Termination of Lazy Rewriting Revisited

Felix Schernhammer and Bernhard Gramlich

November 2007

TECHNISCHE UNIVERSITÄT WIEN

Technical Report E1852-2007-01

Theory and Logic Group, Institute of Computer Languages (E185/2)
TU Wien, Favoritenstraße 9, A-1040 Wien, Austria

Termination of Lazy Rewriting Revisited

Felix Schernhammer and Bernhard Gramlich*
Theory and Logic Group
Institute of Computer Languages
TU Wien, Austria

November 2007

Abstract

Lazy rewriting is a proper restriction of term rewriting that dynamically restricts the reduction of certain arguments of functions in order to obtain termination. In contrast to context-sensitive rewriting, reductions at such argument positions are not completely forbidden but delayed. Based on the observation that the only existing (non-trivial) approach to prove termination of such lazy rewrite systems is flawed, we develop a modified approach for transforming lazy rewrite systems into context-sensitive ones that is sound and complete with respect to termination. First experimental results with this transformation based technique are encouraging.

1 Introduction

In functional programming languages, evaluations are often carried out in a lazy fashion. This means that in the evaluation of an expression, the result of certain subexpressions is not computed until it is known that the particular result is actually needed. A very similar idea is used in lazy rewriting ([FKW00]) where the reduction of certain subterms is delayed as long as possible (cf. also [AEGL03], [HMJ76], and [Luc01]).

Termination analysis of functional programs has recently become a major subject of research in the rewriting community. Due to the similarity of lazy rewriting and the lazy evaluation strategy of functional programs, the use of lazy rewriting seems promising to find new methods for proving termination of functional programs. In particular lazy rewriting and lazy evaluation in functional programming share the idea of postponing certain evaluation steps. More precisely, arguments of functions are only evaluated if their final result is needed to compute the function.

*Email: {felixs,gramlich}@logic.at

Example 1.1 *Consider the following functional program given in term rewriting system (TRS) syntax.*

$$\begin{array}{rr} from(x) \to x : from(s(x)) & take(0, xs) \to [] \\ take(x, []) \to [] & take(s(x), y : ys) \to y : take(x, ys) \end{array}$$

Without an evaluation strategy the input term $take(s(0), from(0))$ *is non-terminating. Yet, when using a lazy evaluation strategy it is terminating and the result is* 0*. The crucial difference is that in a term of the shape* $take(\_, \_ : from(\_))$ *the from subterm may not be evaluated under lazy evaluation, because its result is not needed to evaluate any more outer function.*

However, a reduction of the problem of proving termination of functional programs to the problem of proving termination of lazy TRSs is non-trivial. The reason is that functional programming languages typically allow for features such as higher-order functions, typing and strategies other than lazy evaluation; e.g., in *Haskell* always the "first" defining equation is applied to an object term when several equations are applicable (at the same position) (this has already been pointed out in [GSST06]).

Therefore, lazy rewriting can only approximate lazy (Haskell) evaluations but clearly not simulate them in a one-to-one fashion. Yet, the described features of functional programs can be encoded or approximated through standard transformations such that finally lazy rewriting can be used for a termination analysis (cf. e.g. [GTS05]).

In this work we present a transformation from lazy TRSs into context-sensitive ones that is sound and complete w.r.t. termination and thus allows us to reduce the problem of deciding whether a lazy TRS is terminating or not to the same problem for context-sensitive TRSs, which has already been studied to some extent (cf. eg. [Luc06]). As this is the first sound and complete transformation for lazy TRSs, it enables us for the first time to investigate the use of lazy rewriting in the area of termination analysis of functional programs.

Lazy rewriting was initially introduced by [FKW00] in a graph rewriting setting (although the basic underlying idea is much older, cf. e.g. [Ing61], [FW76], [HMJ76], [HO82], [Str89], [Pv93]), [HFA+96]). However, for the termination analysis of functional programs it makes sense to consider term rewriting instead of graph rewriting, for the following reasons. First of all, using term rewriting instead of graph rewriting is more general and does not prohibit certain evaluations a priori. Moreover, the theory of ordinary term rewriting is much further developed as compared to graph rewriting. In particular, existing methods for the termination analysis of ordinary as well as context-sensitive rewrite systems can be applied where possible. Hence, in this work we will use the notion of lazy term rewriting introduced in [Luc02b].

In [Luc02b] a transformation from lazy rewrite systems into context-sensitive ones was proposed, which was supposed to preserve non-termination and conjectured to be complete w.r.t. termination. Unfortunately, a counterexample (see Example 3.2 below) proves that this transformation is unsound w.r.t. termination. In this paper we repair the transformation and prove both soundness and completeness of the new transformation w.r.t. termination.

In Section 2 of this paper we will present basic definitions and notations of lazy rewriting. In Section 3 we introduce the transformation of [Luc02b] and give a counterexample to its soundness w.r.t. termination. We then propose a modified version of the transformation which is proved to be sound and complete w.r.t. termination. Section 4 contains a discussion of the presented approach and of some experimental results.[1]

2 Preliminaries

We assume familiarity with the basic concepts and notations in term rewriting as well as context-sensitive term rewriting as provided for instance in [BN98, Luc98].

As in [FKW00] and [Luc02b] we are concerned with left-linear lazy rewrite systems in this work.

General assumption: Throughout the paper we assume that all lazy rewrite systems are *left-linear*[2] and *finite*.

