From Outermost to Context-Sensitive Rewriting

Jörg Endrullis and Dimitri Hendriks

Vrije Universiteit Amsterdam
{joerg,diem}@few.vu.nl

Abstract. We define a transformation from term rewriting systems (TRSs) to context-sensitive TRSs in such a way that termination of the target system implies outermost termination of the original system. For the class of left-linear TRSs the transformation is complete. Thereby state-of-the-art termination methods and automated termination provers for context-sensitive rewriting become available for proving termination of outermost rewriting. The translation has been implemented in Jambox, making it the most successful tool in the category of outermost rewriting of the last edition of the annual termination competition.

1 Introduction

Termination is a key aspect of program correctness, and therefore a widely studied subject in term rewriting and program verification. While termination is undecidable in general, various automated techniques have been developed for proving termination. One of the most powerful techniques is the method of dependency pairs [2]. Recently [1], this method has been generalized to context-sensitive TRSs, thereby significantly extending the class of context-sensitive TRSs for which termination can be shown automatically. Context-sensitive rewriting [6] is a restriction on term rewriting where rewriting in some fixed arguments of function symbols is disallowed. It offers a flexible paradigm to analyze properties of rewrite strategies, in particular of (lazy) evaluation strategies employed by functional programming languages.

In this paper context-sensitive rewriting is the target formalism for a transformational approach to the problem of outermost termination, that is, termination with respect to outermost rewriting. Outermost rewriting is a rewriting strategy where a redex may be contracted as long as it is not a proper subterm of another redex occurrence. The main reason for studying outermost termination is its practical relevancy: lazy functional programming languages like Miranda, Haskell or Clean, are based on outermost rewriting as an evaluation strategy, and in implementations of rewrite logic such as Maude and CafeOBJ, outermost rewriting can be specified. Consider the TRS R_0 consisting of the following rules:

$$a \to f(a) \qquad\qquad f(f(x)) \to b \qquad\qquad (R_0)$$

Clearly, this system is not terminating as witnessed by the infinite rewrite sequence $a \to f(a) \to f(f(a)) \to f(f(f(a))) \to \ldots$, but it is outermost terminating. Indeed, the third step in the infinite sequence is not an outermost step, since

the contraction takes place inside another redex. The only (maximal) outermost rewrite sequence the term a admits is $a \to f(a) \to f(f(a)) \to b$.

Our contribution is a transformation of arbitrary TRSs into context-sensitive TRSs (μTRSs) in such a way that rewriting in the μTRS corresponds to outermost rewriting in the original TRS. As a result advanced termination techniques for μTRSs become applicable for proving outermost termination, and automated termination provers for μTRSs can directly (without modification, only preprocessing) be used for proving outermost termination. Our transformation is complete for the class of quasi left-linear TRSs (a generalized form of left-linear TRSs, see [8]), that is, termination of the resulting μTRS is equivalent to outermost termination of the original system.

The transformation is comprised of a variant of semantic labeling [12]. In semantic labeling the function symbols in a term are labeled by the interpretation of their arguments (or a label depending on these values) according to some given semantics. We employ semantic labeling in order to mark symbols at redex positions, and we obtain a μTRS by defining a replacement map that disallows rewriting inside arguments of marked symbols.

We illustrate our use of semantic labeling by the TRS R_0 from the first page. We choose the algebra $\mathcal{A}_0 = \langle \{0,1\}, [\cdot] \rangle$ where the interpretation indicates the presence of the symbol f, thus $[a] = [b] = 0$, and $[f](x) = 1$ for $x \in \{0,1\}$. Then we write $f^\star$ if the value of its argument is 1, and just f if the value is 0. The symbol a is always marked, and b never is. If f is marked it corresponds to a redex position with respect to the rule $f(f(x)) \to b$. For example the term $f(f(f(a)))$ is labeled as $f^\star(f^\star(f(a^\star)))$. We obtain a μTRS by forbidding rewriting inside the argument of the symbol $f^\star$; since $a^\star$ is a constant, there is nothing to be forbidden. Then for correctly labeled terms, rewriting inside redex positions is disallowed, corresponding to the outermost rewriting strategy. In order to rewrite labeled terms we have to label the rules of the TRS. By labeling R_0 with the algebra $\mathcal{A}_0$ we obtain the μTRS $\overline{R}_0$:

$$a^\star \to f(a^\star) \qquad f^\star(f(x)) \to b \qquad f^\star(f^\star(x)) \to b \qquad (\overline{R}_0)$$

which has two instances of the second rule, one for each possible value of x.

Now, despite the fact that the original TRS is outermost terminating, the transformed μTRS $\overline{R}_0$ admits an infinite rewrite sequence:

$$a^\star \to f(a^\star) \to f(f(a^\star)) \to f(f(f(a^\star))) \to \dots \qquad (1)$$

The reason is that the term $f(f(a^\star))$ is not correctly labeled, as the root symbol f should have been marked. In [12] this problem is avoided by allowing labeling only with 'models'. Roughly, an algebra is a model if left- and right-hand sides of all rewrite rules have equal interpretations. However, this requirement is too strict for the purpose of marking redexes, because contraction of a redex at a position p may create a redex above p in the term tree, as witnessed by (1). In fact, for R_0 there exists *no* model which is able to distinguish between redex and non-redex positions. The rewrite step $f(a) \to f(f(a))$ creates a redex at the top. The term $f(a)$ is not a redex, and therefore its root symbol f should not

be marked. On the other hand $f(f(a))$ is a redex and so the outermost f has to be marked. The change of the labeling of a context (here $f(\Box)$) implies that the interpretation of its arguments a and $f(a)$ cannot be the same. Therefore we cannot require the rule $a \to f(a)$ to preserve the interpretation.

For this reason, we generalize the concept of model, and relax the requirement $[\ell] = [r]$ to $\exists n.\ [C[\ell]] = [C[r]]$ for all contexts C of depth n. Thus rules are allowed to change the interpretation as long as the effect is limited to contexts of a bounded depth. Algebras satisfying this weaker requirement, which we call 'C-models', are strong enough to recognize redex positions. In particular, the algebra $\mathcal{A}_0$ given above is a C-model for the system R_0. As demonstrated by the rewrite sequence (1) in the μTRS $\overline{R}_0$, for C-models it is no longer sufficient to simply label the rules: an application of the rule $a^\star \to f(a^\star)$ in the term $f(a^\star)$ creates the invalid labeled term $f(f(a^\star))$. In order to preserve correct labeling, we sometimes need to extend the rewrite rules by putting contexts around their left- and right-hand sides. Using the C-model $\mathcal{A}_0$, our algorithm transforms R_0 into the μTRS $\triangle^\pi_{\mathcal{A}_0}(R_0)$, which truthfully simulates outermost rewriting of R_0:

$$\begin{array}{cc} f(a^\star) \to f^\star(f(a^\star)) & \mathsf{top}(f^\star(f(x))) \to \mathsf{top}(b) \\ \mathsf{top}(a^\star) \to \mathsf{top}(f(a^\star)) & \mathsf{top}(f^\star(f^\star(x))) \to \mathsf{top}(b) \end{array} \qquad (\triangle^\pi_{\mathcal{A}_0}(R_0))$$

We explain this magical transformation. The rule $f(a^\star) \to f^\star(f(a^\star))$ is obtained from prepending the context $f(\Box)$ to $a \to f(a)$. This enables correct updating of the labeling of the context during rewriting. Because we still have to allow rewrite steps $a \to f(a)$ of the original TRS at the top of a term, we extend the signature with a unary function symbol top which represents the top of a term. Thus when prepending contexts we include $\mathsf{top}(\Box)$, giving rise to the rule $\mathsf{top}(a^\star) \to \mathsf{top}(f(a^\star))$. The necessity of the symbol top becomes especially apparent when considering the rule $f(f(x)) \to b$. Here prepending the context $f(\Box)$ is not even an option since $f(f(f(x))) \to f(b)$ is not an outermost rewrite step; this rule can only be applied at the top of a term. Hence we get the two rules displayed on the right, one for each possible interpretation of the variable x.

