

# Natural rewriting and narrowing for general term rewriting systems

Santiago Escobar<sup>1</sup>, José Meseguer<sup>2</sup>, and Prasanna Thati<sup>2</sup>

<sup>1</sup> Universidad Politécnica de Valencia, Spain  
sescobar@dsic.upv.es

<sup>2</sup> University of Illinois at Urbana-Champaign, USA  
{meseguer, thati}@cs.uiuc.edu

**Abstract.** We address the problem of efficient rewriting and narrowing strategies for general term rewriting systems. Several strategies have been proposed over the last two decades, the most efficient of all being the natural rewriting and narrowing strategies of Escobar. All the strategies so far, including natural rewriting and narrowing, assume that the given term rewriting system is left-linear and constructor-based. Although these restrictions are reasonable for some functional programming languages, they limit the expressive power of equational languages, and they preclude certain applications of narrowing to equational theorem proving and to languages combining equational and logic programming. In this paper, we propose a conservative generalization of natural rewriting and narrowing that does not require the rules to be left-linear and constructor-based. We also establish the soundness and completeness of these generalizations.

## 1 Introduction

In this paper we address the problem of efficient rewriting and narrowing strategies for general term rewriting systems. The problem was first addressed for rewriting in a seminal paper by Huet and Levy [10], where the *strongly needed reduction* strategy was proposed. Several refinements of this strategy have been proposed over the last two decades, the most significant ones being Sekar and Ramakrishnan's *parallel needed reduction* [13] and Antoy, Echahed and Hanus' *(weakly) outermost-needed rewriting* [1,2]. Narrowing is an extension of rewriting where pattern matching is substituted by unification (as in logic programming). It has several applications in programming languages and in automated deduction (see [9] for a survey). Antoy, Echahed and Hanus extended their (weakly) outermost-needed rewriting strategy to *(weakly) needed narrowing* strategy [3,2]. Recently, both (weakly) outermost-needed rewriting and (weakly) outermost-needed narrowing have been improved by Escobar by means of the *natural rewriting* and *natural narrowing* strategies [6].

A typical assumption of the above rewriting and narrowing strategies is that the rewrite rules are left-linear and constructor-based. These restrictions are reasonable for some functional programming languages, but they limit the expressive power of equational languages such as OBJ [8], CafeOBJ [7], ASF+SDF [15],

and *Maude* [5], where non-linear left-hand sides are perfectly acceptable. This extra generality is also necessary for applications of narrowing to equational theorem proving, and to languages combining equational and logic programming, since in both cases assuming left-linearity is too restrictive. Furthermore, for rewrite systems whose semantics is not equational but is instead rewriting logic based, such as rewrite rules in *ELAN*[4], or *Maude* system modules, the constructor-based assumption is unreasonable and almost never holds. In summary, generalizing natural rewriting and narrowing to *general* rewriting systems will extend the scope of applicability of the strategies to more expressive equational languages and to rewriting logic based languages, and will open up a much wider range of applications.

*Example 1.* Consider the following TRS for proving equality of arithmetic expressions built using modulus (i.e., remainder) and subtraction operations on natural numbers.

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

Note that this TRS is not left-linear because of rule 5 and is not constructor-based because of rule 3. Therefore, it is outside the scope of all the strategies mentioned above. Now, consider the term  $t_1 = 10! \% 0$ , where  $!$  denotes the auxiliary factorial function. If we only had rules (1), (2), and (4), the natural rewriting strategy [6] would be applicable and no reductions on  $t_1$  would be performed since  $t_1$  is a head-normal form. In contrast, the other strategies mentioned above, for example, outermost-needed rewriting, would force the evaluation of the subterm  $10!$  (see [6, Example 21]). We would like to generalize natural rewriting to a version that enjoys this optimality and that can also handle non-left-linear and non-constructor-based rules such as (3) and (5).

Now, consider the term  $t_2 = 10! \% (1-1) \approx 10! \% 0$ . We would like the generalized natural rewriting strategy to perform only the optimal computation:

$$\begin{aligned} & 10! \% (s(0) - s(0)) \approx 10! \% 0 \\ & \rightarrow 10! \% (0 - 0) \approx 10! \% 0 \rightarrow \underline{10! \% 0} \approx 10! \% 0 \rightarrow \text{True} \end{aligned}$$

that avoids unnecessary reduction of the subterm  $10! \% 0$ . We expect the generalized narrowing strategy to have a similar optimality feature while narrowing the terms  $t_3 = X \% 0$  and  $t_4 = X \% (1-1) \approx X \% 0$ , where instantiations of the variable  $X$  to  $0$  and  $s(Y)$  are unnecessary and hence can be avoided.

