma(t))$, then there exists $\theta \in DSub(t)$ s.t. $\theta(t|_p) \neq t|_p$ and $\theta|_{Var(t)} \leq \sigma|_{Var(t)}$.*

Example 7. Consider the subterm $t_2|_{1.2} = \min(X, X-0)$ and the lhs's $l_5 = \min(0, N)$, $l_6 = \min(s(N), 0)$, and $l_7 = \min(s(N), s(M))$ of Example 1. The reader can check that $DP_{l_5}(t_2|_{1.2}) = \{1\}$, $DP_{l_6}(t_2|_{1.2}) = \{1, 2\}$, $DP_{l_7}(t_2|_{1.2}) = \{1, 2\}$, and $DP_l(t_2|_{1.2}) = \{A\}$ for any other lhs l . Thus $FP(t_2|_{1.2}) = \{A, 1\}$ according to Definition 4. Then, position 1 is a variable position and we have that the demanded substitutions for its variable X are $DSub(t_2|_{1.2}) = \{[X \mapsto 0, X \mapsto s(X')]\}$, since $1 \in Pos_\Sigma(l_5)$, $1 \in Pos_\Sigma(l_6)$, $root(l_5|_1) = 0$, and $root(l_6|_1) = s$.

Thus, substitutions are computed in a lazy and *incremental* fashion *without resorting to an explicit unification algorithm*. Now, we formally define our generalized natural narrowing strategy.

Definition 8 (Generalized Natural Narrowing). *We define the set of demanded narroxes of a term t as*

$$DN(t) = \{\langle A, l \rightarrow r, id \rangle \mid l \in L(\mathcal{R}) \wedge l \leq t\} \cup \bigcup_{q \in SP(t) \setminus \{A\}} q \cdot DN(t|_q) \cup \bigcup_{\sigma \in DSub(t)} DN(\sigma(t)) @ \sigma$$

where for a set of narroxes S we define $q.S = \{\langle q.p, l \rightarrow r, \theta \rangle \mid \langle p, l \rightarrow r, \theta \rangle \in S\}$ and $S @ \sigma = \{\langle p, l \rightarrow r, \theta \circ \sigma \rangle \mid \langle p, l \rightarrow r, \theta \rangle \in S\}$. We say that term t reduces by *natural narrowing* to term s , denoted by $t \rightsquigarrow_{\langle p, l \rightarrow r, \sigma \rangle}^m s$ (or simply $t \rightsquigarrow^m s$) if $t \rightsquigarrow_{\langle p, l \rightarrow r, \sigma \rangle} s$, $\langle p, l \rightarrow r, \sigma \rangle \in DN(t)$, and $p \in Pos_{\mathcal{D}}(t)$.

In the following, we omit the rule $l \rightarrow r$ in a narrowing step $\langle p, l \rightarrow r, \sigma \rangle$, whenever there is no scope for ambiguity about the rule.

Example 8. Consider again the term $t_2 = 10! \% \min(X, X-0) \approx 10! \% 0$ from Example 1 and the computation of $DN(t_2)$. Since t_2 is not a redex, we have that $DN(t_2) = \bigcup_{q \in SP(t_2) \setminus \{A\}} q \cdot DN(t_2|_q) \cup \bigcup_{\sigma \in DSub(t_2)} DN(\sigma(t_2)) @ \sigma$. By Example 6, $SP(t_2) = \{A, 1, 1.2, 2\}$. We also have $DSub(t_2) = \emptyset$, since no position in $FP(t_2)$ is a variable. Thus, $DN(t_2) = 1.DN(t_2|_1) \cup 2.DN(t_2|_2) \cup 1.2.DN(t_2|_{1.2})$. This implies that we recursively compute $DN(t_2|_1)$, $DN(t_2|_{1.2})$ and $DN(t_2|_2)$.

Now consider $DN(t_2|_{1.2}) = DN(\min(X, X-0))$. Since it is not a redex, we have $DN(t_2|_{1.2}) = \bigcup_{q \in SP(t_2|_{1.2}) \setminus \{A\}} q \cdot DN(t_2|_{1.2}|_q) \cup \bigcup_{\sigma \in DSub(t_2|_{1.2})} DN(\sigma(t_2|_{1.2})) @ \sigma$. By Example 7, we have $SP(t_2|_{1.2}) = D_{t_2|_{1.2}}^\uparrow(FP(t_2|_{1.2})) = D_{t_2|_{1.2}}^\uparrow(\{A, 1\}) = \{A\}$ and $DSub(t_2|_{1.2}) = \{\sigma, \sigma'\}$ for $\sigma = [X \mapsto 0]$ and $\sigma' = [X \mapsto s(X')]$. Then, $DN(t_2|_{1.2}) = DN(\sigma(t_2|_{1.2})) @ \sigma \cup DN(\sigma'(t_2|_{1.2})) @ \sigma'$, and we recursively call to

$DN(\min(0,0-0))$ and $DN(\min(\mathbf{s}(X'),\mathbf{s}(X')-0))$. Now, the reader can check that $DN(\min(0,0-0)) = \{ \langle A, id \rangle \}$, since term $\min(0,0-0)$ matches lhs l_5 , and $DN(\min(\mathbf{s}(X'),\mathbf{s}(X')-0)) = 2.DN(\mathbf{s}(X')-0) = \{ \langle 2, id \rangle \}$, since term $\mathbf{s}(X')-0$ matches lhs l_3 . Hence, we can conclude $DN(t_2|_{1.2}) = \{ \langle A, id \rangle \} @ \sigma \cup \{ \langle 2, id \rangle \} @ \sigma' = \{ \langle A, [X \mapsto 0] \rangle, \langle 2, [X \mapsto \mathbf{s}(X')] \rangle \}$.