Semantic labeling has the nice property that it does not complicate termination proofs. Although semantic labeling increases the search space, termination proofs for the unlabeled system carry over to the labeled one. That is, whenever R' is a labeling of a TRS R and $\mathcal{A} = \langle A, [\cdot], \succ, \sqsupseteq \rangle$ is a monotone Σ-algebra [4] which proves termination of R, then extending $[\cdot]$ to the labeled signature Σ' by $[f^\lambda] = [f]$ for every $f \in \Sigma$ and label λ, yields a monotone Σ'-algebra which proves termination of R'. Consequently our transformation, based on a variant of semantic labeling, also does not complicate termination proofs. On the contrary, the labeled systems often allow for simpler proofs arising from the enriched signature which provides more freedom for the choice of interpretations, see [12].

Semantic labeling possibly creates extra copies of rules, and our extension might even create more copies: one for each context that has to be prepended. Despite of this fact, the implementation of our transformation in the termination prover Jambox performs efficiently on the set of examples from the Termination Problem Database (TPDB [10]).

Jambox was best in proving termination in the category of outermost rewriting of the termination competition of 2008 [10], see Figure 1. With an average time of 4.1 seconds per termination proof, Jambox was also faster than the other participants, providing empirical evidence for the efficiency of our transformation. Not listed in Figure 1 is TTT2, which did not prove outermost termination, but performed best in disproving outermost termination.

	score	average time
Jambox	72 (93.5%)	4.1s
TrafO	46 (59.7%)	8.1s
AProVE	27 (35.0%)	10.8s

Fig. 1. Results of proving outermost termination in the competition of 2008.

The secret behind the efficiency of Jambox is threefold: First, we construct and minimize the algebras employed for marking redex positions, see Sections 4 and 5. Secondly, we try two labeling strategies: minimal and maximal. Minimal labeling is very efficient and contributes to 75% of the success of Jambox. In order to have a complete transformation we also employ maximal labeling. Both labelings are described in Section 6. Thirdly, we combine labeling and context extension into what we call 'dynamic labeling', where contexts are prepended depending on the interpretation of the variables. This is formalized in Section 3. All these optimizations minimize the number of rules and their size in the transformed μTRS, which is important to keep a manageable search space.

Related work. Cariboo [5] deals with outermost termination using a stand-alone approach based on induction. The very idea for a transformational approach to outermost termination comes from [8]. There the signature is enriched with unary symbols top, up, and down and the TRS is extended with 'anti-matching' rules such that $\mathsf{down}(t)$ is a redex if and only if t is not a redex with respect to the original TRS. The idea is that the symbol down is moved down in the term tree as long as no redex is encountered. Once a redex is encountered, a rewrite step is performed, and the symbol down is replaced by up, which then moves upwards again to the top of the term, marked by top. This transformation is implemented in TrafO. Based on a similarly elegant idea, Thiemann [11] defines a complete transformation from outermost to innermost rewriting, which is implemented in AProVE. For traversal to the redex positions, rules of the form $\mathsf{down}(\mathsf{isRedex}(f(\ldots))) \to f(\ldots,\mathsf{down}(\mathsf{isRedex}(\ldots)),\ldots)$ are used. In order to simulate outermost rewriting and to prevent from moving inside redexes, rules $\mathsf{isRedex}(\ell) \to \mathsf{up}(r)$ are added for every rule $\ell \to r$ of the original TRS. Then, by the innermost rewriting strategy, the latter rules have priority over the traversal rules, whenever an original redex is encountered. The simplicity of both approaches is attractive, but the yo-yoing effect in the resulting TRSs makes that the original outermost rewrite steps are 'hidden' among a vast amount of auxiliary steps. This increases derivational complexity, and makes it hard for automated termination provers to find proofs for the transformed systems.

2 Preliminaries

For a general introduction to term rewriting and to context-sensitive rewriting, the reader is referred to [9] and [6], respectively. Here we repeat some of the main definitions, for the sake of completeness, and to fix notations.

A *signature* Σ is a non-empty set of symbols each having a fixed *arity*, given by a mapping $\sharp : \Sigma \to \mathbb{N}$. Given Σ and a set $\mathcal{X}$ of variables, the *set* $\mathit{Ter}(\Sigma, \mathcal{X})$ *of terms over* Σ is the smallest set satisfying: $\mathcal{X} \subseteq \mathit{Ter}(\Sigma, \mathcal{X})$, and $f(t_1, \ldots, t_n) \in \mathit{Ter}(\Sigma, \mathcal{X})$ if $f \in \Sigma$ of arity n and $t_i \in \mathit{Ter}(\Sigma, \mathcal{X})$ for all $1 \leq i \leq n$. We use $x, y, z, \ldots$ to range over variables, and write $\mathit{Var}(t)$ for the set of variables occurring in a term t. Usually we leave $\mathcal{X}$ implicit and write $\mathit{Ter}(\Sigma)$ for the set of terms over Σ and a fixed, countably infinite set of variables $\mathcal{X}$. The set of positions $\mathcal{P}os(t) \subseteq \mathbb{N}^*$ of a term $t \in \mathit{Ter}(\Sigma)$ is defined as follows: $\mathcal{P}os(x) = \{\epsilon\}$ for variables $x \in \mathcal{X}$ and $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{ip \mid 1 \leq i \leq \sharp f,\ p \in \mathcal{P}os(t_i)\}$. We write $t(p)$ to denote the root symbol of $t|_p$, the subterm of t rooted at p, and we write $\mathit{root}(t)$ for the root symbol of t, that is $\mathit{root}(t) = t(\epsilon)$.

A *substitution* σ is a map $\sigma : \mathcal{X} \to \mathit{Ter}(\Sigma, \mathcal{X})$ from variables to terms. For terms $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ and substitutions σ, $t\sigma$ is inductively defined by $x\sigma = \sigma(x)$ if $x \in \mathcal{X}$, and $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ otherwise. Let $\Box$ be a fresh symbol, i.e. $\Box \notin \Sigma \cup \mathcal{X}$. A *context* C is a term from $\mathit{Ter}(\Sigma, \mathcal{X} \cup \{\Box\})$ containing precisely one occurrence of $\Box$. By $C[s]$ we denote the term $C\sigma$ where $\sigma(\Box) = s$ and $\sigma(x) = x$ for all $x \in \mathcal{X}$. The *depth* of a context C is defined as the length $|p|$ of the position p at which $\Box$ resides, that is, the position p such that $C(p) = \Box$.

A *term rewriting system (TRS)* over Σ is a set R of pairs $\langle \ell, r\rangle \in \mathit{Ter}(\Sigma, \mathcal{X})^2$, called *rewrite rules* and written as $\ell \to r$, for which the *left-hand side* ℓ is not a variable ($\ell \notin \mathcal{X}$) and all variables in the *right-hand side* r occur in ℓ: $\mathit{Var}(r) \subseteq \mathit{Var}(\ell)$. For a TRS R we define $\to_R$, the *rewrite relation* induced by R as follows. For terms $s, t \in \mathit{Ter}(\Sigma, \mathcal{X})$ we write $s \to_R t$, or just $s \to t$ if R is clear from the context, if there exists a rule $\ell \to r \in R$, a substitution σ and a context $C \in \mathit{Ter}(\Sigma, \mathcal{X} \cup \{\Box\})$ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$; we sometimes write $s \to_{R,p} r$ to explicitly indicate the rewrite position p, i.e. when $C(p) = \Box$. Then s *outermost rewrites to* t at a position $p \in \mathcal{P}os(s)$, denoted by $s \xrightarrow{\text{out}}_{R,p} t$, if $s \to_{R,p} t$ and for all positions p' that are a proper prefix of p: $s|_{p'}$ is not a redex.