Natural rewriting and narrowing [6] use a more refined demandedness notion in comparison with other strategies such as needed rewriting and narrowing [1,3]. This leads to a very efficient lazy evaluation strategy. In this paper, we propose a conservative generalization of this demandedness notion that drops the assumptions that the rewrite rules are left-linear and constructor-based, and therefore can be used for general TRS's. We show soundness and completeness of the generalized strategy w.r.t. (head-)normal forms.

## 2 Generalizing Natural Rewriting and Narrowing

A TRS is a pair  $\mathcal{R} = (\mathcal{F}, R)$  where  $R$  is a set of rewrite rules.  $L(\mathcal{R})$  denotes the set of *lhs*'s of  $\mathcal{R}$ . Given  $\mathcal{R} = (\mathcal{F}, R)$ , we take  $\mathcal{F}$  as the disjoint union  $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$  of symbols  $c \in \mathcal{C}$ , called *constructors* and symbols  $f \in \mathcal{D}$ , called *defined functions*, where  $\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} - \mathcal{D}$ . By  $\mathcal{P}os(t)$  we denote the set of positions of a term  $t$ . Given a set  $S \subseteq \mathcal{F} \cup \mathcal{X}$ ,  $\mathcal{P}os_S(t)$  denotes positions in  $t$  where symbols in  $S$  occur. We denote the root position by  $\Lambda$ . The subterm at position  $p$  of  $t$  is denoted as  $t|_p$ , and  $t[s]_p$  is the term  $t$  with the subterm at position  $p$  replaced by  $s$ . The rewriting and narrowing relations are defined as usual; see [14] for details. The relations  $\xrightarrow{\Lambda}$  and  $\xrightarrow[\Lambda]$  denote a rewriting and narrowing step respectively, at a position strictly under  $\Lambda$ .

The natural rewriting and natural narrowing strategies [6] perform outermost evaluations which are demanded for applying a left-hand side of a rule at the root position. In fact, the strategy performs only those outermost evaluations which are the most frequently demanded by different rules. We first start with the formal definitions of the positions in a term  $t$  that are demanded for evaluation by the left-hand side  $l$  of a rule which is to be applied at the root position of  $t$ . The following definition introduces the set of disagreeing positions between two terms  $t, t'$ .

**Definition 1.** *Given two terms  $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we define the set of disagreeing positions between  $t$  and  $t'$  as*

$$\mathcal{P}os_{\neq}(t, t') = \text{minimal}_{\leq}(\{p \in \mathcal{P}os(t) \cap \mathcal{P}os(t') \mid \text{root}(t|_p) \neq \text{root}(t'|_p)\})$$

Now, we define the set  $DP_l(t)$  containing all the positions in  $t$  that are demanded by a left-hand side  $l$ .

**Definition 2.** *We define the set of demanded positions between a term  $t$  and a left-hand side  $l$  of a rule as  $DP_l(t) = DP_{\neq}(t, l) \cup DP_{\mathcal{X}}(t, l)$ , where*

1.  $DP_{\neq}(t, l) = \mathcal{P}os_{\neq}(t, l) \cap \mathcal{P}os_{\mathcal{F}}(l)$
2.  $DP_{\mathcal{X}}(t, l) = \bigcup_{\substack{x \in \text{Var}(l) \\ p, q \in \mathcal{P}os_x(l)}} \left\{ \begin{array}{l} p.\mathcal{P}os_{\neq}(t|_p, t|_q) \\ \cup \\ q.\mathcal{P}os_{\neq}(t|_p, t|_q) \end{array} \middle| \begin{array}{l} \forall p' < p, q' < q : \\ p', q' \notin \mathcal{P}os_{\neq}(t, l) \end{array} \right\}$

Since we are only interested in matching  $l$  to  $t$  (or to unify in case of narrowing), the set  $DP_{\neq}(t, l)$  does not include those disagreeing positions at which  $l$  has a variable. To account for non-linear variables in  $l$ , we define the set  $DP_{\mathcal{X}}(t, l)$ . This set contains the disagreeing positions between subterms  $t|_p$  and  $t|_q$  for all pairs of position  $p, q$  in  $t$  that: (i) correspond to positions in  $l$  occupied by the same non-linear variable, and (ii)  $t$  and  $l$  match along the path from the root to  $p$  and  $q$ . These positions are indeed demanded for  $l$  to match  $t$ .