Lazy rewriting operates on labelled terms. Each function and variable symbol of a term has either an *eager* label e or a *lazy* label l which we will write as superscripts. So, given a signature $\Sigma = \{f_1, ..., f_n\}$, we consider a new signature $\Sigma' = \{f_1^e, f_1^l, ..., f_n^e, f_n^l\}$. We denote by V' the set of labelled variables, so $\mathcal{T}(\Sigma', V')$ is the set of labelled terms of a labelled signature Σ'. The notation t^e (resp. t^l) for a labelled term t indicates that t has an eager (resp. lazy) root label. Following [Luc02b] we use a replacement map μ to specify for each function $f \in \Sigma$ which arguments should be evaluated eagerly. Given a replacement map μ we define the *canonical labelling* of terms as a mapping $label_\mu \colon \mathcal{T}(\Sigma, V) \to \mathcal{T}(\Sigma', V')$, where Σ' is the labelled signature and V' are the labelled variables [Luc02b]:

$$\begin{aligned} label_\mu(t) &= label_\mu^e(t) \\ label_\mu^\alpha(x) &= x^\alpha \ (\alpha \in \{e, l\}) \\ label_\mu^\alpha(f(t_1, ..., t_n)) &= f^\alpha(label_\mu^{\alpha_1}(t_1), ..., label_\mu^{\alpha_n}(t_n)) \\ &\quad \text{where } \alpha_i = e \text{ if } i \in \mu(f),\ l \text{ otherwise, and } \alpha \in \{e, l\} \end{aligned}$$

Given a labelled term t, the unlabeled term $erase(t)$ is constructed from t by omitting all labels. A position p of a term t is said to be *eager* (resp. *lazy*), if the symbol at the root of the subterm starting at position p of t has an *eager* (resp. *lazy*) label. Note that the *lazy* positions of a term are not the same as the non-replacing positions in context-sensitive rewriting. The reason is that in lazy rewriting eager positions may occur below lazy ones whereas in context-sensitive rewriting all positions which are below a non-replacing position are non-replacing.

[1] The proofs of Propositions 3.1, 3.2 and 3.3 and of the Lemmata 3.1, 3.2, 3.3, 3.4 and 3.5) can be found in the appendix.

[2] Nevertheless, for clarity we will mention this assumption in the main results.

However, rewrite steps may only be performed at so-called *active* positions. A position p is called *active* if all positions on the path from the root to p are eager. Note that, given an unlabeled term t and a replacement map μ, the active positions of $label_\mu(t)$ are exactly the replacing positions w.r.t. context-sensitive rewriting.

Definition 2.1 ([Luc02b], [FKW00]) *The* active *positions of a labelled term t (denoted $Act(t)$) are recursively defined as follows.*

- *The root position ϵ of t is* active.
- *If p is an* active *position and position $p.i$ is* eager*, then position $p.i$ is* active.

Example 2.1 *Consider a labelled term $f^e(a^l, g^e(h^l(a^e)))$. Positions $\epsilon, 2$ and $2.1.1$ are eager. Positions 1 and 2.1 are lazy and positions ϵ and 2 are active.*

Definition 2.2 ([Luc02b], [FKW00]) *Let $l \in \mathcal{T}(\Sigma, V)$ be linear, $t \in \mathcal{T}(\Sigma', V')$ be a labelled term and let p be an* active *position of t. Then l matches $t|_p$ modulo laziness if either*

- *$l \in V$ or*
- *if $l = f(l_1, ..., l_n)$ and $t|_p = f^e(t_1^\alpha, ..., t_n^\alpha)$ ($\alpha \in \{e, l\}$), then for all* eager *subterms t_i^e, l_i matches modulo laziness t_i^e.*

If t_i^l at position $p.i$ is a lazy subterm and $l_i \notin V$, then position $p.i$ is called essential.

Informally, a matching modulo laziness is a partial matching ignoring (possible) clashes at lazy positions. Positions where such clashes occur may be activated (i.e., their label may be changed from lazy to eager).

Definition 2.3 ([Luc02b]) *Let $\mathcal{R} = (\Sigma, R)$ be a (left-linear) TRS. Let t be a labelled term and let l be the left-hand side of a rule of R. If l matches modulo laziness $t|_p$, and this matching gives rise to an essential position $p.i$ ($t|_{p.i} = f^l(t_1, ..., t_n)$), then $t \xrightarrow{A} t[f^e(t_1, ..., t_n)]_{p.i}$. The relation $\xrightarrow{A}_\mathcal{R}$ is called* activation relation.

Example 2.2 *Let $l = f(a, b)$ be a linear unlabeled left-hand side of a rule, and $t = f^e(a^e, c^l)$. Then l matches t modulo laziness giving rise to the* essential *position 2 and we have $f^e(a^e, c^l) \xrightarrow{A} f^e(a^e, c^e)$. The left-hand side l does not match the labelled term $f^e(a^e, c^e)$ modulo laziness.*

Definition 2.4 ([Luc02b]) *Let l be the (linear) left-hand side of a rewrite rule and t be a labelled term. If l matches $erase(t)$, then the mapping $\sigma_{l,t} : Var(l) \to \mathcal{T}(\Sigma', V')$ is defined as follows. For all $x \in V$, with $l|_q = x$: $\sigma_{l,t}(x) = t|_q$.*

Informally, $\sigma_{l,t}$ is the matcher when matching l against $erase(t)$, where one adds the appropriate labels of t.

This substitution is modified to operate on labelled terms in the following way, yielding the mapping $\sigma_{s,t}\colon V' \to \mathcal{T}(\Sigma', V')$ [Luc02b]:

$$\sigma_{s,t}(x^e) = \begin{cases} y^e & \text{if } \sigma_{s,t}(x) = y^\alpha \in V' \\ f^e(t_1, ..., t_n) & \text{if } \sigma_{s,t}(x) = f^\alpha(t_1, ..., t_n) \end{cases}$$
$$\sigma_{s,t}(x^l) = \begin{cases} y^l & \text{if } \sigma_{s,t}(x) = y^\alpha \in V' \\ f^l(t_1, ..., t_n) & \text{if } \sigma_{s,t}(x) = f^\alpha(t_1, ..., t_n) \end{cases}$$

σ is homeomorphically extended to a mapping $\mathcal{T}(\Sigma', V') \to \mathcal{T}(\Sigma', V')$ as usual.