A mapping $\mu : \Sigma \to \mathbf{2}^{\mathbb{N}}$ is called a *replacement map (for* Σ*)* if for all $f \in \Sigma$ we have $\mu(f) \subseteq \{1, \ldots, \sharp f\}$. A *context-sensitive term rewriting system (*μ*TRS)* is a pair $\langle R, \mu\rangle$ consisting of a TRS R and a replacement map μ. The set of μ*-replacing positions* $\mathcal{P}os^\mu(t)$ of a term $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ is defined by $\mathcal{P}os^\mu(x) = \{\epsilon\}$ for $x \in \mathcal{X}$ and $\mathcal{P}os^\mu(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{ip \mid i \in \mu(f),\ p \in \mathcal{P}os^\mu(t_i)\}$. In context-sensitive term rewriting only redexes at μ-replacing positions are contracted: s *μ-rewrites to* t, denoted $s \to_{R,\mu} t$, whenever $s \to_{R,p} t$ with $p \in \mathcal{P}os^\mu(s)$.

A Σ*-algebra* $\langle A, [\cdot]\rangle$ consists of a non-empty set A and for each n-ary $f \in \Sigma$ a function $[f] : A^n \to A$, called the *interpretation of* f. Given an *assignment* $\alpha : \mathcal{X} \to A$, the interpretation of terms $t \in \mathit{Ter}(\Sigma)$ is defined by: $[x, \alpha] = \alpha(x)$ and $[f(t_1, \ldots, t_n), \alpha] = [f]([t_1, \alpha], \ldots, [t_n, \alpha])$. For substitutions $\sigma : \mathcal{X} \to \mathit{Ter}(\Sigma, \mathcal{X})$, we write $[\sigma, \alpha]$ for the function $\lambda x.[\sigma(x), \alpha]$. For ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$ and substitutions $\sigma : \mathcal{X} \to \mathit{Ter}(\Sigma, \varnothing)$ we write $[t]$ and $[\sigma]$ for short, respectively.

3 Transformation by Dynamic Labeling

In outermost rewriting the only redexes which are allowed to be rewritten are those which are not nested within any other redex occurrence. We represent this strategy by context-sensitive rewriting by using semantic labeling: we mark the symbols which are the root of a redex in order to disallow rewriting within that redex. We first recall the definition of semantic labeling and models from [12], and then generalize these to fit our purpose.

Definition 3.1 ([12]). Let Σ be a signature. A *semantic labeling* $\langle A, [\cdot], \pi \rangle$ consists of a Σ-algebra $\langle A, [\cdot] \rangle$ and a family $\pi = \{\pi_f\}_{f \in \Sigma}$ of functions $\pi_f : A^{\sharp f} \to \Lambda_f$ where, for each $f \in \Sigma$, Λ_f is a finite and non-empty set of labels. For a term $t \in \mathit{Ter}(\Sigma)$ and $\alpha : \mathit{Var}(t) \to A$, an interpretation of its variables, we define the *labeling* $\mathit{lab}(t, \alpha)$ *of* t *with respect to* α inductively as follows:

$$\mathit{lab}(x, \alpha) = x \,,$$
$$\mathit{lab}(f(t_1, \ldots, t_n), \alpha) = f^{\pi_f([t_1,\alpha],\ldots,[t_n,\alpha])}(\mathit{lab}(t_1, \alpha), \ldots, \mathit{lab}(t_n, \alpha)) \,.$$

For ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$ we just write $\mathit{lab}(t)$. Let R be a TRS over Σ. The *semantic labeling of* R is the TRS $\mathit{lab}(R)$ over the labeled signature $\mathit{lab}(\Sigma) = \{f^\lambda \mid f \in \Sigma,\ \lambda \in \Lambda_f\}$, defined by:

$$\mathit{lab}(R) = \{\mathit{lab}(\ell, \alpha) \to \mathit{lab}(r, \alpha) \mid \ell \to r \in R \,,\ \alpha : \mathit{Var}(\ell) \to A\} \,.$$

Term labeling satisfies the following useful property; see [12] for a proof.

Lemma 3.2 ([12]). *Let* $\langle A, [\cdot] \rangle$ *be a* Σ*-algebra,* $\alpha : \mathcal{X} \to A$*,* $\sigma : \mathcal{X} \to \mathit{Ter}(\Sigma)$*, and* $\overline{\sigma}(x) = \mathit{lab}(\sigma(x), \alpha)$*. Then* $\mathit{lab}(t\sigma, \alpha) = \mathit{lab}(t, [\sigma, \alpha])\overline{\sigma}$*, for all* $t \in \mathit{Ter}(\Sigma)$*.*

The Σ-algebra of a semantic labeling has to satisfy certain constraints in order to obtain that a TRS is terminating if and only if its labeled version is. In [12] the algebra has to be a 'model': A Σ-algebra $\langle A, [\cdot] \rangle$ is called a *model* of a TRS R if for all rules $\ell \to r \in R$ and assignments of variables $\alpha : \mathit{Var}(\ell) \to A$ we have that $[\ell, \alpha] = [r, \alpha]$. In the introduction we explained why this notion of model is too restrictive for our purpose. In order to be able to distinguish between redex and non-redex positions we introduce C-models, a generalization of models.

Definition 3.3. A *C-model* for a TRS R is a Σ-algebra $\langle A, [\cdot] \rangle$ such that for every rule $\ell \to r \in R$ there exists $n \in \mathbb{N}$ such that for every context C of depth n and assignment $\alpha : \mathcal{X} \to A$ we have $[C[\ell], \alpha] = [C[r], \alpha]$. When $n \in \mathbb{N}$ is minimal for a rule $\ell \to r$ with respect to this property, we call n the *C-depth for* $\ell \to r$.

As this definition suggests, it is possible to use a C-model $\mathcal{A}$ to transform a TRS R by prepending contexts to its rules in such a way that $\mathcal{A}$ becomes a model for the transformed system $\tilde{R}$, and then perform semantic labeling. But then, from the labeled version $\mathit{lab}(\tilde{R})$, every rule that contains a marked (redex) symbol within the context that was prepended to the rule in the construction of $\tilde{R}$ has to be removed again, as it would enable a rewrite step which is not outermost.

We choose for a different approach which we call 'dynamic labeling'. We stepwise extend rules by contexts, only when needed and dependent on the variable

interpretation used for the semantic labeling. For different interpretations of the variables usually different context depths are necessary for achieving equal interpretations of left- and right-hand side. In each extension step we check whether a candidate symbol is a redex symbol, and, if it is, this symbol is excluded from prepending. Here, by a redex symbol we mean a labeled symbol which indicates the presence of a redex in the original system (at the same position). Dynamic labeling is more efficient in that both the number and the size of the rules of the resulting μTRS are smaller than in the 'static' version. We explain dynamic labeling by means of the TRS R_1 consisting of the rules:

$$f(g(x)) \to f(f(g(x))) \qquad f(f(f(x))) \to x \qquad (R_1)$$

and the algebra $\mathcal{A}_1 = \langle A_1, [\cdot] \rangle$ where $A_1 = \{g, f_0, f_1, f_2\}$ and where the interpretation of the symbols is defined by $[g](x) = g$ for all $x \in A_1$, $[f](g) = f_1$ and $[f](f_i) = f_{\min(i+1,2)}$ for $i = 0, 1, 2$. Let further $\langle \mathcal{A}_1, \pi \rangle$ be the semantic labeling where π labels the symbols with the interpretations of their arguments. Then the symbols f^g and f^{f_2} are redex symbols, corresponding to redex positions with respect to the first and the second rule of R_1, respectively. The algebra $\mathcal{A}_1$ is a C-model where for the first rule the C-depth is 1, and for the second rule it is 2.