*Example 2.* Continuing Example 1, consider the left-hand side  $l = X \approx X$  and the term  $t_2 = 10! \% (1-1) \approx 10! \% 0$ . We have  $\mathcal{P}os_{\neq}(t_2, l) = \{1, 2\}$  and  $DP_{\neq}(t_2, l) = \emptyset$ . Further,  $DP_{\mathcal{X}}(t_2, l) = \{1.2, 2.2\}$ , which implies that the subterm  $1-1$  is demanded for evaluation. For the term  $t_4 = X \% (1-1) \approx X \% 0$ ,

we have  $DP_{\mathcal{X}}(t_4, l) = \{1.2, 2.2\}$ , i.e., the variable  $X$  is not demanded, and hence will not be instantiated by our narrowing strategy. However, for  $t_5 = 10! \% X \approx 10! \% 0$ , we have  $DP_{\mathcal{X}}(t_5, l) = \{1.2, 2.2\}$ , i.e., the variable  $X$  is demanded, and hence will be instantiated by the narrowing strategy.

Now, we formally define the case where it is impossible to match (or unify)  $l$  and  $t$  even after possible evaluations in  $t$ .

**Definition 3.** *Let  $t, l \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . We say  $DP_l(t)$  is failing if either*

1. *there exist  $p \in DP_{\neq}(t, l)$  such that and for all  $p' \leq p$  (different from  $\Lambda$ ),  $root(p') \in \mathcal{C}$ ; or*
2. *there exist  $p, q \in DP_{\mathcal{X}}(t, l)$ , a non-linear variable  $x$  in  $l$ , and  $p', q' \in Pos_x(l)$  such that  $p' < p$ ,  $q' < q$ , and for all  $p'' \leq p$  and  $q'' \leq q$  (both different from  $\Lambda$ ),  $root(p''), root(q'') \in \mathcal{C}$ .*

*Example 3.* For  $t_1 = 10! \% 0$  and every left-hand side  $l$  that is rooted by  $\%$  in Example 1, we have that  $DP_l(t_1)$  is failing because  $t_1|_2 = 0$ , whereas  $root(l|_2) = \mathbf{s}$ . Likewise, for  $l' = X \approx X$  and  $t' = \mathbf{s}(10! \% 0) \approx 0$ , we have  $DP_{l'}(t')$  is failing because  $t'|_2 = 0$ , whereas  $root(t'|_1) = \mathbf{s}$ . Now, consider the term  $t'' = \mathbf{s}(\mathbf{s}(\mathbf{s}(0))) \% \mathbf{s}(\mathbf{s}(0)) \approx \mathbf{s}(0) \% \mathbf{s}(\mathbf{s}(0))$ . We have  $root(t''|_{1.1.1}) \neq root(t''|_{2.1.1})$ , but  $DP_{l'}(t'')$  is not failing, because several evaluations under position 1 (with rules 1, 4, 2 in that order) would make  $t''|_1$  and  $t''|_2$  equal.

The above example motivates the following definition that includes all the defined symbols above disagreeing positions in the demanded set.

**Definition 4.** *Given a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and a left-hand side  $l$  of a rule, we define the set of demanded positions in  $t$  as  $DP_l^\dagger(t) = \emptyset$  if  $DP_l(t)$  is failing, and otherwise  $DP_l^\dagger(t) = DP_l(t) \cup \{q \in Pos_{\mathcal{D}}(t) - \{\Lambda\} \mid p \in DP_l(t) \wedge q < p\}$ .*

*Example 4.* Continuing Example 3, we have  $DP_{l'}^\dagger(t') = \emptyset$  and  $DP_{l'}^\dagger(t'') = \{1.1.1, 2.1.1, 1, 2\}$  because  $root(t''|_1) = root(t''|_2) = \%$ , which is a defined symbol. Also, for  $t_2 = 10! \% (1-1) \approx 10! \% 0$ , we have  $DP_{l'}^\dagger(t_2) = \{1.2, 2.2, 1, 2\}$ .