Definition 2.5 ([Luc02b]) *Let $\mathcal{R} = (\Sigma, R)$ be a (left-linear) TRS with replacement map μ. The* active rewrite *relation $\overset{R}{\to}_\mu\colon \mathcal{T}(\Sigma', V') \times \mathcal{T}(\Sigma', V')$ is defined as follows: Let t be a labelled term such that the left-hand side of a rewrite rule $l \to r$ matches $erase(t|_p)$ with $\sigma_{l,t|_p}$ and let $p \in Act(t)$. Then $t \overset{R}{\to}_\mu t[\sigma_{l,t|_p}(label_\mu(r))]_p$.*

Informally, the active rewrite relation $\overset{R}{\to}_\mu$ performs rewrite steps according to rewrite rules as usual, but only at active positions, and taking labels into account when constructing the contractum.

Example 2.3 *Let $l = f(x, b) \to g(x)$ be a rewrite rule and $t = f^e(h^e(a^l), b^e)$ be a labelled term. Furthermore, consider a replacement map $\mu(f) = \mu(h) = \{1\}, \mu(g) = \emptyset$. Then $\sigma_{l,t}(x) = h^e(a^l)$ and $\sigma_{l,t}(x^l)$ as appearing in the labelled right-hand side of the rule is $h^l(a^l)$. Thus we have $f^e(h^e(a^l), b^e) \overset{R}{\to}_\mu g^e(h^l(a^l))$.*

The lazy rewrite relation $\overset{LR}{\to}_\mu$ is the union of the activation relation and the active rewrite relation.

Definition 2.6 ([Luc02b]) *Let $\mathcal{R}$ be a (left-linear) TRS and let μ be a replacement map for $\mathcal{R}$. The lazy rewrite relation $\overset{LR}{\to}_\mu$ induced by $(\mathcal{R}, \mu)$ is the union of the two relations $\overset{A}{\to}$ and $\overset{R}{\to}_\mu$ ($\overset{LR}{\to}_\mu = \overset{A}{\to}_\mathcal{R} \cup \overset{R}{\to}_\mu$).*

Example 2.4 *Consider the functional program of Example 1.1.*

$$\begin{array}{rr} from(x) \to x : from(s(x)) & take(0, xs) \to [] \\ take(x, []) \to [] & take(s(x), y : ys) \to y : take(x, ys) \end{array}$$

Now we interpret it as lazy term rewriting system with a replacement map μ given by $\mu(f) = \emptyset$ for all functions f. When evaluating the canonically labelled term $take^e(s^l(0^l), from^l(0^l))$ it is obvious that the subterm $from^l(0^l)$ must be activated and evaluated before take can be computed. According to the lazy rewrite relation we have: $take^e(s^l(0^l), from^l(0^l)) \overset{LR}{\to}_\mu take^e(s^l(0^l), from^e(0^l)) \overset{LR}{\to}_\mu take^e(s^l(0^l), 0^l : from^l(s^l(0^l))) \overset{LR}{\to}_\mu 0^l : take^e(0^l, from^e(s^l(0^l))) \overset{LR}{\to}_\mu 0^l$.

Definition 2.7 *Let $\mathcal{R}$ be a TRS with a replacement map μ. Then $\mathcal{R}$ is $LR(\mu)$-terminating if there is no infinite $\overset{LR}{\to}_\mu$-sequence starting from a term t, whose labelling is canonical or more liberal (i.e., whenever $label_\mu(erase(t))|_p$ is eager, then $t|_p$ is eager as well).*

Informally, we call a labelled term t more liberal than its canonically labelled version $label_\mu(erase(t))$ if it has strictly more eager labels. The reason for considering terms with canonical or more liberal labelling in the definition of $LR(\mu)$-termination is that only such terms appear in lazy reduction sequences starting from canonically labelled terms, in which we are actually interested.

Example 2.5 *Consider a lazy TRS consisting of one rule $f(a) \to b$ with a replacement map $\mu(f) = \emptyset$. Now, when considering the canonically labelled term $f^e(a^l)$ we can reduce it to $f^e(a^e)$ according to the lazy rewrite relation. The latter term is more liberally labelled than its canonically labelled version.*

Note that $LR(\mu)$-termination and well-foundedness of $\overset{LR}{\to}_\mu$ do not coincide in general. The reason is that $LR(\mu)$-termination concerns the non-existence of lazy reduction sequences starting from canonically (or more liberally) labelled terms, while there may still be infinite reduction sequences starting from other terms. Example 2.6 shows that the two notions are indeed different.

Example 2.6 *Consider the TRS $\{g(f(a), c) \to a,\ h(x, f(b)) \to g(x, h(x, x))\}$ with a replacement map $\mu(f) = \mu(g) = \{1\}$ and $\mu(h) = \{1, 2\}$. This system is $LR(\mu)$-terminating. This can be shown with the transformation of Definition 3.5 and Theorem 3.2. However, $\overset{LR}{\to}_\mu$ is not well-founded:*

$$\begin{array}{lll} \underline{g^e(f^e(b^l), h^l(f^e(b^l), f^e(b^l)))} & \overset{LR}{\to}_\mu & g^e(f^e(b^l), h^e(f^e(b^l), f^e(b^l))) \\ & \overset{LR}{\to}_\mu & g^e(f^e(b^l), \underline{g^e(f^e(b^l), h^l(f^e(b^l), f^e(b^l)))}) \\ & \overset{LR}{\to}_\mu & \dots \end{array}$$

Note that the term b^l at position 1.1 of the starting term is lazy while it would have been eager in the canonically labelled version of the starting term. This being more lazy is the key for the existence of the infinite reduction sequence.