We iteratively construct sets $P_0, P_1, \ldots$, until $P_{i+1} = P_i$ for some i. The initial set P_0 consists of pairs $\langle \ell \to r, \alpha \rangle$ for each rule $\ell \to r$, and each interpretation $\alpha : \mathit{Var}(\ell) \to A_1$ of the variables. Then, in each step, P_{i+1} is obtained from P_i by replacing every pair $\langle \ell \to r, \alpha \rangle$ of P_i for which the interpretation of the left-hand side differs from the right-hand side ($[\ell, \alpha] \neq [r, \alpha]$), by the pairs $\langle C[\ell] \to C[r], \alpha' \rangle$ for every flat context C (see (2) on the next page) and every extension $\alpha' : \mathit{Var}(C[\ell]) \to A_1$ of α, such that the root of the labeled, extended left-hand side $\mathit{lab}(C[\ell], \alpha')$ is not a redex symbol. Among the flat contexts to prepend we include $\mathsf{top}(\Box)$ to cater for the case that the rule is applied at the top of the term. For R_1 the initial set P_0 is:

$$P_0 = \left\{ \langle f(g(x)) \to f(f(g(x))), \lambda x.a \rangle, \langle f(f(f(x))) \to x, \lambda x.a \rangle \mid a \in A_1 \right\}.$$

The only element $\langle \ell \to r, \alpha \rangle$ of P_0 such that $[\ell, \alpha] = [r, \alpha]$ is $\langle f(f(f(x))) \to x, \lambda x.f_2 \rangle$. For this pair no context needs to be prepended. The other pairs have to be replaced by their context extensions, and thus P_1 consists of:

$$\begin{aligned}
&\langle C[f(g(x))] \to C[f(f(g(x)))], \lambda x.a \rangle, \quad \text{for all } a \in A_1,\ C \in \{\mathsf{top}(\Box), f(\Box), g(\Box)\},\\
&\qquad \langle f(f(f(x))) \to x, \lambda x.f_2 \rangle,\\
&\qquad \langle C[f(f(f(x)))] \to C[x], \lambda x.a \rangle, \quad \text{for all } a \in A_1 \setminus \{f_2\},\ C \in \{\mathsf{top}(\Box), g(\Box)\}.
\end{aligned}$$

In the last line the context $f(\Box)$ is excluded, because the labeled left-hand side of the rule would contain the redex symbol f^{f_2} within the prepended context, and thus the step would not be outermost. Due to the outermost strategy, the original rule is only applicable under a context $C[g(\Box)]$ (where C does not contain any redexes) or at the top of a term. Now for all $(4 \cdot 3 + 1 + 3 \cdot 2 = 19)$ pairs of P_1 left- and right-hand side have equal interpretations, and the iterative construction is ended. We define $\triangle_{\mathcal{A}_1}(R_1) = P_1$, and call this set the 'dynamic context extension' of R_1 with respect to $\mathcal{A}_1$.

Secondly, the dynamic extension $\triangle_{\mathcal{A}_1}(R_1)$ is labeled using the family π which returns for each symbol f a label consisting of the interpretation of its arguments, i.e., $\pi_f(a_1, \ldots, a_{\sharp f}) = \langle a_1, \ldots, a_{\sharp f} \rangle$. Then the desired μTRS $\triangle^{\pi}_{\mathcal{A}_1}(R_1)$, which we call the *dynamic labeling of* R_1, consists of the rules $lab(\ell, \alpha) \to lab(r, \alpha)$ for every $\langle \ell \to r, \alpha \rangle \in \triangle_{\mathcal{A}_1}(R_1)$, with the replacement map μ defined by $\mu(f) = \varnothing$ if $f \in \{f^g, f^{f_2}\}$, and $\mu(f) = \{1, \ldots, \sharp f\}$ otherwise, for all $f \in lab(\Sigma)$.

We now formalize dynamic labeling. For the remainder of this section we fix an arbitrary TRS R over Σ, and let $\mathcal{A} = \langle A, [\cdot] \rangle$ be a C-model for R. We assume top to be a fresh symbol ($\mathsf{top} \notin \Sigma$) representing the top of a term, and abbreviate $\Sigma_{\mathsf{top}} = \Sigma \cup \{\mathsf{top}\}$. We extend $\mathcal{A}$ to a Σ_{top}-algebra by choosing an arbitrary but fixed element $a \in A$ and defining the interpretation of top as the constant function $[\mathsf{top}] = \lambda x.a$. Furthermore, we let $\langle \mathcal{A}, \pi \rangle$ be a semantic labeling, and $\Sigma^{red} \subseteq lab(\Sigma)$ a subset of the labeled signature, called the set of *redex symbols*. The definition makes use of *flat contexts fresh for* $t \in Ter(\Sigma_{\mathsf{top}})$:

$$\mathcal{C}^{\flat}_t = \left\{ f(x_1, \ldots, x_{j-1}, \Box, x_{j+1}, \ldots, x_{\sharp f}) \mid f \in \Sigma_{\mathsf{top}},\ j \in \{1, \ldots, \sharp f\} \right\} \qquad (2)$$

where $x_1, x_2, \ldots \in \mathcal{X}$ such that $x_i \notin Var(t)$. Furthermore, for partial functions $f, g : S \rightharpoonup T$ with disjoint domains, we write $f+g$ for the *union of* f *and* g, defined by $(f + g)(x) = f(x)$ if $x \in dom(f)$, and $(f + g)(x) = g(x)$ if $x \in dom(g)$.

Definition 3.4. The *dynamic context extension of* R, denoted by $\triangle_{\mathcal{A}}(R)$, is defined as the fixed point of the following construction of sets $P_0, P_1, \ldots$, that is, $\triangle_{\mathcal{A}}(R) = P_i$ as soon as $P_{i+1} = P_i$ for some i. The initial set is defined by:

$$P_0 = \left\{ \langle \ell \to r, \alpha \rangle \mid \ell \to r \in R,\ \alpha : Var(\ell) \to A \right\},$$

and for $i = 0, 1, \ldots$ the set P_{i+1} is obtained from P_i by replacing every pair $\langle \ell \to r, \alpha \rangle$ such that $[\ell, \alpha] \neq [r, \alpha]$, or $r \in \mathcal{X}$,[1] by all pairs in $\triangle(\ell \to r, \alpha)$ where:

$$\triangle(\ell \to r, \alpha) = \left\{ \langle C[\ell] \to C[r], \alpha + \beta \rangle \;\middle|\; C \in \mathcal{C}^{\flat}_{\ell},\ \beta : Var(C) \to A,\ root(lab(C[\ell], \alpha + \beta)) \notin \Sigma^{red} \right\}.$$

Then, the *dynamic labeling of* R is the μTRS $\langle \triangle^{\pi}_{\mathcal{A}}(R), \mu \rangle$ consisting of:

$$\triangle^{\pi}_{\mathcal{A}}(R) = \left\{ lab(\ell, \alpha) \to lab(r, \alpha) \mid \langle \ell \to r, \alpha \rangle \in \triangle_{\mathcal{A}}(R) \right\},$$

and the replacement map μ, defined by $\mu(f) = \varnothing$ if $f \in \Sigma^{red}$, and $\mu(f) = \{1, \ldots, \sharp f\}$ otherwise, for all $f \in lab(\Sigma_{\mathsf{top}})$. Whenever the set Σ^{red}, which determines the replacement map, is clear from the context, we write $\triangle^{\pi}_{\mathcal{A}}(R)$ as a shorthand for $\langle \triangle^{\pi}_{\mathcal{A}}(R), \mu \rangle$.

Notice that the construction of $\triangle_{\mathcal{A}}(R)$ is guaranteed to terminate because of the assumption that $\mathcal{A}$ is a C-model.

We come to the first main theorem, stating that outermost ground termination of R is implied by termination of the transformed system $\triangle^{\pi}_{\mathcal{A}}(R)$.