Since our strategy is to evaluate only the most frequently demanded positions, we keep track of the number of times each position is demanded by different rules using multisets (see [6] for details).

**Definition 5.** *We define the multiset of demanded positions of a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  w.r.t. TRS  $\mathcal{R}$  as  $DP_{\mathcal{R}}(t) = \oplus\{DP_l^\dagger(t) \mid l \rightarrow r \in \mathcal{R} \wedge root(t) = root(l)\}$  where  $M_1 \oplus M_2$  denotes the union of multisets  $M_1$  and  $M_2$ .*

*Example 5.* Consider  $t = 10! \% (1 - 1)$ . For the TRS  $\mathcal{R}$  of Example 1, we have  $DP_{\mathcal{R}}(t) = \{1, 2, 2\}$ .

We now pick the most frequently demanded positions in such a way that we cover all the rules involved in evaluating the term (see [6] for details).

**Definition 6.** We define the set of demanded positions of a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  w.r.t. TRS  $\mathcal{R}$  as  $FDP_{\mathcal{R}}(t) = \left\{ p \mid \begin{array}{l} \exists l \in L(\mathcal{R}). p \in DP_l^\uparrow(t) \text{ and} \\ \forall q \in FDP_{\mathcal{R}}(t) : p <_{DPR(t)} q \Rightarrow q \notin DP_l^\uparrow(t) \end{array} \right\}$  where  $x <_M y$  denotes that the number of occurrences of  $x$  in the multiset  $M$  is less than the number of occurrences of  $y$ .

*Example 6.* Continuing Example 5, we have that  $FDP_{\mathcal{R}}(t) = \{2\}$ , and hence only the subterm  $1 - 1$  is evaluated, and not  $10!$ .

Following are the definitions of our natural rewriting and narrowing strategies that only pick the most frequently demanded positions.

**Definition 7 (Natural rewriting).** Given a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and a TRS  $\mathcal{R}$ ,  $\mathbf{m}(t)$  is defined as the smallest set satisfying

$$\mathbf{m}(t) \ni \begin{cases} (\Lambda, l \rightarrow r) & \text{if } l \rightarrow r \in \mathcal{R} \text{ and } l \leq t \\ (p.q, l \rightarrow r) & \text{if } p \in FDP_{\mathcal{R}}(t) \text{ and } (q, l \rightarrow r) \in \mathbf{m}(t|_p) \end{cases}$$

*Example 7.* Consider the term  $t_2 = 10! \% (1-1) \approx 10! \% 0$  in Example 1. Natural rewriting performs the following optimal (desired) sequence of evaluations:

$$\begin{aligned} & 10! \% (\underline{\mathbf{s}(0) - \mathbf{s}(0)}) \approx 10! \% 0 \\ & \xrightarrow{\mathbf{m}} 10! \% (\underline{0 - 0}) \approx 10! \% 0 \xrightarrow{\mathbf{m}} \underline{10! \% 0} \approx 10! \% 0 \xrightarrow{\mathbf{m}} \mathbf{True} \end{aligned}$$

**Definition 8 (Natural narrowing).** Given a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and a TRS  $\mathcal{R}$ ,  $\mathbf{mn}(t)$  is defined as the smallest set satisfying

$$\mathbf{mn}(t) \ni \begin{cases} (\Lambda, id, l \rightarrow r) & \text{if } l \rightarrow r \in \mathcal{R} \text{ and } l \leq t \\ (p.q, \theta, l \rightarrow r) & \text{if } p \in FDP_{\mathcal{R}}(t) \cap \mathcal{P}os_{\mathcal{D}}(t) \text{ and } (q, \theta, l \rightarrow r) \in \mathbf{mn}(t|_p) \\ (p, \theta \circ \sigma, l \rightarrow r) & \text{if } p \in FDP_{\mathcal{R}}(t) \cap \mathcal{P}os_{\mathcal{X}}(t) \text{ and } (p, \theta, l \rightarrow r) \in \mathbf{mn}(\sigma(t)) \\ & \text{where either } \sigma(t|_p) = f(\bar{w}) \text{ for } f \in \mathcal{F} \text{ and } \bar{w} \text{ fresh, or} \\ & \sigma(t|_p) = y \text{ for a variable } y \in \mathcal{V}ar(t) \setminus \{t|_p\} \end{cases}$$