3 Transforming Lazy Rewrite Systems

3.1 Lucas' Transformation

We start with the definition of the transformation of [Luc02b], because it provides the basic ideas for our new one. The main idea of the transformation is to explicitly mimic activation steps of lazy rewriting through special activation rules in the transformed system which basically exchange function symbols to make them more eager (this goes back to [Ngu01]). Activations in lazy rewriting

are possible at positions which correspond to a non-variable position of the left-hand side of some rule in a partial matching. This is why in the transformation we are concerned with non-variable lazy positions of left-hand sides of rules.

The transformation is iterative. In each iteration new rules are created until a fixed point is reached. The following definition identifies for a rule $l \to r$ and a position p the positions $p.i$ which are lazy in $label_\mu(l)$. These positions are dealt with in parallel in one step of the transformation.

Definition 3.1 ([Luc02b]) *Let $l \to r$ be a rewrite rule and p a non-variable position of l, then*

$$\mathcal{I}(l,p) = \{i \in \{1,...,ar(root(l|_p))\} \mid i \notin \mu(root(l|_p)) \wedge p.i \in Pos_\Sigma(l)\}.$$

Example 3.1 *Consider a rewrite rule $l = f(a,b,x) \to r$ and a replacement map $\mu(f) = \{1\}$. Then $\mathcal{I}(l,\epsilon) = \{2\}$.*

Definition 3.2 ([Luc02b]) *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with replacement map μ and let $\mathcal{I}(l,p) = \{i_1,\ldots,i_n\} \neq \emptyset$ for some rule $l \to r \in R$ and $p \in Pos_\Sigma(l)$ where $root(l|_p) = f$. The transformed system $\mathcal{R}^\diamond = (\Sigma^\diamond, R^\diamond)$ and $\mu^\diamond$ are defined as follows:*

- *$\Sigma^\diamond = \Sigma \cup \{f_j \mid 1 \leq j \leq n\}$ where f_j are new function symbols of arity $ar(f_j) = ar(f)$*
- *$\mu^\diamond(f_j) = \mu(f) \cup \{i_j\}$ for all $1 \leq j \leq n$ and $\mu^\diamond(g) = \mu(g)$ for all $g \in \Sigma$*
- *$R^\diamond = R - \{l \to r\} \cup \{l'_{i_j} \to r \mid 1 \leq j \leq n\} \cup \{l[x]_{p.i_j} \to l'_j[x]_{p.i_j} \mid 1 \leq j \leq n\}$*

where $l'_j = l[f_j(l|_{p.1},...,l|_{p.m})]_p$ if $ar(f) = m$, and x is a fresh variable.

The transformation of Definition 3.2 is iterated until arriving at a system $\mathcal{R}^\natural = (\Sigma^\natural, R^\natural)$ and $\mu^\natural$ such that $\mathcal{I}(l,p) = \emptyset$ for every rule $l \to r \in R^\natural$ and every position $p \in Pos_\Sigma(l)$. For further motivation and examples concerning Definition 3.2 we refer to [Luc02b].

In [Luc02b] it remains unspecified how the pair l, p is selected in one step of the transformation. However, it turns out that the order in which those pairs are considered can be essential.

Example 3.2 *Consider the TRS Consider the TRS*

$$f(g(a),a) \to a \qquad b \to f(g(c),b)$$

with a replacement map $\mu(f) = \{1\}$ and $\mu(g) = \emptyset$. This system is not $LR(\mu)$-terminating:

$$b^e \overset{LR}{\to}_\mu f^e(g^e(c^l),b^l) \overset{LR}{\to}_\mu f^e(g^e(c^l),b^e) \overset{LR}{\to}_\mu f^e(g^e(c^l),f^e(g^e(c^l),b^l)) \overset{LR}{\to}_\mu \ldots$$

However, if we start the transformation with the first rule and position $p = \epsilon$, and consider position 1 of the first rule in the second step of the transformation, then we arrive at the context-sensitive system

$$\begin{array}{rlrl} f_2(g_1(a),a) &\to a & f(g_1'(a),x) &\to f_2(g(a),x)\\ f_2(g(x),a) &\to f_2(g_1(x),a) & f(g(x),y) &\to f(g_1'(x),y)\\ b &\to f(g(c),b) & & \end{array}$$

with $\mu(f) = \mu(g_1) = \mu(g_1') = \{1\}$ *and* $\mu(f_2) = \{1,2\}$.[3] *This system is* μ*-terminating (proved with AProVE [GTSK06]). The lazy reduction sequence starting from* b *cannot be mimicked anymore, because due to the two transformation steps first the argument of* g *has to be activated which prevents the activation of the* b *in the second argument of* f*.*

3.2 The New Transformation

3.2.1 Definition

In Lucas' transformation, positions that are dealt with last during the transformation must be activated first in rewrite sequences of the transformed system. This can be seen in Example 3.2 where $\mathcal{I}(f(g(a),a),\epsilon)$ is considered in the first step of the transformation but position 2 must be activated after position 1.1 (whose activation is enabled by a later transformation step considering $\mathcal{I}(f(g(a),x),1)$).

Thus, the order in which lazy positions of rules are dealt with during the transformation is the reverse order in which they may be activated in the resulting transformed system. Since we want to simulate lazy rewriting, and in lazy rewriting only outermost lazy positions may be activated in a labelled term, we consider more inner lazy positions first in our new transformation. Therefore, with the resulting context-sensitive system more outer positions may be activated only before more inner ones.