[1] The condition $r \notin \mathcal{X}$ eliminates collapsing rules. This is used in the proof of Thm. 6.5, which states completeness. Without this condition, the transformation is still sound (Thm. 3.7). Nota bene: in the TRS R_1 worked out before, $r \notin \mathcal{X}$ is not used.

For the remainder of this section, we assume that the set $\Sigma^{red} \subseteq lab(\Sigma)$ contains redex symbols only, that is, for all ground terms t and $p \in \mathcal{P}os(t)$:

$$lab(t)(p) \in \Sigma^{red} \text{ implies that } t|_p \text{ is a redex with respect to } R \qquad (\ddagger)$$

Lemma 3.5. *Let $s, t \in Ter(\Sigma, \varnothing)$ be ground terms and $p \in \mathcal{P}os(s)$ such that $s \xrightarrow{out}_{R,p} t$. Then for all proper prefixes q of $1p$ we have $lab(\mathsf{top}(s))(q) \notin \Sigma^{red}$.*

Proof. If $q = \epsilon$, this follows from $\mathsf{top}^\lambda \notin \Sigma^{red}$ for any label λ. If $q \neq \epsilon$, then $lab(\mathsf{top}(s))(q) = lab(s)(q')$ with q' a proper prefix of p, and if $lab(s)(q') \in \Sigma^{red}$, then by assumption $(\ddagger)$ the term s contains a redex at position q', quod non. □

The following lemma states that any outermost ground rewrite step in R can be transformed into a rewrite step in $\triangle^{\pi}_{\mathcal{A}}(R)$. For ground substitutions $\sigma : \mathcal{X} \to Ter(\Sigma, \varnothing)$ define $\overline{\sigma}(x) = lab(x\sigma)$.

Lemma 3.6. *Let $s, t \in Ter(\Sigma, \varnothing)$ be ground terms such that $s \xrightarrow{out}_R t$. Then:*

$$lab(\mathsf{top}(s)) \to_{\triangle^{\pi}_{\mathcal{A}}(R)} lab(\mathsf{top}(t)) \,.$$

Proof. Assume $s \xrightarrow{\text{out}}_{R,p} t$ for some position $p \in \mathcal{P}os(s)$. Then there exists a rule $\ell \to r \in R$, a context C with $C(p) = \Box$ and a ground substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$. We consider the construction of the dynamic context extension from Definition 3.4, and prove by induction that for all $i = 0, 1, \ldots$ there exists a context C_i which is a prefix of $\mathsf{top}(C)$, a ground substitution σ_i, and terms ℓ_i, r_i such that $\mathsf{top}(s) = C_i[\ell_i\sigma_i]$, $\mathsf{top}(t) = C_i[r_i\sigma_i]$ and $\langle \ell_i \to r_i, [\sigma_i] \rangle \in P_i$. For the base case we have $\langle \ell_0 \to r_0, [\sigma_0] \rangle \in P_0$ with $\ell_0 = \ell$, $r_0 = r$, $\sigma_0 = \sigma$, and $C_0 = \mathsf{top}(C)$. For the induction step we assume the existence of C_i, σ_i, and $\langle \ell_i \to r_i, [\sigma_i] \rangle \in P_i$ with the above properties. If $[\ell_i, [\sigma_i]] = [r_i, [\sigma_i]]$ and $r_i \notin \mathcal{X}$ then by definition $\langle \ell_i \to r_i, [\sigma_i] \rangle \in P_{i+1}$, and so we are done. For the remaining cases $[\ell_i, [\sigma_i]] \neq [r_i, [\sigma_i]]$ and $r_i \in \mathcal{X}$, we first show that $C_i \neq \Box$. If $[\ell_i, [\sigma_i]] \neq [r_i, [\sigma_i]]$ and $C_i = \Box$, then $\ell_i\sigma_i = \mathsf{top}(s)$ and $r_i\sigma_i = \mathsf{top}(t)$, and hence $root(\ell_i) = root(r_i) = \mathsf{top}$, contradicting $[\ell_i, [\sigma_i]] \neq [r_i, [\sigma_i]]$ (recall that the interpretation of top is constant). Furthermore, we have $r_i \in \mathcal{X}$ only if $i = 0$, and then $C_i = \mathsf{top}(C) \neq \Box$. Thus we have $C_i = D[D'\sigma']$ for some context D, flat context $D' \in \mathcal{C}^{\flat}_{\ell_i}$ and substitution σ'. We choose $C_{i+1} = D$, $\ell_{i+1} = D'[\ell_i]$, $r_{i+1} = D'[r_i]$, and $\sigma_{i+1} = \sigma_i + \sigma'$. It remains to be shown that $\langle \ell_{i+1} \to r_{i+1}, [\sigma_{i+1}] \rangle \in P_{i+1}$. For this it suffices to prove that $root(lab(\ell_{i+1}, [\sigma_{i+1}])) \notin \Sigma^{red}$. We have $C_{i+1}[\ell_{i+1}\sigma_{i+1}] = \mathsf{top}(s)$. Let q be the position such that $C_{i+1}(q) = \Box$. Then, by Lemma 3.2 we obtain $root(lab(\ell_{i+1}, [\sigma_{i+1}])) = root(lab(\ell_{i+1}\sigma_{i+1})) = \mathsf{top}(lab(s))(q)$. Note that q is a proper prefix of $1p$, and so $lab(s)(q') \notin \Sigma^{red}$ by Lemma 3.5.

Let $i \in \mathbb{N}$ be such that $P_{i+1} = P_i$. By the result above we have $\langle \ell_i \to r_i, [\sigma_i] \rangle \in \triangle_{\mathcal{A}}(R)$, with $[\ell_i\sigma_i] = [r_i\sigma_i]$, and then $lab(\ell_i, [\sigma_i]) \to lab(r_i, [\sigma_i]) \in \triangle^{\pi}_{\mathcal{A}}(R)$ by definition. Let τ and υ be defined by $\tau(\Box) = \ell_i\sigma_i$, $\upsilon(\Box) = r_i\sigma_i$, and $\tau(x) = \upsilon(x) = x$ for $x \in \mathcal{X}$. Then we have that $lab(C_i, [\tau]) = lab(C_i, [\upsilon])$ since $[\tau] = [\upsilon]$. Let $E = lab(C_i, [\tau])$. We get $lab(\mathsf{top}(s)) = lab(C_i[\ell_i\sigma_i]) = lab(C_i\tau) = E\overline{\tau} = E[lab(\ell_i\sigma_i)] = E[lab(\ell_i, [\sigma_i])\overline{\sigma_i}]$ and $lab(\mathsf{top}(t)) = \ldots = E[lab(r_i, [\sigma_i])\overline{\sigma_i}]$, by Lemma 3.2. Then, by Lemma 3.5 we have $lab(\mathsf{top}(s)) \to_{\triangle^{\pi}_{\mathcal{A}}(R)} lab(\mathsf{top}(t))$. □

Theorem 3.7. *R is outermost ground terminating if $\triangle^{\pi}_{\mathcal{A}}(R)$ is terminating.*

Proof. Assume R admits an infinite outermost rewrite sequence $t_1 \overset{\text{out}}{\to}_R t_2 \overset{\text{out}}{\to}_R t_3 \overset{\text{out}}{\to}_R \ldots$. Then from Lemma 3.6 it follows that $\triangle^{\pi}_{\mathcal{A}}(R)$ allows an infinite rewrite sequence: $lab(\mathsf{top}(t_1)) \to_{\triangle^{\pi}_{\mathcal{A}}(R)} lab(\mathsf{top}(t_2)) \to_{\triangle^{\pi}_{\mathcal{A}}(R)} lab(\mathsf{top}(t_3)) \to_{\triangle^{\pi}_{\mathcal{A}}(R)} \ldots$. □

Theorem 3.7 is about outermost ground termination. This is not a restriction because by adding a fresh constant 0 and a fresh unary symbol s, outermost ground termination coincides with outermost termination:

Lemma 3.8. *The TRS R over the signature Σ is outermost terminating if and only if R over the signature $\Sigma \cup \{s, 0\}$ is outermost ground terminating.* □

4 Constructing Suitable Algebras

In this section we construct C-models that are able to recognize redex positions with respect to left-linear rules. The construction of C-models is similar to the construction of a deterministic tree automaton (DTA) for recognizing left-linear redexes. A DTA is a Σ-algebra $\langle A, [\cdot]\rangle$ with a distinguished set $A_F \subseteq A$ of final states. A term t is accepted by the automaton whenever $[t] \in A_F$. The difference with the construction of a DTA is that for the construction of a C-model we do not distinguish final and non-final states, but instead have a family of functions $isRedex_f : A^{\sharp f} \to Bool$ for indicating the presence of a redex.