Now consider $DN(t_2|_1) = DN(10! \% \min(X, X-0))$. Since it is not a redex, we have $DN(t_2|_1) = \cup_{q \in SP(t_2|_1) \setminus \{A\}} q.DN(t_2|_{1.q}) \cup \cup_{\sigma \in DSub(t_2|_1)} DN(\sigma(t_2|_1)) @ \sigma$ and $SP(t_2|_1) = D_{t_2|_1}^\uparrow(FP(t_2|_1))$. By Example 5, $FP(t_2|_1) = \{A, 2\}$, and then $SP(t_2|_1) = D_{t_2|_1}^\uparrow(\{A, 2\}) = \{A, 2\}$. Moreover, we have $DSub(t_2|_1) = \emptyset$, since no position in $FP(t_2|_1)$ is rooted by a variable. Thus, we have $DN(t_2|_1) = 2.DN(t_2|_{1.2})$. However, $DN(t_2|_{1.2})$ was already computed before, and we obtain $DN(t_2|_1) = \{ \langle 2, [X \mapsto 0] \rangle, \langle 2.2, [X \mapsto \mathbf{s}(X')] \rangle \}$.

Finally, consider $DN(t_2|_2) = DN(10! \% 0)$. Since it is not a redex, we have $DN(t_2|_2) = \cup_{q \in SP(t_2|_2) \setminus \{A\}} q.DN(t_2|_{2.q}) \cup \cup_{\sigma \in DSub(t_2|_2)} DN(\sigma(t_2|_2)) @ \sigma$. But by Example 4, we have $l_1 \blacktriangleleft t_2|_2, l_2 \blacktriangleleft t_2|_2$, and $DP_l(t_2|_2) = \{A\}$ for any other lhs l . Thus, we can conclude $DN(t_2|_2) = \emptyset$. Finally, putting everything together we have $DN(t_2) = \{ \langle 1.2, [X \mapsto 0] \rangle, \langle 1.2.2, [X \mapsto \mathbf{s}(X')] \rangle \}$. Note these narroxes correspond to the optimal narrowing sequences in Example 1.

Note that the strategy ignores the narroxes $\langle p, l \rightarrow r, \sigma \rangle$ at positions within the computed substitution σ , as shown in the following example.

Example 9. Consider the rewrite system with rules: $\mathbf{f}(a) \rightarrow \mathbf{b}$ and $\mathbf{a} \rightarrow \mathbf{b}$. Given $t = \mathbf{f}(X)$, we have $DN(t) = \{ \langle A, \mathbf{f}(a) \rightarrow \mathbf{b}, [X \mapsto a] \rangle, \langle 1, \mathbf{a} \rightarrow \mathbf{b}, [X \mapsto a] \rangle \}$ but only the first narrox is considered by the generalized natural narrowing.

This is sufficient for the correctness and completeness results we are interested in, which are concerned only with normalized substitutions (see Theorems 2 and 3). Of course, we can modify Definition 8 so that $DN(t)$ does not return the narroxes at positions within the computed substitutions. However, we have defined $DN(t)$ as above in the interest of a simpler exposition. We are now ready to state the correctness and completeness properties of our generalized natural narrowing strategy. The key fact used in establishing these properties is the correspondence between generalized natural rewriting and generalized natural narrowing, as stated by the following lemma.

Lemma 3 (Completeness w.r.t. rewriting). *For a normalized substitution σ , if $\sigma(t) \xrightarrow{m}_{\langle p, l \rightarrow r \rangle} s$, then there are η, θ, s' such that $t \xrightarrow{m}_{\langle p, l \rightarrow r, \theta \rangle} s'$, $\sigma|_{\mathcal{V}ar(t)} = (\eta \circ \theta)|_{\mathcal{V}ar(t)}$, $s = \eta(s')$, and η is normalized.*

Using Lemmas 2 and 3 we get the following correctness and completeness results for generalized natural narrowing.

Theorem 2 (Correctness). *If t is not a variable and is a \xrightarrow{m} -normal form, then for every normalized σ we have $\sigma(t)$ is root-stable.*

Theorem 3 (Completeness). *If $\sigma(t) \rightarrow^* s$ and σ is a normalized substitution, then there are s', θ, θ' s.t. $t \xrightarrow{m}_\theta^* s'$, $\theta'(s') \xrightarrow{\geq A}^* s$, and $\sigma|_{\mathcal{V}ar(t)} = (\theta' \circ \theta)|_{\mathcal{V}ar(t)}$.*

The following example shows that our completeness result needs not hold for non-normalized substitutions.

Example 10. Consider the rewrite system from [35] with rules: (i) $\mathbf{f}(\mathbf{b}, \mathbf{c}) \rightarrow \mathbf{d}$, (ii) $\mathbf{a} \rightarrow \mathbf{b}$, and (iii) $\mathbf{a} \rightarrow \mathbf{c}$. For the term $t = \mathbf{f}(\mathbf{X}, \mathbf{X})$ and substitution $\sigma = [\mathbf{X} \mapsto \mathbf{a}]$ we have $\sigma(t) \rightarrow^* \mathbf{d}$. But the generalized natural narrowing strategy cannot compute the normal form \mathbf{d} . Specifically, positions 1 and 2 in t are demanded by rule (i), and the variable \mathbf{X} is instantiated with substitutions $[\mathbf{X} \mapsto \mathbf{b}]$ and $[\mathbf{X} \mapsto \mathbf{c}]$, which are inferred from rule (i). However, both $\mathbf{f}(\mathbf{b}, \mathbf{b})$ and $\mathbf{f}(\mathbf{c}, \mathbf{c})$ are failing w.r.t. the left-hand side of rule (i). Thus $DN(t) = \emptyset$, and the strategy does not narrow the term $\mathbf{f}(\mathbf{X}, \mathbf{X})$ any further.

4 Application Areas

In this section, we show how narrowing can be used as a unified mechanism for programming and proving, and explain informally how the generalized natural narrowing strategy makes the specific applications more efficient. Our purpose in this section is motivational. More details and examples are given in [16].