Despite considering more inner positions first in the transformation, we do not want to prioritize any (orthogonal) lazy positions. Thus, we define $\mathcal{I}(l)$ which identifies the innermost lazy positions in a term with respect to a given replacement map μ.

Definition 3.3

$$\begin{array}{lll} \mathcal{I}(l) = \{p \in Pos_\Sigma(l) & | & p \textit{ is lazy in } label_\mu(l) \wedge \\ & & \wedge\, (\nexists q \in Pos_\Sigma(l) : q \textit{ lazy in } label_\mu(l) \wedge q > p)\}. \end{array}$$

Before presenting the formal definition of our new transformation (see Definition 3.5 below), which crucially relies on Definition 3.3, we want give an informal explanation and illustration of its essential features. In our transformation we distinguish two kinds of rules that are generated. On the one hand we have *activation* rules which are characterized by the fact that in these rules the left- and right-hand side differ only at exactly one position, where in the right-hand side a different function symbol as in the left-hand side is used. While having

[3] Here and subsequently the subscripts of function symbols indicate *additional* replacing positions. So the replacement map of a symbol f_i differs from that of f in that i is replacing in f_i.

the same arity, the different function symbol in the right-hand side has exactly one more replacing position and the argument at that position is a variable (in both sides). All other rules are *active rewrite* rules.

The actual transformation proceeds in 3 stages. First, a set of initial *activation* rules is created. These rules enable the activation of one *innermost* position of a left-hand side of the original rules of the lazy TRS. As already indicated, by a rule activating position $p.i$ we mean a rule $l \to r$ where l and r differ only in the function symbol at position p and $p.i$ is replacing in r but non-replacing in l.

In the second stage one rule $l \to r$ (activating a position p) created in stage 1 (or stage 2) is replaced by a set of *activation* rules. This set contains two *activation* rules for each innermost lazy non-variable position of l. Let q be such a position (note that q is either above or orthogonal to p). Then the first of these two rules is a rule which activates q. Apart from that, both sides of this rule are identical to l (i.e. position p is still inactive). The second rule activates position p. However, in this rule position q is already active. So the second rule differs from the initial rule $l \to r$ only in that position q is replacing in both sides (cf. Example 3.3).

Example 3.3 *Assume an activation rule $f(g(x), a) \to f(g_1(x), a)$ was generated in step 1 of the transformation where $\mu(g) = \mu(f) = \emptyset$ and $\mu(g_1) = \{1\}$. The rule activates position $p = 1.1$. The left-hand side of this rule (if canonically labelled) has two innermost non-variable lazy positions $\{1, 2\}$. Thus, it will be replaced by a set of new rules. We first consider the innermost lazy position $q = 1$. So first, a rule is created which activates q where p is non-replacing (in this special case position p does not even occur in the rule). This rule is*

$$f(y, a) \to f_1(y, a)$$

Second, we create a rule which activates p while q already is active.

$$f_1(g(x), a) \to f_1(g_1(x), a)$$

These two rules illustrate that in a system obtained by this kind of transformation more outer positions (like position 1) may be activated only before *more inner ones (like 1.1). For the second innermost lazy position of the original rule we also obtain two rules:*

$$f(g(x), z) \to f_2(g(x), z) \qquad f_2(g(x), a) \to f_2(g_1(x), a)$$

We have $\mu(f_1) = \{1\}$ and $\mu(f_2) = \{2\}$. Note that all of the generated rules still have non-replacing non-variable positions in their left-hand sides. Hence, each of them must be transformed (and replaced) further in the same way as the original rule of this example was processed.

This construction ensures that with the obtained rules in every derivation q (which was considered after p in the transformation) is activated before p, which

is sound as p is a more inner (or parallel) position compared to q (since it was considered first).

The latter construction is repeated until the rules obtained do not have any lazy (non-variable) positions. We would like to point out once again that, as we consider *innermost* positions of terms in stage one and one iteration of stage two in our transformation, the outermost lazy positions of the initial rules of the lazy system are dealt with last. Hence, these are the positions which may be activated first in reduction sequences of the transformed system.

In the third stage of the transformation for each rule of the original lazy system one *active rewrite* rule is created. This rule differs from the original rule from which it was created only in the fact that the left-hand side is fully activated, i.e. it contains no lazy non-variable position. The reason is that whenever an active reduction step is performed on a lazy sequence it can be simulated by first fully activating the redex with the generated activation rules and afterwards performing the actual active step. This is done in derivations of our transformed system.

Since all new function symbols which are introduced by the transformation are substituted for function symbols of the original signature, we define the mapping *orig* from the signature of the transformed system into the original signature which identifies for each new function symbol the original one for which it was substituted.

Definition 3.4 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with replacement map μ. If in one step of the transformation $f \in \Sigma$ is replaced by a new function symbol f', then $orig(f') = f$. Furthermore, if f' is substituted for a function symbol $g \notin \Sigma$, then $orig(f') = orig(g)$. For function symbols $h \in \Sigma$, we set $orig(h) = h$ and for variables we have $orig(x) = x$.*

Definition 3.5 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with replacement map μ. The transformed system $\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R})$ with $\widetilde{\mu}$ is constructed in the following three stages.*

1 **Generation of Initial Activation Rules.** *The transformed signature $\widetilde{\Sigma} \supseteq \Sigma$ and the set $A(l)$ for every rule $l \to r \in R$ are defined as the least sets satisfying*

$$l[x]_{p.i} \to l'[x]_{p.i} \in A(l) \text{ if } p.i \in \mathcal{I}(l) \text{ and } l' = l[f_i(l|_{p.1}, ..., l|_{p.n})]_p \quad (1)$$
$$\wedge\ f_i \in \widetilde{\Sigma}$$
$$\wedge\ orig(g) = orig(h) \wedge \widetilde{\mu}(g) = \widetilde{\mu}(h) \Rightarrow g = h \text{ for all } g, h \in \widetilde{\Sigma}$$

where $\widetilde{\mu}$ is defined by $\widetilde{\mu}(f) = \mu(f)$ for all $f \in \Sigma$ and $\widetilde{\mu}(f_i) = \mu(orig(f_i)) \cup \{i\}$ if f_i was introduced in (1). Then we have $\widetilde{R} := \bigcup_{l \to r \in R} A(l)$.