Definition 4.1. A *redex-algebra* $\mathcal{A} = \langle A, [\cdot], isRedex\rangle$ is a Σ-algebra $\langle A, [\cdot]\rangle$ with a family $\{isRedex_f\}_{f\in\Sigma}$ of functions $isRedex_f : A^{\sharp f} \to Bool$. The *language of* $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) = \{f(t_1, \ldots, t_n) \in Ter(\Sigma, \varnothing) \mid isRedex_f([t_1], \ldots, [t_n]) = true\}$.

By this separation of tasks our approach allows for smaller algebras, because, intuitively, the algebra needs to 'remember' only the subterms t_1,...,t_n and not $f(t_1, \ldots, t_n)$ itself. To see this, consider the single-rule system:

$$f(g(x)) \to a\,.$$

A tree automaton recognizing redex positions for this TRS needs at least three states: one for indicating a redex $f(g(\ldots))$, one for $g(\ldots)$, and one garbage state. For redex-algebras two states suffice: one state for $g(\ldots)$ and one for garbage. Then $isRedex_f(g(\ldots)) = true$ and $false$, otherwise.

The 'core' of a redex-algebra consists of all interpretations of ground terms:

Definition 4.2. Let $\mathcal{A} = \langle A, [\cdot], isRedex\rangle$ be a redex-algebra over Σ. The *core of* $\mathcal{A}$ is the redex-algebra $\mathcal{A}_c = \langle A_c, [\cdot]_c, isRedex_c\rangle$ where A_c is the smallest set such that $[f](a_1, \ldots, a_n) \in A_c$ whenever $f \in \Sigma$ and $a_1, \ldots, a_n \in A_c$, and where $[\cdot]_c$ and $isRedex_c$ are the restrictions of $[\cdot]$ and $isRedex$ to A_c, respectively. Furthermore we say that $\mathcal{A}$ *is core* whenever $\mathcal{A} = \mathcal{A}_c$.

Lemma 4.3. *Let $\mathcal{A} = \langle A, [\cdot], isRedex\rangle$ be a redex-algebra over Σ. Then for every $a \in \mathcal{A}_c$ there exists a ground term $t \in Ter(\Sigma, \varnothing)$ with $[t] = a$.* □

We now describe a syntactical construction of redex-algebras. The idea is to build Σ-algebras that 'remember' proper subterms of left-hand sides. Given this interpretation, the *isRedex* functions decide whether a redex is present. We first define some auxiliary functions.

Let $\bot$ be a fresh symbol, $\bot \notin \Sigma$, and define $\mathcal{T} = \mathit{Ter}(\Sigma \cup \{\bot\}, \varnothing)$. The function $\mathit{cut} : \mathit{Ter}(\Sigma, \mathcal{X}) \to \mathcal{T}$ is defined such that $\mathit{cut}(t)$ is the result of replacing all variables in t by $\bot$. We define $\mathit{match} : \mathcal{T} \times \mathcal{T} \to \mathit{Bool}$ such that $\mathit{match}(s,t) = \mathit{true}$ if s can be obtained from t by replacing subterms of t by $\bot$, and $\mathit{match}(s,t) = \mathit{false}$, otherwise. Let further $\mathit{merge}(s,t)$ be the 'most general common instance' of s and t, that is, $\mathit{merge} : \mathcal{T} \times \mathcal{T} \rightharpoonup \mathcal{T}$ is defined by: $\mathit{merge}(\bot, t) = t$, $\mathit{merge}(t, \bot) = t$, and $\mathit{merge}(f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)) = f(\mathit{merge}(s_1, t_1), \ldots, \mathit{merge}(s_n, t_n))$, and undefined whenever there exists a position $p \in \mathcal{P}os(s)$ such that $s(p) \in \Sigma$, $t(p) \in \Sigma$, and $s(p) \neq t(p)$. Finally we define $\mathit{shrink} : \mathcal{T} \times \mathbf{2}^{\mathcal{T}} \to \mathcal{T}$ such that $\mathit{shrink}(s, T)$ is the largest $t \in T$ (with respect to the number of symbols) such that $\mathit{match}(t, s)$ is true. Note that $\mathit{shrink}(s, T)$ is well-defined whenever T is closed under *merge* and $\bot \in T$: whenever two terms $t_1 \neq t_2$ of equal size match s then $\mathit{merge}(t_1, t_2)$ is larger and matches s.

Definition 4.4. Let R be a TRS. The *redex-algebra for* R is the core of the redex-algebra $\langle A, [\cdot], \mathit{isRedex}\rangle$, where A is the smallest set such that $\bot \in A$ and

- $t \in A$ for every proper subterm t of $\mathit{cut}(\ell)$ with ℓ a linear left-hand side of R,
- $\mathit{merge}(s,t) \in A$ whenever $s, t \in A$ and $\mathit{merge}(s,t)$ is defined.

Then $[\cdot]$ is defined by $[f](t_1, \ldots, t_n) = \mathit{shrink}(f(t_1, \ldots, t_n), A)$. And, for every $f \in \Sigma$ we define $\mathit{isRedex}_f(t_1, \ldots, t_n) = \mathit{true}$ if $f(t_1, \ldots, t_n)$ is an instance of a linear left-hand side of R, and $\mathit{isRedex}_f(t_1, \ldots, t_n) = \mathit{false}$, otherwise.

Example 4.5. Consider the term rewriting system R consisting of the rules:

$$c(c(c(x))) \to a\,, \qquad\qquad c(c(a)) \to c(c(c(c(a))))\,.$$

By Definition 4.4 we obtain $A = \{c(c(\bot)), c(\bot), \bot, c(a), a\}$. Here $[a] = a$, $[c](a) = c(a)$, $[c](c(a)) = c(c(\bot))$ and $[c](c(c(\bot))) = c(c(\bot))$. Thus the elements $\bot$ and $c(\bot)$ are not part of the core and hence not of the redex-algebra for the TRS. Here we have $\mathit{isRedex}_c(c(a)) = \mathit{isRedex}_c(c(c(\bot))) = \mathit{true}$, and *false* otherwise.

Example 4.6. We compute the domain of the redex-algebra for the TRS:

$$\begin{aligned} f(x,y) &\to a(f(c(x), y))\,, & a(f(c(c(x)), y)) &\to e\,,\\ f(x,y) &\to b(f(x, c(y)))\,, & b(f(x, c(c(y)))) &\to e\,. \end{aligned}$$

The subterms of $\mathit{cut}(\ell)$ of linear left-hand sides ℓ are: $S = \{\, \bot,\ f(c(c(\bot)), \bot),\ f(\bot, c(c(\bot))),\ c(c(\bot)),\ c(\bot)\,\}$. The closure of S under *merge* yields the domain: $A = S \cup \{f(c(c(\bot)), c(c(\bot)))\}$.

Lemma 4.7. *Let R be a TRS over Σ and $\mathcal{A}$ the redex-algebra for R. Then for all ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$ we have $t \in \mathcal{L}(\mathcal{A})$ if and only if t is a redex with respect to a left-linear rule in R.*

The proof proceeds by induction over the term structure. The 'only if'-part is crucial for soundness of our transformation, whereas the 'if'-part is needed for completeness for left-linear TRSs.