4.1 Symbolic Model Checking

Narrowing can be used as a technique for symbolic reachability analysis of concurrent systems with infinite state space. Specifically, a concurrent system can naturally be expressed as a rewrite system $\mathcal{R} = (\Sigma, \phi, R)$, where terms represent states, and a rewrite rule $t \rightarrow t'$ is understood as a (parametric) local transition [32,34]. We can then formalize a reachability problem for a concurrent system thus specified as solving an existential formula $(\exists \vec{x}) t(\vec{x}) \rightarrow^* t'(\vec{x})$ where the *source* $t(\vec{x})$ is a term with variables representing a possibly infinite set of *initial* states (namely all its instances by *ground substitutions*) and the *target* $t'(\vec{x})$ represents a likewise possibly infinite set of *final* states that we want to reach by a sequence of transitions. *Solutions* to this reachability problem can then be described by substitutions σ for which indeed we have, $\sigma(t(\vec{x})) \rightarrow^* \sigma(t'(\vec{x}))$. More generally, we may consider conjunctive reachability goals of the form $G = (\exists \vec{x}) t_1(\vec{x}) \rightarrow^* t'_1(\vec{x}) \wedge \dots \wedge t_n(\vec{x}) \rightarrow^* t'_n(\vec{x})$.

We can reduce solving a conjunctive reachability goal such as G , to the case of solving a *single* reachability goal by means of a theory transformation associating to a rewrite system $\mathcal{R} = (\Sigma, \phi, R)$, a corresponding rewrite system $\hat{\mathcal{R}} = (\hat{\Sigma}, \hat{\phi}, \hat{R})$, where $\hat{\Sigma} = \Sigma \cup \{\wedge, \rightarrow, \mathbf{True}\}$, $\hat{\phi}$ extends ϕ with $\hat{\phi}(\rightarrow) = \{2\}$, and $\hat{R} = R \cup \{x \rightarrow x \rightarrow \mathbf{True}, \mathbf{True} \wedge \mathbf{True} \rightarrow \mathbf{True}\}$. Note that the second argument of \rightarrow is frozen, since only the sources of a goal are to be rewritten. Then, σ is a solution of the conjunctive goal G in the rewrite system R if and only if σ is a solution of the *single* reachability goal $(\exists \vec{x}) \hat{G} \rightarrow^* \mathbf{True}$ in the rewrite system \hat{R} , where \hat{G} is the $\hat{\Sigma}$ -term $\hat{G} = t_1 \rightarrow t'_1 \wedge \dots \wedge t_n \rightarrow t'_n$.

Now, since the term \mathbf{True} in the transformed theory $\hat{\mathcal{R}}$ is a head normal form, any root normalizing strategy \mathcal{S} [36], including our efficient generalized natural narrowing strategy, gives us a semi-decision procedure to find all the normalized solutions of a goal G . Specifically, the algorithm incrementally builds the

narrowing tree starting from the term \hat{G} , and searches, using \mathcal{S} , for narrowing derivations that result in **True**. When one such derivation is found, the composition of substitutions accumulated during the narrowing derivation in the reverse order gives us a solution of the goal G . Of course, the narrowing tree generated by \mathcal{S} has to be explored in a *fair* manner, since the narrowing derivations can be infinitely long. The reader is referred to [35] for further details and an example where the above technique is applied for analysis of safety properties of security protocols.

Further, note that since the set of initial and final states specified in a goal can be infinite, and likewise there is no restriction on the number of reachable states, one can view the above narrowing procedure as a new form of “symbolic model checking” for infinite state systems.

4.2 Theorem Proving

Equational unification: Narrowing was originally introduced as a complete method for generating all solutions of an equational unification problem, i.e., for goals F of the form $(\exists \vec{x}) t_1(\vec{x}) = t'_1(\vec{x}) \wedge \dots \wedge t_n(\vec{x}) = t'_n(\vec{x})$ in free algebras modulo a set of equations that can be described by a set of confluent and terminating rewrite rules [18,26,28]. We note that the problem of solving reachability goals in rewrite systems generalizes the problem of equational unification. Specifically, suppose we are to solve the equational goal F above in the equational theory $\mathcal{E} = (\Sigma, E)$ where the equations E are confluent and terminating. Note that σ is a solution of F if and only if both $\sigma(t_i)$ and $\sigma(t'_i)$ can be reduced by the (oriented) equations E to a common term. We can thus consider the rewrite system $\mathcal{R}_{\mathcal{E}} = (\tilde{\Sigma}, \phi, R_E)$, where $\tilde{\Sigma} = \Sigma \cup \{\approx, \mathbf{True}\}$, $\phi(f) = \emptyset$ for all $f \in \tilde{\Sigma}$, and $R_E = E \cup \{x \approx x \rightarrow \mathbf{True}\}$. Then σ is a solution of the system of equations F in the equational theory \mathcal{E} if and only if it is a solution of the reachability goal $G = (\exists \vec{x}) t_1 \approx t'_1 \rightarrow^* \mathbf{True} \wedge \dots \wedge t_n \approx t'_n \rightarrow^* \mathbf{True}$ in the rewrite system $\mathcal{R}_{\mathcal{E}}$.

Inductive theorem proving: The just-described reduction of existential equality goals to reachability goals, when combined with the reduction described in Section 4.1 of conjunctive reachability goals to a single goal has important applications to *inductive theorem proving*. Specifically, it is useful in proving existentially quantified inductive theorems like $E \vdash_{ind} (\exists \vec{x}) t = t'$ in the initial model defined by the equations E . Natural narrowing can, using the two reductions just mentioned, provide a very efficient semidecision procedure (and even a decision procedure for some restricted theories [37]) for proving such inductive goals because it will *detect failures* to unify, stopping with a counterexample instead of blindly expanding the narrowing tree. Furthermore, natural narrowing can make *inductionless induction* provers, particularly in the most recent formulations in [10,11], more effective and efficient, and can be used in such provers to prove also universal inductive theorems like $E \vdash_{ind} (\forall \vec{x}) t = t'$ (or, more generally, clauses). The point is that natural narrowing’s complete narrowing strategy can be used instead of the unrestricted narrowing carried out by *superposition* to compute a smaller set of deductions, which can increase the chances of termination of the inductionless induction procedure without loss of soundness. The

extended version of this work [16] illustrates the previous point with an example where inductionless induction is able to prove an inductive theorem with natural narrowing, but not with unrestricted narrowing.