2 **Saturation of Activation Rules.**

2.a **Processing one Activation Rule** *Let $\widetilde{R} = A(l_1) \cup ... \cup A(l_m)$ and let $l \to r \in A(l_j)$ for some $j \in \{1, ..., m\}$ such that $\mathcal{I}(l)$ is not empty. Then we modify the set $A(l_j)$ in the following way:*

$$A(l_j) = A(l_j) - \{l \to r\} \cup \{l[x]_{p.i} \to l'[x]_{p.i}\} \cup \{l' \to r'\}$$

$\forall p.i \in \mathcal{I}(l)$ where $l' = l[f_i(l|_{p.1}, ..., l|_{p.n})]_p$ and $r' = r[f_i'(r|_{p.1}, ..., r|_{p.n})]_p$. If there is no $g \in \widetilde{\Sigma}$ with $orig(g) = orig(f_i)$ and $\widetilde{\mu}(g) = \widetilde{\mu}(root(l|_p)) \cup \{i\}$, then $\widetilde{\Sigma} = \widetilde{\Sigma} \cup \{f_i\}$ and $\widetilde{\mu}(f_i) = \widetilde{\mu}(root(l|_p)) \cup \{i\}$, otherwise $f_i = g$. Analogously, if there is no $g \in \widetilde{\Sigma}$ with $orig(g) = orig(f_i')$ and $\widetilde{\mu}(g) = \widetilde{\mu}(root(r|_p)) \cup \{i\}$, then $\widetilde{\Sigma} = \widetilde{\Sigma} \cup \{f_i'\}$ and $\widetilde{\mu}(f_i') = \widetilde{\mu}(root(r|_p)) \cup \{i\}$, otherwise $f_i' = g$.

$$\widetilde{R} := \bigcup_{l \to r \in R} A(l)$$

2.b **Iteration** *Step 2.a is iterated until for all rules $l \to r$ of $\widetilde{R}$ we have that $\mathcal{I}(l) = \emptyset$.*

3 **Generation of Active Rewrite Rules.** *For each rule $l \to r \in R$ we add one active rewrite rule to $\widetilde{R}$. For every position $p \in Pos_\Sigma(l)$, we consider the set*

$$Symb(p, l) = \{root(r'|_p) \mid l' \to r' \in A(l) \wedge p \in Pos_\Sigma(r')\}.$$

The function symbol which is least restrictive in this set (i.e., the maximal element of $\widetilde{\mu}(f)$ w.r.t. the subset relation of all $f \in Symb(p, l)$) is unique and denoted by $maxSymb(p, l)$. We set

$$\widetilde{R} := \widetilde{R} \cup \bigcup_{l \to r \in R} l'' \to r$$

where l'' is given by $Pos(l) = Pos(l'')$, $root(l''|_p) = maxSymb(p, l)$ for all $p \in Pos_\Sigma(l)$ and $root(l''|_p) = root(l|_p)$ for all $p \in Pos_V(l)$. The signature of the transformed system is not altered in this stage.

Proposition 3.1 *Let $\mathcal{R}$ be a TRS with replacement map μ and let $\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R})$ be the transformed system with replacement map $\widetilde{\mu}$. For $f, g \in \widetilde{\Sigma}$*

$$orig(f) = orig(g) \wedge \widetilde{\mu}(f) = \widetilde{\mu}(g) \Rightarrow f = g\,.$$

Proposition 3.2 *The transformation of Definition 3.5 terminates and yields a finite transformed system for every TRS $\mathcal{R}$ and every replacement map μ.*

Proposition 3.3 *Let $\mathcal{R}$ be a TRS with replacement map μ. Let $\widetilde{\mathcal{R}}$ and $\widetilde{\mu}$ be the TRS (resp. replacement map) obtained after stages 1 and 2 of the transformation of Definition 3.5. Then the symbol $maxSymb(p, l)$ is unique for every rule $l \to r \in R$ and every $p \in Pos_\Sigma(l)$.*

Example 3.4 *Consider the TRS from Example 3.2*

$$l_1 = f(g(a), a) \to a \qquad l_2 = b \to f(g(c), b)$$

with a replacement map μ s.t. $\mu(f) = \{1\}$ and $\mu(g) = \emptyset$. In the first stage of the transformation we have $\mathcal{I}(l_1) = \{1.1, 2\}$ and the following two initial activation *rules are added (i.e., $A(l_1)$).*

$$f(g(x), a) \to f(g_1(x), a) \qquad f(g(a), x) \to f_2(g(a), x)$$

with $\widetilde{\mu}(g_1) = \{1\}$ and $\widetilde{\mu}(f_2) = \{1, 2\}$. $A(l_2) = \emptyset$, because l_2 does not contain any lazy non-variable positions. In step 2.a, the first rule of $A(l_1)$ is replaced by

$$f(g(x), y) \to f_2(g(x), y) \qquad f_2(g(x), a) \to f_2(g_1(x), a)$$

where the position p.i that was used is 2 (i.e., ϵ.2) and thus the new function symbol introduced is f_2. In the second iteration, the second rule of $A(l_1)$ is replaced by

$$f(g(x), y) \to f(g_1(x), y) \qquad f(g_1(a), x) \to f_2(g_1(a), x).$$

Here, the position p.i that was used is 1.1 and thus the new function symbol is g_1. Finally, the following active rewrite rules *are added:*