5 Minimizing Algebras

In this section we are concerned with the minimization of redex-algebras. The algorithm is similar to the minimization of deterministic tree automata, see [3]. For the set of 291 TRSs of the outermost termination competition of 2008 [10], the redex-algebras constructed according to Definition 4.4 have an average size of 4.6 elements. After an application of the minimization algorithm described here, the average size falls to 3.4, a reduction of 27%. This reduction has a polynomial influence on the number of rules of the transformed system.

Definition 5.1. Core redex-algebras $\mathcal{A}_1$, $\mathcal{A}_2$ are *equivalent* if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

For a given core redex-algebra we now construct a minimal equivalent algebra. The difference to the minimization of tree automata from [3] lies in the initial equivalence E_0. For tree automata E_0 consists of two partitions, the final and the non-final states. In our setting two states are initially equivalent if they cannot be distinguished using the *isRedex* functions. In general this can yield any number of partitions between 1 and $|A|$.

Definition 5.2. Let $\mathcal{A} = \langle A, [\cdot], \mathit{isRedex}\rangle$ be a core redex-algebra over Σ. We define equivalence relations E_i for $i \in \mathbb{N}$ on the elements of A. Initially two elements $a, b \in A$ are equivalent, $a\ E_0\ b$, if $\mathit{isRedex}_f(\boldsymbol{x}, a, \boldsymbol{y}) = \mathit{isRedex}_f(\boldsymbol{x}, b, \boldsymbol{y})$ for all symbols $f \in \Sigma$, $j \in \{1, \ldots, \sharp f\}$, $\boldsymbol{x} \in A^{j-1}$, and $\boldsymbol{y} \in A^{\sharp f - j}$. Then for $i = 0, 1, \ldots$ and $a, b \in A$ we define $a\ E_{i+1}\ b$ to be the conjunction of $a\ E_i\ b$ and $[f](\boldsymbol{x}, a, \boldsymbol{y})\ E_i\ [f](\boldsymbol{x}, b, \boldsymbol{y})$ for all $f \in \Sigma$, $j \in \{1, \ldots, \sharp f\}$, $\boldsymbol{x} \in A^{j-1}$, and $\boldsymbol{y} \in A^{\sharp f - j}$. We stop when $E_{i+1} = E_i$ for some $i \in \mathbb{N}$. Then we define $E = E_i$.

For $a \in A$ we use $\lceil a \rceil$ to denote the equivalence class of a with respect to E. The *minimized redex-algebra* is defined by $\mathcal{A}^{min} = \langle E, [\cdot]^E, \mathit{isRedex}^E\rangle$ where for every $f \in \Sigma$ we define $[f]^E : E^{\sharp f} \to E$ by $[f]^E(\lceil a_1 \rceil, \ldots, \lceil a_n \rceil) = \lceil f(a_1, \ldots, a_n) \rceil$, and $\mathit{isRedex}_f^E : E^{\sharp f} \to \mathit{Bool}$ by $\mathit{isRedex}_f^E(\lceil a_1 \rceil, \ldots, \lceil a_n \rceil) = \mathit{isRedex}_f(a_1, \ldots, a_n)$.

Lemma 5.3. *Let $\mathcal{A}$ be a core redex-algebra, then $\mathcal{A}$ is equivalent to $\mathcal{A}^{min}$.* □

Example 5.4. We consider the TRS R consisting of the following three rules:

$$f(i(a)) \to a\,, \qquad f(j(a)) \to a\,, \qquad f(a) \to a\,.$$

The redex-algebra for R is $A = \{a, i(a), j(a), \bot\}$ with the interpretation $[a] = a$, $[i](a) = i(a)$, $[j](a) = j(a)$, and the interpretation is $\bot$ in all non-listed cases; $\mathit{isRedex}_f(x) = \mathit{true}$ for all $x \neq \bot$, and *false*, otherwise.

The minimization algorithm starts with $E_0 = \{\{a, i(a), j(a)\}, \{\bot\}\}$ as initial equivalence, since $\bot$ can be distinguished from the other elements due to $[f](\bot) = \mathit{false}$. The first iteration of the algorithm yields $E_1 = \{\{a\}, \{i(a), j(a)\}, \{\bot\}\}$ as

$[i](a) = i(a)$ whereas $[i](i(a)) = [i](j(a)) = \bot$. The elements $i(a)$ and $j(a)$ are indistinguishable, and so in the second iteration we obtain $E_2 = E_1$. Thus the elements $i(a)$ and $j(a)$ are identified and we obtain an algebra that has one element less than the algebra we started with.

6 Two Versions of Dynamic Labeling

In the previous sections we have constructed and minimized redex-algebras for recognizing redex positions. For completing the transformation we still need to define how the symbols are labeled. In this section we introduce two labelings that arise naturally: minimal and maximal labeling. In minimal labeling symbols are marked with a $\star$ if they correspond to redex positions and stay unlabeled otherwise. This labeling creates a small signature and thereby results in a small number of rules of the transformed system.

In the sequel we fix R to be a TRS over Σ and $\langle \mathcal{A}, \mathit{isRedex}\rangle$ with $\mathcal{A} = \langle A, [\cdot]\rangle$ the minimized redex-algebra for R.

Definition 6.1. The *minimal labeling* for R is the semantic labeling $\langle \mathcal{A}, \pi\rangle$ defined for every $f \in \Sigma_{\mathsf{top}}$ by $\pi_f(a_1, \ldots, a_{\sharp f}) = \star$ if $\mathit{isRedex}_f(a_1, \ldots, a_{\sharp f}) = \mathit{true}$, and $\pi_f(a_1, \ldots, a_{\sharp f}) = \epsilon$, otherwise; the redex symbols are $\Sigma^{red} = \{f^\star \mid f \in \Sigma\}$.

Theorem 6.2. *Let $\langle \mathcal{A}, \pi\rangle$ and Σ^{red} be the minimal labeling for R. Then R is outermost ground terminating if $\triangle^{\pi}_{R}(\mathcal{A})$ is terminating.*

Proof. An application of Theorem 3.7 together with Lemmas 4.7 and 5.3. □

Minimal labeling is sound and efficient, but it is not complete:

Example 6.3. The following term rewriting system is outermost terminating:

$$\mathit{inf}(x) \to \mathit{cons}(x, \mathit{inf}(s(x))) \qquad \mathit{cons}(s(x), y) \to \mathit{nil} \qquad (R_2)$$

The minimized redex-algebra for R_2 is $\mathcal{A}_2 = \langle A_2, [\cdot]\rangle$ with $A_2 = \{s, \bot\}$, $[s](x) = s$, $[\mathit{inf}](x) = [\mathit{cons}](x, y) = \bot$, and $[\mathit{nil}] = \bot$. With minimal labeling we have $\pi_{\mathit{cons}}(s, x) = \star$ and $\pi_{\mathit{inf}}(x) = \star$ for all $x \in A_2$, and unmarked (ϵ) otherwise. The dynamic labeling $\triangle^{\pi}_{\mathcal{A}_2}(R_2)$ of R_2 w.r.t. $\langle \mathcal{A}_2, \pi\rangle$ consists of the following rules, the first two of which arise from the *inf*-rule, with $\alpha(x) = \bot$, and $\alpha(x) = s$ resp.:

$$\begin{aligned} \mathit{inf}^\star(x) &\to \mathit{cons}(x, \mathit{inf}^\star(s(x)))\,, \\ \mathit{inf}^\star(x) &\to \mathit{cons}^\star(x, \mathit{inf}^\star(s(x)))\,, \\ \mathit{cons}^\star(s(x), y) &\to \mathit{nil}\,. \end{aligned}$$

with $\mu(\mathit{inf}^\star) = \mu(\mathit{cons}^\star) = \varnothing$. But now $\triangle^{\pi}_{\mathcal{A}_2}(R_2)$ admits an infinite derivation:

$$\mathit{inf}^\star(x) \to \mathit{cons}(x, \mathit{inf}^\star(s(x))) \to \mathit{cons}(x, \mathit{cons}(s(x), \mathit{inf}^\star(s(s(x))))) \to \ldots\,.$$

The third term is labeled incorrectly, as the inner *cons* should be marked. The reason is that in the second step, instead of the first $\mathit{inf}^\star$-rule, the second should have been applied; however, the left-hand side $\mathit{inf}^\star(x)$ contains too little information to 'decide' what the labeling of the right-hand side should be.