4.3 Partial Evaluation

Partial evaluation (PE) is a semantics-based program transformation which specializes a program with respect to known parts of its input [27,1,2,12]. PE has been widely applied in the fields of term rewriting systems [29], functional programming [27,39], logic programming [12], and functional logic programming [1,2]. The different techniques for partial evaluation in all the different contexts share a common idea of narrowing sequences. Indeed, the use of narrowing has already been explicitly considered as a unifying framework for partial evaluation in [1,2]. The core of PE is that a finite narrowing tree is produced from the input term t by using a narrowing strategy and an unfolding rule (or stopping criteria) [1]. Then the specialized program is obtained from the leaves of this narrowing tree. The success of partial evaluation depends on having a good narrowing strategy and a good unfolding rule. As further discussed in the extended version of this work [16], the natural narrowing strategy can be very useful for PE applications because, by lazily instantiating only those variables that are needed, it can avoid combinatorial explosions while constructing the narrowing tree; furthermore, as illustrated by the PE example in [16], it can result in *smaller*, more efficient specialized programs than those obtained using other narrowing strategies.

4.4 Programming Languages

As mentioned in Section 1.1, one of the main features of many rewriting or narrowing strategies in the literature was to extend the class of rewrite systems where the strategy is correct and complete, although not optimal. In the case of this paper, this is one of the main contributions to the programming language area, since generalized natural rewriting and narrowing can serve as efficient lazy evaluation strategies for programming languages such as OBJ, CafeOBJ, ASF+SDF, Maude, and ELAN. The second contribution of this paper is that due to their generality, generalized natural rewriting and narrowing can be fruitfully applied to a much wider range of applications besides programming, such as those concerned with proving. In this regard, the reader may note that the theory transformation used in Section 4.1 introduces the non-linear rule $x \rightarrow x \rightarrow \text{True}$.

5 Conclusions and Future Work

We have generalized the narrowing strategy of [14] so that it can be applied to a much broader class of term rewrite systems. Specifically, the generalization drops the requirement that the rewrite system under consideration is left-linear and constructor-based, which is a typical assumption in functional (logic) programming. As a result of this generality, a much broader range of applications such as,

symbolic reachability analysis of concurrent systems, and theorem proving, can be efficiently supported by our strategy. What thus emerges is an efficient and unified mechanism based on narrowing, that seamlessly integrates programming and proving.

Since our generalization is conservative, we inherit all the optimality results presented in [14] for the class of left-linear constructor-based rewrite systems; note that the strategies in [14] are the best known for the class of left-linear constructor-based systems. An important problem for future research is to investigate optimality results of the generalized strategies for a larger class of rewrite systems. We believe that the notion of *inductively sequential terms* introduced in [14] can provide significant insights for the more general optimality results, since this notion identifies specific terms, rather than classes of rewrite systems, that can be optimally evaluated.

We observe that our generalized strategies are easily implementable, since the demanded set of redexes and narroxes are computed using simple recursive procedures. Indeed in [15], we have proposed a technique for the efficient implementation of the natural rewriting and narrowing strategies of [14] for the special class of left-linear constructor-based systems. Extending this implementation technique to the generalized strategies we have proposed is also a problem of interest.

Another interesting issue is to further generalize the strategies to the case where rewriting and narrowing are performed *modulo* a set of axioms (such as associativity, commutativity, and identity), as in languages such as ELAN [8] and Maude [9]. Specifically we are interested in strategies for rewrite theories of the form $\mathcal{R} = (\Sigma, \phi, E, R)$ where E is a set of axioms. A generalized narrowing strategy for such rewrite theories, that computes substitutions in an *incremental* manner, would have a very important efficiency advantage, since it will not explicitly use unification algorithms for the axioms E .

References

1. M. Alpuente, M. Falaschi, and G. Vidal. Partial Evaluation of Functional Logic Programs. *ACM Transactions on Programming Languages and Systems*, 20(4):768–844, 1998.
2. M. Alpuente, M. Falaschi, and G. Vidal. A Unifying View of Functional and Logic Program Specialization. *ACM Computing Surveys*, 30(3es):9es, 1998.
3. S. Antoy. Definitional trees. In *Proc. of the 3rd International Conference on Algebraic and Logic Programming ALP'92*, volume 632 of *Lecture Notes in Computer Science*, pages 143–157. Springer-Verlag, Berlin, 1992.
4. S. Antoy. Constructor-based conditional narrowing. In *Proc. of 3rd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 199–206, Florence, Italy, Sept. 2001. ACM.
5. S. Antoy, R. Echahed, and M. Hanus. Parallel evaluation strategies for functional logic languages. In *Proc. of the Fourteenth International Conference on Logic Programming (ICLP'97)*, pages 138–152. MIT Press, 1997.
6. S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Journal of the ACM*, volume 47(4), pages 776–822, 2000.