$$f_2(g_1(a), a) \to a \qquad b \to f(g(c), b).$$

For l_1 we have $Symb(\epsilon, l_1) = \{f, f_2\}$, $Symb(1, l_1) = \{g, g_1\}$ and $Symb(1.1, l_1) = Symb(2, l_1) = \{a\}$. Therefore, $maxSymb(\epsilon, l_1) = f_2$, $maxSymb(1, l_1) = g_1$ and $maxSymb(1.1, l_1) = maxSymb(2, l_1) = a$. For l_2 we have $maxSymb(\epsilon, l_2) = b$. Hence, the system $\widetilde{\mathcal{R}}$ consists of

$$\begin{array}{rr} f(g(x), y) \to f_2(g(x), y) & f_2(g(x), a) \to f_2(g_1(x), a) \\ f(g(x), y) \to f(g_1(x), y) & f(g_1(a), x) \to f_2(g_1(a), x) \\ f_2(g_1(a), a) \to a & b \to f(g(c), b) \end{array}$$

with $\widetilde{\mu}(f) = \widetilde{\mu}(g_1) = \{1\}$, $\widetilde{\mu}(f_2) = \{1, 2\}$ and $\widetilde{\mu}(g) = \emptyset$. $\widetilde{\mathcal{R}}$ is not $\widetilde{\mu}$-terminating:

$$\underline{b} \to_{\widetilde{\mu}} \underline{f(g(c), b)} \to_{\widetilde{\mu}} f_2(g(c), \underline{b}) \to_{\widetilde{\mu}} f_2(g(c), \underline{f(g(c), b)}) \to_{\widetilde{\mu}} \ldots$$

Remark 3.1 *Note that the above system could not have been derived with Lucas' transformation regardless of the order in which the positions are processed there. The reason is that when applied to this example, Lucas' transformation always enforces some order of activation of the two orthogonal lazy positions, whereas the new transformation does not.*

The rest of the paper is concerned with the proof of soundness and completeness of the transformation of Definition 3.5 w.r.t. termination. First, we will deal with the simpler case of completeness.

3.2.2 Soundness and Completeness

Theorem 3.1 *Let $\mathcal{R} = (\Sigma, R)$ be a left-linear TRS with replacement map μ, and let $\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R})$, $\widetilde{\mu}$ be the transformed system (resp. replacement map) according to Definition 3.5. If $\mathcal{R}$ is LR(μ)-terminating, then $\widetilde{\mathcal{R}}$ is $\widetilde{\mu}$-terminating.*

Proof: We will prove the result indirectly by showing that every infinite $\widetilde{\mathcal{R}}_{\widetilde{\mu}}$-derivation implies the existence of an infinite lazy $\mathcal{R}$-derivation. Assume there is an infinite $\widetilde{\mathcal{R}}_{\widetilde{\mu}}$-sequence starting from a term t. Then we construct an infinite lazy reduction sequence starting from the labelled term t' defined by

$$Pos(t) = Pos(t') \wedge \forall p \in Pos(t) : (orig(root(t|_p)) = root(erase(t'|_p)) \wedge t'|_p \text{ is eager iff } label_{\widetilde{\mu}}(t)|_p \text{ is eager}).$$

In this case we write $t' \approx t$. Note that t' is labelled canonically or more liberally because $\mu(orig(f)) \subseteq \mu(f)$ for all $f \in \widetilde{\Sigma}$. Now consider a $\widetilde{\mu}$-step $t \to_{\widetilde{\mu}} s$ and a labelled term t' with $t' \approx t$. We will prove that there is a labelled term s', such that $t' \stackrel{LR}{\to}_{\mu} s'$ and $s' \approx s$. We make a case distinction on the type of $\widetilde{\mu}$-step.

1. First assume the step is an activation step. Then there is an activation rule $l' \to l''$ in $\widetilde{R}$ which can be applied to t. This activation rule stems from a rule $l \to r \in R$, and we have that $orig(root(l'|_p)) = root(l|_p)$ for all non-variable positions p of l'. Furthermore, all variable positions of l' which are non-variable in l are lazy in $label_{\widetilde{\mu}}(l')$ and thus in t'. Hence, l matches modulo laziness t' and the same position as in t can be activated yielding s' with $s' \approx s$ (note that the active positions of t' are exactly the replacing positions of t).

2. If the step $t \to_{\widetilde{\mu}} s$ is an active rewrite step, a rule $l' \to r$ matches (a subterm of) t. This rule is the transformed version of a rule $l \to r \in R$ with $orig(root(l'|_p)) = root(l|_p)$ for all $p \in Pos(l) = Pos(l')$. Thus, l matches $erase(t')$ and the rule can be applied to t' yielding s' with $s' \approx s$. The reason is that $orig(root(s'|_p)) = root(s|_p)$ for all position of s (note that the right-hand-sides of the rules applied to t and t' are identical). Regarding the labels of s' assume that the rewrite steps were performed at a position q (in t and t'). For all positions $o \in Pos(t)$ with $o||q \vee o < q$ we have $s'|_o$ is eager if and only if $label_{\widetilde{\mu}}(s)|_o$ is eager because this has already been the case in t' and t. Furthermore, positions $q.o$ where $o \in Pos(r)$ are eager in s' if and only if they are eager in $label_{\widetilde{\mu}}(s)$ because of the canonical labelling of r inside s'. Finally, positions $q.o$ where $o \notin Pos(r)$ are eager in s' if and only if they are eager in $label_{\widetilde{\mu}}(s)$, because a proper superterm of each term $s'|_{q.o}$ occurred already in t' and thus, if an eager position of s' had not been eager in $label_{\widetilde{\mu}}(s)$ (or vice versa), then this would be a contradiction to $t' \approx t$.