This motivates the use of maximal labeling for which correct labeling is preserved under rewriting. Symbols are labeled with the interpretation of their arguments:

Definition 6.4. The *maximal labeling* for R is the semantic labeling $\langle \mathcal{A}, \pi \rangle$ defined for every $f \in \Sigma_{\mathsf{top}}$ by: $\Lambda_f = A^{\sharp f}$, $\pi_f(a_1, \ldots, a_{\sharp f}) = \langle a_1, \ldots, a_{\sharp f} \rangle$ together with the redex symbols $\Sigma^{red} = \{f^{\langle a_1, \ldots, a_{\sharp f} \rangle} \mid isRedex_f(a_1, \ldots, a_{\sharp f}) = true\}$.

Maximal labeling is sound for all TRSs, and complete for quasi-left linear TRSs. A TRS R is called *quasi left-linear* if every non-linear left-hand side of a rule in R is an instance of a linear left-hand side from R.

Theorem 6.5. *Let $\langle \mathcal{A}, \pi \rangle$ with Σ^{red} be the maximal labeling for R. Then R is outermost ground terminating if $\triangle^{\pi}_{R}(\mathcal{A})$ is terminating. Moreover, if R is quasi left-linear, the reverse direction holds as well.*

Proof. The first claim is a consequence of Theorem 3.7 and Lemmas 4.7 and 5.3. For the second claim, let R be quasi left-linear. Assume that R is outermost terminating but $\triangle^{\pi}_{\mathcal{A}}(R)$ is not terminating. We introduce types for $\triangle^{\pi}_{\mathcal{A}}(R)$ over the sorts $A \cup \{\mathsf{top}\}$. For every $f^{\lambda} \in lab(\Sigma_{\mathsf{top}})$ with $\lambda = \langle a_1, \ldots, a_{\sharp f} \rangle$ we define f^{λ} to have input sorts $\langle a_1, \ldots, a_n \rangle$ and output sort $[f](a_1, \ldots, a_n)$, except for top for which we fix output sort top. An adaptation of [7, Proposition 5.5.24] for μTRSs together with non-collapsingness of $\triangle^{\pi}_{\mathcal{A}}(R)$ yields the existence of a well-sorted infinite $\triangle^{\pi}_{R}(\mathcal{A})$ rewrite sequence τ. By Lemma 4.3 we have a ground term for every sort in A. Thus by applying a ground substitution to τ we get a well-sorted infinite ground term rewrite sequence τ'. Well-sortedness implies correct labeling: for every well-sorted term $t \in Ter(lab(\Sigma_{\mathsf{top}}), \varnothing)$ there exists a term $t' \in Ter(\Sigma_{\mathsf{top}}, \varnothing)$ such that $t = lab(t')$. Moreover, by well-sortedness the symbol top can only occur at the top of a term; without loss of generality we assume that every term in τ' has top as root. Hence to arrive at a contradiction it suffices to show that for all terms $s, t \in Ter(\Sigma, \varnothing)$ with $lab(\mathsf{top}(s)) \to_{\triangle^{\pi}_{R}(\mathcal{A})} lab(\mathsf{top}(t))$ we have $s \xrightarrow{\text{out}}_{R} t$. By construction, every rule in $\triangle^{\pi}_{R}(\mathcal{A})$ is a context extension plus labeling of a rule in R. Let $\rho : s \to_R t$ be the corresponding step. What remains to be shown is that ρ is an outermost step. Assume there would be a redex u above the rewrite position. Then by Lemma 4.7 we have $u \in \mathcal{L}(\mathcal{A}_R)$ where $\mathcal{A}_R$ is the redex-algebra for R (Definition 4.4). From Lemma 5.3 it follows that $u \in \mathcal{L}(\mathcal{A})$ as $\mathcal{A}$ is the minimization of $\mathcal{A}_R$. By definition of maximal labeling we get $root(lab(u)) \in \Sigma^{red}$. But then this symbol must be in $lab(\mathsf{top}(s))$, either above the applied $\triangle^{\pi}_{R}(\mathcal{A})$ rule or within the prepended context. Both cases yield a contradiction: the former since $\mu(root(lab(u))) = \varnothing$ would prohibit the μ-step, and the latter since we do not prepend symbols from Σ^{red}. □

Example 6.6. We revisit Example 6.3, but this time we use maximal labeling:

$$\begin{aligned} inf^{\perp}(x) &\to cons^{\perp,\perp}(x, inf^{s}(s^{\perp}(x)))\,, & inf^{s}(x) &\to cons^{s,\perp}(x, inf^{s}(s^{s}(x)))\,,\\ cons^{s,\perp}(s^{\perp}(x), y) &\to nil\,, & cons^{s,\perp}(s^{s}(x), y) &\to nil\,,\\ cons^{s,s}(s^{\perp}(x), y) &\to nil\,, & cons^{s,s}(s^{s}(x), y) &\to nil\,, \end{aligned}$$

with $\mu(inf^{\perp}) = \mu(inf^{s}) = \mu(cons^{s,\perp}) = \mu(cons^{s,s}) = \varnothing$. This μTRS is indeed terminating as opposed to the μTRS constructed in Example 6.3.

7 Discussion

For arbitrary TRSs our transformation (including the construction of C-models) is sound, and for quasi left-linear TRSs it is complete (see Theorem 6.5). The redex-algebra we construct recognizes redexes with respect to left-linear rules. As a consequence, in the μTRS $\triangle^{\pi}_{\mathcal{A}}(R)$ rewriting is forbidden only inside such redex positions. This corresponds to a weakening of the outermost rewriting strategy: contraction of redexes is allowed as long as they are not strictly contained in a redex occurrence with respect to a left-linear rule. We stress that our transformation with maximal labeling is complete for termination with respect to this rewrite strategy for all TRSs.

An open question is whether there are interesting labelings between minimal and maximal. In particular, are there more efficient complete labelings? Here efficiency is measured in the size of the signature and the number of rules of the transformed system. In Example 6.6 it would have been sufficient to label *cons* with the interpretation of the left argument, saving two symbols and two rules of the transformed system.

References

1. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving Context-Sensitive Dependency Pairs. In *LPAR*, volume 5330 of *LNCS*, pages 636–651. Springer, 2008.
2. T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236:133–178, 2000.
3. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. Available at `http://www.grappa.univ-lille3.fr/tata`, 2007.
4. J. Endrullis, J. Waldmann, and H. Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. In U. Furbach and N. Shankar, editors, *IJCAR 2006*, volume 4130 of *LNAI*, pages 574–588. Springer, 2006.
5. O. Fissore, I. Gnaedig, and H. Kirchner. Cariboo, a Termination Proof Tool for Rewriting-Based Programming Languages with Strategies, Version 1.0, 2004.
6. S. Lucas. Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming*, 1998(1), 1998.
7. E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, New York, 2002.
8. M. Raffelsieper and H. Zantema. A Transformational Approach to Prove Outermost Termination Automatically. *ENTCS*, 237:3–21, 2009.
9. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
10. Termination Competition. `http://www.termination-portal.org/`.
11. R. Thiemann. From Outermost Termination to Innermost Termination. In *SOFSEM '09*, pages 533–545, 2009.
12. H. Zantema. Termination of Term Rewriting by Semantic Labelling. *Fundamenta Informaticae*, 24:89–105, 1995.