7. S. Antoy and S. Lucas. Demandness in rewriting and narrowing. In M. Comini and M. Falaschi, editors, *Proc. of the 11th Int'l Workshop on Functional and (Constraint) Logic Programming WFLP'02*, volume 76 of *Electronic Notes in Theoretical Computer Science*. Elsevier Sciences Publisher, 2002.
8. P. Borovanský, C. Kirchner, H. Kirchner, and P.-E. Moreau. ELAN from a rewriting logic point of view. *Theoretical Computer Science*, 285:155–185, 2002.
9. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, 2002.
10. H. Comon. Inductionless induction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 14, pages 913–962. Elsevier Science, 2001.
11. H. Comon and R. Nieuwenhuis. Induction = I-Axiomatization + First-Order Consistency. *Information and Computation*, 159(1/2):151–186, 2000.
12. D. De Schreye, R. Glück, J. Jørgensen, M. Leuschel, B. Martens, and M. Sørensen. Conjunctive Partial Deduction: Foundations, Control, Algorithms, and Experiments. *Journal of Logic Programming*, 41(2-3):231–277, 1999.
13. A. van Deursen, J. Heering, and P. Klint. *Language Prototyping: An Algebraic Specification Approach*. World Scientific, 1996.
14. S. Escobar. Refining weakly outermost-needed rewriting and narrowing. In D. Miller, editor, *Proc. of 5th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'03*, pages 113–123. ACM Press, New York, 2003.
15. S. Escobar. Implementing natural rewriting and narrowing efficiently. In Yuki Yoshi Kameyama and Peter J. Stuckey, editors, *7th International Symposium on Functional and Logic Programming (FLOPS 2004)*, volume 2998 of *Lecture Notes in Computer Science*, pages 147–162. Springer-Verlag, Berlin, 2004.
16. Santiago Escobar, José Meseguer, and Prasanna Thati. Natural narrowing as a general unified mechanism for programming and proving. Technical Report DSIC-II/16/04, DSIC, Universidad Politécnica de Valencia, 2004. Available at <http://www.dsic.upv.es/users/elp/papers.html>.
17. Santiago Escobar, José Meseguer, and Prasanna Thati. Natural rewriting for general term rewriting systems. In Sandro Etalle, editor, *Pre-proceedings of 14th International Workshop on Logic-based Program Synthesis and Transformation, LOPSTR'04*, 2004. Submitted for final proceedings.
18. M. Fay. First order unification in equational theories. In W. Bibel and R. Kowalski, editors, *4th Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 161–167. Springer, 1979.
19. K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.
20. E. Giovannetti, G. Levi, C. Moiso, and C. Palamidessi. Kernel Leaf: A Logic plus Functional Language. *Journal of Computer and System Sciences*, 42(2):139–185, 1991.
21. Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In *Software Engineering with OBJ: Algebraic Specification in Action*, pages 3–167. Kluwer, 2000.
22. Juan Carlos González-Moreno, M. Teresa Hortalá-González, Francisco J. López-Fraguas, and Mario Rodríguez-Artalejo. An approach to declarative programming based on a rewriting logic. *Journal of Logic Programming*, 40(1):47–87, 1999.
23. M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

24. M. Hanus, S. Lucas, and A. Middeldorp. Strongly sequential and inductively sequential term rewriting systems. *Information Processing Letters*, 67(1):1–8, 1998.
25. G. Huet and J.-J. Lévy. Computations in Orthogonal Term Rewriting Systems, Part I + II. In *Computational logic: Essays in honour of J. Alan Robinson*, pages 395–414 and 415–443. The MIT Press, Cambridge, MA, 1992.
26. J.M. Hullot. Canonical forms and unification. In W. Bibel and R. Kowalski, editors, *5th Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer, 1980.
27. N.D. Jones, C.K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
28. Jean-Pierre Jouannaud, Claude Kirchner, and Helene Kirchner. Incremental construction of unification algorithms in equational theories. In *10th International Colloquium on Automata, Languages and Programming*, volume 154 of *Lecture Notes in Computer Science*, pages 361–373. Springer, 1983.
29. L. Lafave and J.P. Gallagher. Constraint-based Partial Evaluation of Rewriting-based Functional Logic Programs. In *Proc. of 7th International Workshop on Logic-based Program Synthesis and Transformation, LOPSTR'97*, volume 1463 of *Lecture Notes in Computer Science*, pages 168–188. Springer-Verlag, Berlin, 1997.
30. R. Loogen, F. López-Fraguas, and M. Rodríguez-Artalejo. A Demand Driven Computation Strategy for Lazy Narrowing. In *Proc. of PLILP'93*, volume 714 of *Lecture Notes in Computer Science*, pages 184–200. Springer-Verlag, Berlin, 1993.
31. S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002.
32. Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. In D. M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 9, pages 1–88. Kluwer Academic Publishers, 2001.
33. José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
34. José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
35. José Meseguer and Prasanna Thati. Symbolic reachability analysis using narrowing and its application to analysis of cryptographic protocols. In *Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004. To appear, also available at <http://osl.cs.uiuc.edu/docs/wrla04/main.ps>.
36. A. Middeldorp. Call by Need Computations to Root-Stable Form. In *Proceedings of the 24th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 94–105. ACM, New York, 1997.
37. R. Nieuwenhuis. Basic paramodulation and decidable theories. In *Proceedings of the Eleventh Annual IEEE Symposium On Logic In Computer Science (LICS'96)*, pages 473–483, New York, USA, 1996. IEEE Computer Society Press.
38. R.C. Sekar and I.V. Ramakrishnan. Programming in equational logic: Beyond strong sequentiality. *Information and Computation*, 104(1):78–109, 1993.
39. M. H. Sørensen, R. Glück, and N. D. Jones. A positive supercompiler. *Journal of Functional Programming*, 6(6):811–838, 1996.
40. TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, 2003.
41. P. Thati and J. Meseguer. Complete symbolic reachability analysis using back-and-forth narrowing. 2004. Submitted for publication.

A Proofs of Section 3

Lemma 2 *Let l, t, σ , and $p \in FP(t) \cap Pos_X(t)$. If $l \blacktriangleleft t$, $p \in DP_l(t)$, and $p \notin DP_l(\sigma(t))$, then there exists $\theta \in DSub(t)$ s.t. $\theta(t|_p) \neq t|_p$ and $\theta|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$.*

Proof. First, note that $\sigma(t|_p) \neq t|_p$, since $p \in DP_l(t)$ but $p \notin DP_l(\sigma(t))$. Then, we have that there exists $\rho = \{t|_p \mapsto s\}$ such that $\rho \leq \sigma|_{\mathcal{V}ar(t)}$ and either (i) $s = f(\bar{w})$ for $f \in \Sigma$ and \bar{w} fresh variables, or (ii) $s = x$ for x a fresh variable. Now, we are done if we prove, for both cases, that there exists θ s.t. $\rho \leq \theta|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$ and $\theta \in DSub(t)$.