*Example 8.* Consider the term  $t_4 = \mathbf{X} \% (1-1) \approx \mathbf{X} \% 0$  in Example 1. Natural narrowing performs the following optimal (desired) sequence of evaluations:

$$\begin{aligned} & \mathbf{X} \% (\underline{\mathbf{s}(0) - \mathbf{s}(0)}) \approx \mathbf{X} \% 0 \\ & \xrightarrow{\mathbf{m}}_{id} \mathbf{X} \% (\underline{0 - 0}) \approx \mathbf{X} \% 0 \xrightarrow{\mathbf{m}}_{id} \underline{\mathbf{X} \% 0} \approx \mathbf{X} \% 0 \xrightarrow{\mathbf{m}}_{id} \mathbf{True} \end{aligned}$$

Note that Definitions 7 and 8 specialize to the original definitions in [6] for the case of left-linear constructor-based TRS's.

The following are statements of correctness and completeness of natural rewriting and narrowing for general term rewriting systems. For soundness of natural rewriting we show that the normal forms w.r.t. the natural rewriting strategy are indeed root-stable forms.

**Theorem 1 (Rewriting Correctness).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS. If  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is a  $\xrightarrow{\mathbf{m}}$ -normal form, then  $s$  is root-stable.

Conversely, for completeness we show that the natural rewriting strategy can approximate any given rewrite sequence in an appropriate sense.

**Theorem 2 (Rewriting Completeness).** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $t \rightarrow^* s$ , then  $\exists s' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  s.t.  $t \xrightarrow{m}^* s'$ ,  $\text{root}(s') = \text{root}(s)$  and  $s' \xrightarrow{\Delta}^* s$ .*

Correctness of natural narrowing is similar to that of natural rewriting.

**Theorem 3 (Narrowing Correctness).** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS. If  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is a  $\xrightarrow{m}$ -normal form, then  $s$  is root-stable.*

Completeness of natural narrowing is also similar, except that in addition, a given narrowing sequence can be approximated using a natural narrowing derivation that produces a more general substitution.

**Theorem 4 (Narrowing Completeness).** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $t \rightsquigarrow_{\sigma}^* s$ , then  $\exists s' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  s.t.  $t \rightsquigarrow_{\theta}^* s'$ ,  $\text{root}(s') = \text{root}(s)$ ,  $s' \xrightarrow{\Delta}_{\theta'}^* s$ , and  $\theta' \circ \theta \leq \sigma$ .*

### 3 Conclusions

We have extended natural rewriting and narrowing to general rewriting systems. This makes these strategies available for both expressive equational languages and for rewriting logic languages. In case of these latter languages, i.e. rewrite systems specify concurrent systems [11], narrowing provides a general method for *reachability analysis* in such systems [12]. This work provides a basis for a subsequent generalization of natural rewriting and narrowing to even more expressive rewrite theories suited for concurrent system specifications and supporting: (i) sorts and subsorts; (ii) rewriting *modulo* axioms such as associativity, commutativity, identity, and so on; and (iii) conditional rewriting.

### References

1. S. Antoy. Definitional trees. In *Proc. of the 3rd International Conference on Algebraic and Logic Programming ALP'92*, LNCS 632:143–157. Springer, 1992.
2. S. Antoy, R. Echahed, and M. Hanus. Parallel evaluation strategies for functional logic languages. In *Proc. of ICLP'97*, pages 138–152. MIT Press, 1997.
3. S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Journal of the ACM*, volume 47(4), pages 776–822, 2000.
4. 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.
5. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, 2002.
6. S. Escobar. Refining weakly outermost-needed rewriting and narrowing. In *Proc. of PPDP'03*, pages 113–123. ACM Press, 2003.

7. K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.
8. J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In *Software Engineering with OBJ: Algebraic Specification in Action*, pages 3–167. Kluwer, 2000.
9. M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.
10. 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, 1992.
11. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
12. J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. In *Proc. of WRLA'04*, ENTCS to appear. Elsevier, 2004.
13. R. Sekar and I. Ramakrishnan. Programming in equational logic: Beyond strong sequentiality. *Information and Computation*, 104(1):78–109, 1993.
14. TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, 2003.
15. A. van Deursen, J. Heering, and P. Klint. *Language Prototyping: An Algebraic Specification Approach*. World Scientific, 1996.