For case (i), we know that either $R_l(p) = \{p\}$ or $R_l(p) = \{p, p_1, \dots, p_k\}$ for $k > 0$. If $R_l(p) = \{p\}$, then $p \in Pos_\Sigma(l)$, since $R_l(p) = \{p\}$ implies that p is not under a variable in l . Since $p \notin DP_{\mathcal{R}}(\sigma(t))$, we have $root(l|_p) = f$ and $\rho \in DSub(t)$, by Definition 7. Then, the statement holds with $\theta = \rho = \{t|_p \mapsto f(\bar{w})\}$. If $R_l(p) = \{p, p_1, \dots, p_k\}$ for $k > 0$, then we have that p is under a non-linear variable in l . Since $p \notin DP_{\mathcal{R}}(\sigma(t))$, we have that σ partially unifies the set of terms $t|_p, t|_{p_1}, \dots, t|_{p_k}$, i.e., $root(\sigma(t|_p)) = root(\sigma(t|_{p_1})) = \dots = root(\sigma(t|_{p_k})) = f$. Then, by Definition 7, we have that there exists $\theta \in DSub(t)$ s.t. $\rho \leq \theta|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$, since for every $p' \in \{p, p_1, \dots, p_k\}$, either $root(t|_{p'}) \in X$ or $root(t|_{p'}) = f$. And the conclusion follows.

For case (ii), note that it is impossible that $R_l(p) = \{p\}$, otherwise $p \in Pos_\Sigma(l)$ and $\sigma(t|_p)$ cannot be a variable. Thus, $R_l(p) = \{p, p_1, \dots, p_k\}$ for $k > 0$. Then, we have that σ partially unifies the set of terms $t|_p, t|_{p_1}, \dots, t|_{p_k}$, i.e., $root(\sigma(t|_p)) = root(\sigma(t|_{p_1})) = \dots = root(\sigma(t|_{p_k})) = x$, and thus $p, p_1, \dots, p_k \in Pos_X(t)$. Hence, we can compute a $\theta \in DSub(t)$, by Definition 7, such that $\theta(t|_{p'}) = y$ for a unique fresh variable y and every $p' \in \{p, p_1, \dots, p_k\}$. Thus, the statement holds with such θ , where we have that $\rho|_{\mathcal{V}ar(t)} < \theta|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$. \square

Lemma 4. *Let $t, l \rightarrow r$, and σ . If $l \leq \sigma(t)$, then there exist θ such that $\theta|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$, $l \leq \theta(t)$, and $\langle \Lambda, l \rightarrow r, \theta \rangle \in DN(t)$.*

Proof. By induction on $\sigma|_{\mathcal{V}ar(t)}$. If $\sigma|_{\mathcal{V}ar(t)} = id$, it is immediate with $\theta = id$, since $l \leq t$ and $\langle \Lambda, l \rightarrow r, id \rangle \in DN(t)$. If $\sigma|_{\mathcal{V}ar(t)} \neq id$, then $DP_l(t) \subseteq Pos_X(t)$ and we have that any set cover computed by $FP(t)$ in Definition 4 has to include one variable position of $DP_l(t)$. Now, $l \leq \sigma(t)$ implies $l \blacktriangleleft t$ and, by Lemma 2, we have that there exist $\rho \in DSub(t)$ such that $\rho|_{\mathcal{V}ar(t)} \leq \sigma|_{\mathcal{V}ar(t)}$ and, by Definition 8, $DN(\rho(t))@_\rho \subseteq DN(t)$. Let η be such that $(\eta \circ \rho)|_{\mathcal{V}ar(t)} = \sigma|_{\mathcal{V}ar(t)}$. Then, by applying the induction hypothesis for $t' = \rho(t)$, $l \rightarrow r$ and η , we have that there exist ρ' such that $\rho'|_{\mathcal{V}ar(t')} \leq \eta|_{\mathcal{V}ar(t')}$, $l \leq \rho'(t')$, and $\langle \Lambda, l \rightarrow r, \rho' \rangle \in DN(t')$. Hence, we conclude with $\theta = (\rho' \circ \rho)|_{\mathcal{V}ar(t)}$, $\theta|_{\mathcal{V}ar(t')} \leq \sigma|_{\mathcal{V}ar(t')}$, $l \leq \theta(t)$, and $\langle \Lambda, l \rightarrow r, \rho' \circ \rho \rangle \in DN(\rho(t))@_\rho \subseteq DN(t)$. \square

Lemma 5. *Let t, σ , and $P = FP(\sigma(t))$. There exists $P' = FP(t)$ such that either $P \subseteq P'$ or there exist $l \in L(\mathcal{R})$ and $q \in DP_l(t) \cap P' \cap Pos_X(t)$ such that $l \blacktriangleleft t$ and $q \notin DP_l(\sigma(t))$.*

Proof. First, note that given an l , for each position $p \in DP_l(\sigma(t))$, there is a position $p' \in DP_l(t)$ s.t. $p' \leq p$. Indeed, if $p' \neq p$, then there is $p'' \in R_l(p')$ such that $p'' \in DP_l(t) \cap \mathcal{Pos}_X(t)$ and $\sigma(t|_{p''}) \neq t|_{p''}$. Furthermore, if $l \blacktriangleleft \sigma(t)$, then $p'' \notin DP_l(\sigma(t))$. Note also that $l \blacktriangleleft \sigma(t)$ implies $l \blacktriangleleft t$.

Now, let $P = \{p_1, \dots, p_n\}$, we have that there are rules $l_1, \dots, l_n \in L(\mathcal{R})$ such that $l_i \blacktriangleleft \sigma(t)$ and $p_i \in DP_{l_i}(\sigma(t))$ for $1 \leq i \leq n$. If there exists $P' = FP(t)$ such that $p_1, \dots, p_n \in P'$, we are done.

Let us assume that there is no P' such that $p_1, \dots, p_n \in P'$. In the following, we prove that there are $l \in L(\mathcal{R})$ and $q \in DP_l(t) \cap P' \cap \mathcal{Pos}_X(t)$ s.t. $l \blacktriangleleft t$ and $q \notin DP_l(\sigma(t))$. By the note above, we know that there exist p'_1, \dots, p'_n such that $p'_i \leq p_i$ and $p'_i \in DP_{l_i}(t)$ for $1 \leq i \leq n$. Let us consider whether there exists $P' = FP\mathcal{R}$ such that $p'_1, \dots, p'_n \in P'$ or not. If such set P' exists, then p'_1, \dots, p'_n cannot be exactly p_1, \dots, p_n , since we assumed that $p_1, \dots, p_n \notin P'$. Thus, by the note above and since P' is closed by reflection, we know that there exists an k , $1 \leq k \leq n$, such that $p'_k \in DP_{l_k}(t) \cap \mathcal{Pos}_X(t)$, $\sigma(t|_{p'_k}) \neq t|_{p'_k}$, and $p'_k \notin DP_{l_k}(\sigma(t))$. Hence, the statement is proved with $q = p'_k$ and $l = l_k$.

Let us assume that there is no P' such that $p'_1, \dots, p'_n \in P'$. Now, consider an arbitrary set P' . Since rules l_1, \dots, l_n are covered by P' , there are positions $\hat{p}_1, \dots, \hat{p}_n \in P'$ such that $\hat{p}_i \in DP_{l_i}(t)$, $1 \leq i \leq n$. Furthermore, since \hat{p} positions are included in P' but p' positions are not, we have that there is some other extra rule $l' \rightarrow r' \in \mathcal{R}$ such that $l' \blacktriangleleft t$, $DP_{l'}(t) \cap \{p'_1, \dots, p'_n\} = \emptyset$, $DP_{l'}(t) \cap \{\hat{p}_1, \dots, \hat{p}_n\} \neq \emptyset$, and either $l' \blacktriangleleft \sigma(t)$ or $DP_{l'}(\sigma(t)) = \emptyset$. But this means that there exists an k , $1 \leq k \leq n$, such that $\hat{p}_k \in DP_{l'}(t) \cap \mathcal{Pos}_X(t)$ and $\sigma(t|_{\hat{p}_k}) \neq t|_{\hat{p}_k}$. Now, if $DP_{l'}(\sigma(t)) = \emptyset$, then the statement is proved with $q = \hat{p}_k$ and $l = l'$. And if $l' \blacktriangleleft \sigma(t)$, then we have that $\hat{p}_k \in DP_{l_k}(t) \cap \mathcal{Pos}_X(t)$ and $\hat{p}_k \notin DP_{l_k}(\sigma(t))$, since $l_k \blacktriangleleft \sigma(t)$ and, by Definition 3, there is no defined symbol above or at position \hat{p}_k in $\sigma(t)$. Hence, the statement is proved with $q = \hat{p}_k$ and $l = l_k$. \square

Lemma 6. *Let t , σ , and $p \in \mathcal{Pos}_{\mathcal{D}}(t)$. If $p \in SP(\sigma(t))$, then $\exists \theta$ such that $\theta|_{\text{var}(t)} \leq \sigma|_{\text{var}(t)}$, $p \in SP(\theta(t))$, and $(p.DN(\theta(t)|_p))@ \theta \subseteq DN(\theta(t))@ \theta \subseteq DN(t)$.*

Proof. By induction on $\sigma|_{\text{var}(t)}$. The base case $\sigma|_{\text{var}(t)} = id$ is immediate. Thus, let us consider $\sigma|_{\text{var}(t)} \neq id$. We have that $p \in SP(\sigma(t))$ implies the following cases

1. there exists $l \in L(\mathcal{R})$ s.t. $l \leq \sigma(t)$ and $p \in D_l^\dagger(\mathcal{Pos}_X(l))$, or
2. there exist $l \in L(\mathcal{R})$ and $p' \in DP_l(\sigma(t))$ such that $l \blacktriangleleft \sigma(t)$, $p \leq p'$, and $p' \in FP(\sigma(t))$, i.e., $p \in D_{\sigma(t)}^\dagger(FP(\sigma(t)))$.

The first case is proved as follows. By Lemma 4, we have that there exists θ such that $\theta|_{\text{var}(t)} \leq \sigma|_{\text{var}(t)}$, $l \leq \theta(t)$, and $\langle \Lambda, l \rightarrow r, \theta \rangle \in DN(t)$. This implies that, by Definition 8, $\langle \Lambda, l \rightarrow r, id \rangle \in DN(\theta(t))@ \theta \subseteq DN(t)$ and $D_l^\dagger(\mathcal{Pos}_X(l)) \subseteq SP(\theta(t))$. Now, we have that $p \in D_l^\dagger(\mathcal{Pos}_X(l))$, since $D_l^\dagger(\mathcal{Pos}_X(l)) = D_l^\dagger(\mathcal{Pos}_X(l))$, and thus $p \in SP(\theta(t))$. Hence, by Definition 8, $(p.DN(\theta(t)|_p))@ \theta \subseteq DN(\theta(t))@ \theta \subseteq DN(t)$, and the statement is proved.

Now, let us consider the second case. By Lemma 5, there exists a set cover P' for the term t , i.e., $P' = FP(t)$, such that either $P \subseteq P'$ or there exist a

lhs $l' \in L(\mathcal{R})$ and a position $q \in DP_{l'}(t) \cap P' \cap \mathcal{Pos}_X(t)$ such that $l' \blacktriangleleft t$ and $q \notin DP_{l'}(\sigma(t))$. If $P \subseteq P'$, we are done, since $p \in \mathcal{Pos}_{\mathcal{D}}(t)$, $p \in D_t^\uparrow(P') \subseteq SP(t)$, and thus the statement holds with substitution $\theta = id$ and $(p.DN(t|_p)) \subseteq DN(t)$.

If there are $l' \in L(\mathcal{R})$ and $q \in DP_{l'}(t) \cap P' \cap \mathcal{Pos}_X(t)$ s.t. $l' \blacktriangleleft t$ and $q \notin DP_{l'}(\sigma(t))$, then, by Lemma 2, there exist substitutions ρ, η such that $\rho \in DSub(t)$, $\rho(t|_q) \neq t|_q$, $\sigma|_{\text{var}(t)} = (\eta \circ \rho)|_{\text{var}(t)}$, and, by Definition 8, $DN(\rho(t))@_\rho \subseteq DN(t)$. Now, by applying the induction hypothesis to $p \in SP(\eta(w))$ for $w = \rho(t)$, we have there exist ρ', η' s.t. $\eta|_{\text{var}(w)} = \eta' \circ \rho'|_{\text{var}(w)}$, $p \in SP(\rho'(w))$, and

$$(p.DN(\rho'(w)|_p))@_{\rho'} \subseteq DN(\rho'(w))@_{\rho'} \subseteq DN(w)$$

Hence, the statement is proved with substitution $\theta = (\rho' \circ \rho)|_{\text{var}(t)}$, $\sigma|_{\text{var}(t)} = (\eta \circ \rho)|_{\text{var}(t)} = (\eta' \circ \rho' \circ \rho)|_{\text{var}(t)}$, $p \in SP(\rho'(\rho(t)))$, and $(p.DN(\rho'(\rho(t))|_p))@_{\rho'}@_\rho \subseteq DN(\rho'(\rho(t))@_{\rho'})@_\rho \subseteq DN(\rho(t))@_\rho \subseteq DN(t)$. \square

Lemma 3 *For a normalized substitution σ , if $\sigma(t) \xrightarrow{m}_{\langle p, l \rightarrow r \rangle} s$, then there are η, θ, s' such that $t \xrightarrow{m}_{\langle p, l \rightarrow r, \theta \rangle} s'$, $\sigma|_{\text{var}(t)} = (\eta \circ \theta)|_{\text{var}(t)}$, $s = \eta(s')$, and η is normalized.*

Proof. We prove the lemma by induction on $|p|$. For the base case $p = \Lambda$, we know that $l \leq \sigma(t)$, and by Lemma 4, we have that there exist θ and a normalized η such that $\sigma|_{\text{var}(t)} = (\eta \circ \theta)|_{\text{var}(t)}$, $l \leq \theta(t)$, and $\langle \Lambda, l \rightarrow r, \theta \rangle \in DN(t)$. Hence, the statement holds with $t \xrightarrow{m}_{\langle \Lambda, l \rightarrow r, \theta \rangle} s'$, $s' = \rho(r)$ for some ρ s.t. $\theta(t) = \rho(l)$, and $s = \sigma(r) = \eta \circ \rho(r) = \eta(s')$.

Now, consider the induction step, $p > \Lambda$. By Definition 6 of $DR(\sigma(t))$, we have that there are p', p'' such that $p = p'.p''$, $p' \in SP(\sigma(t)) \setminus \{\Lambda\}$, and $\langle p'', l \rightarrow r \rangle \in DR(\sigma(t)|_{p'})$, i.e., $\sigma(t)|_{p'} \xrightarrow{m}_{\langle p'', l \rightarrow r \rangle} s|_{p'}$. Note that $p \in \mathcal{Pos}_{\mathcal{D}}(t)$, since σ is normalized. Then, by Lemma 6, there exist some θ and a normalized η such that $\sigma|_{\text{var}(t)} = (\eta \circ \theta)|_{\text{var}(t)}$, $p' \in SP(\theta(t))$, and $(p'.DN(\theta(t)|_{p'}))@_\theta \subseteq DN(\theta(t))@_\theta \subseteq DN(t)$. Now, by applying the induction hypothesis to $\eta(w) \xrightarrow{m}_{\langle p'', l \rightarrow r \rangle} u$ where $w = \theta(t)|_{p'}$ and $u = s|_{p'}$, we have that there exist θ', u' , and a normalized η' such that $w \xrightarrow{m}_{\langle p'', l \rightarrow r, \theta' \rangle} u'$, $\eta|_{\text{var}(w)} = (\eta' \circ \theta')|_{\text{var}(w)}$, and $u = \eta'(u')$, i.e. $\langle p'', l \rightarrow r, \theta' \rangle \in DN(\theta(t)|_{p'})$. Hence, $\langle p'.p'', l \rightarrow r, \theta' \circ \theta \rangle \in (p'.DN(\theta(t)|_{p'}))@_\theta \subseteq DN(\theta(t))@_\theta \subseteq DN(t)$ and the statement holds with $t \xrightarrow{m}_{\langle p, l \rightarrow r, \theta' \circ \theta \rangle} s'$, $s' = \theta' \circ \theta(t)[u']_{p'}$, $\sigma|_{\text{var}(t)} = (\eta \circ \theta)|_{\text{var}(t)} = (\eta' \circ \theta' \circ \theta)|_{\text{var}(t)}$, and $s = \sigma(t)[u]_{p'} = \eta' \circ \theta' \circ \theta(t)[u]_{p'} = \eta'(\theta' \circ \theta(t)[u']_{p'}) = \eta'(s')$. \square

Theorem 2 *If t is not a variable and is a \xrightarrow{m} -normal form, then $\sigma(t)$ is root-stable for every normalized σ .*

Proof. We prove the contrapositive. Specifically, suppose for a normalized substitution σ , we have that $\sigma(t)$ is not root-stable. Then, by Lemma 1, we have $\sigma(t) \xrightarrow{m}_{\langle p, l \rightarrow r \rangle} s$ for some s . Then, using Lemma 3, we conclude that t is not a \xrightarrow{m} -normal form. \square

Theorem 3 *If $\sigma(t) \rightarrow^* s$ and σ is a normalized substitution, then there are s', θ, θ' such that $t \overset{\mathfrak{m}}{\rightsquigarrow}_{\theta}^* s'$, $\theta'(s') \xrightarrow{\geq \Lambda}^* s$, and $\sigma|_{\text{var}(t)} = (\theta' \circ \theta)|_{\text{var}(t)}$.*

Proof. By Theorem 1, we have that there exists w such that $\sigma(t) \overset{\mathfrak{m}}{\rightarrow}^* w$, $\text{root}(w) = \text{root}(s)$ and $w \xrightarrow{\geq \Lambda}^* s$. By a simple induction on the length of $\sigma(t) \overset{\mathfrak{m}}{\rightarrow}^* w$ using Lemma 3, we deduce that there exists s', θ, θ' such that $t \overset{\mathfrak{m}}{\rightsquigarrow}_{\theta}^* s'$, $\sigma|_{\text{var}(t)} = (\theta' \circ \theta)|_{\text{var}(t)}$, and $w = \theta'(s')$. \square