■

In order to prove the soundness of our transformation, we are going to show the existence of an infinite reduction sequence in the transformed system, whenever there is an infinite lazy reduction sequence in the original system. So assume there is an infinite lazy reduction sequence in a TRS $\mathcal{R}$ with replacement map μ. The first observation is that every lazy reduction sequence naturally corresponds to a context-free (i.e. ordinary) $\to_{\mathcal{R}}$-sequence, which performs the active

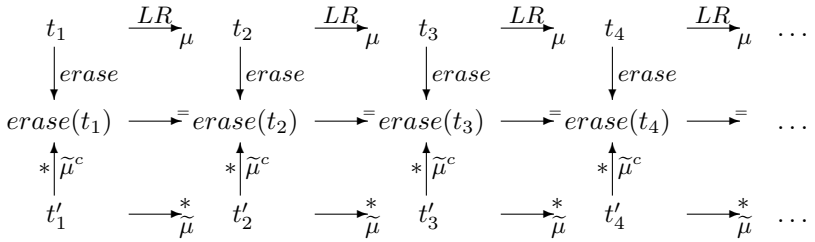

Figure 1: Relation between the various rewrite sequences occurring in the soundness proof.

rewrite steps of the lazy reduction sequence. We will construct a $\to_{\widetilde{\mu}}$-reduction sequence in the transformed system $\widetilde{\mathcal{R}}$ that corresponds to a context-free $\to_{\mathcal{R}}$-sequence, cf. Figure 1. Terms in the context-free $\to_{\mathcal{R}}$-sequence and terms in the corresponding $\widetilde{\mu}$-sequence are in a special relationship.

Definition 3.6 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, μ a replacement map and let $s, t \in \mathcal{T}(\Sigma, V)$ be two terms. Abusing notation we write $s \to^*_{\mu^c} t$ if and only if*

1. *for all positions $p \in Pos^\mu(t)$ we have $root(t|_p) = root(s|_p)$, and*
2. *for all minimal positions $q \in Pos(t) \setminus Pos^\mu(t)$ we have $s|_q \to^*_\mu s'$ and $s' \to^*_{\mu^c} t|_q$.*

The idea behind $\to^*_{\mu^c}$ is that context-free reduction steps which occur at positions that are in the replacing part of the simulating term should be simulated, thus the replacing parts of two terms s and t with $s \to^*_{\mu^c} t$ must be entirely equal. On the other hand, context-free steps that occur at positions which are forbidden in the simulating term are ignored. Yet, if the forbidden subterm in which they occur eventually gets activated, then these steps may still be simulated.

As minimal non-replacing positions in a term are always strictly below the root, the recursive description of $\to^*_{\mu^c}$ in Definition 3.6 is well-defined.

We have $s = t \Rightarrow s \to^*_{\mu^c} t$. Figure 1 illustrates the correspondence between a lazy reduction sequence, the corresponding context-free one, and the $\to_{\widetilde{\mu}}$-sequence. Note that if the lazy reduction sequence is infinite, then there are infinitely many non-empty steps in the context-free reduction sequence, as every labelled term admits only finitely many activation steps.

In the first part of the soundness proof we show the existence of a $\to_{\widetilde{\mu}}$-sequence of the shape as in Figure 1.

The next lemma provides a criterion for the existence of an activation rule in the transformed system that is able to activate a certain position in a term t over the new signature.

Lemma 3.1 *Let $(\mathcal{R} = (\Sigma, R), \mu)$ be a TRS with replacement map and let $(\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R}), \widetilde{\mu})$ be the system obtained by the transformation of Definition 3.5. Let $t \in \mathcal{T}(\widetilde{\Sigma}, V)$ be a term and $\alpha\colon l \to r \in R$ a rewrite rule of the original TRS, such that the following preconditions hold.*

1. *For all* replacing *positions p in t with $p \in Pos_\Sigma(l)$: $orig(root(t|_p)) = root(l|_p)$.*
2. *For all positions $p.i$ that are variable positions in l we have that $t|_q \in \mathcal{T}(\Sigma, V)$ for some $q \leq p$.*

Then, every position q, which is minimal non-replacing in t and non-variable in l, can be activated (i.e. we have $t \to_{\widetilde{\mu}} t'$ such that q is $\widetilde{\mu}$-replacing in t' and $orig(root(t|_p)) = orig(root(t'|_p))$ for all $p \in Pos(t)$).

The next lemma establishes the relationship between a context-free reduction sequence and a corresponding $\to_{\widetilde{\mu}}$ reduction of Figure 1.

Lemma 3.2 *Let $(\mathcal{R} = (\Sigma, R), \mu)$ be a TRS with replacement map and let $(\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R}), \widetilde{\mu})$ be the system obtained by the transformation of Definition 3.5. Let s and t be terms from $\mathcal{T}(\Sigma, V)$, such that $s \to^*_{\widetilde{\mu}^c} t$. If $t \xrightarrow{p} t^*$ (with a rule $l \to r$) and $p \in Pos^{\widetilde{\mu}}(s)$, then $s \to^+_{\widetilde{\mu}} s^*$ and $s^* \to^*_{\widetilde{\mu}^c} t^*$. Otherwise, if $t \xrightarrow{p} t^*$ and $p \notin Pos^{\widetilde{\mu}}(s)$, then $s \to^*_{\widetilde{\mu}} s^*$ and $s^* \to^*_{\widetilde{\mu}^c} t^*$.*

Unfortunately, the last lemma and the correspondence of lazy, context-free and $\to_{\widetilde{\mu}}$-reduction sequences of Figure 1 are not sufficient to prove the existence of an infinite $\to_{\widetilde{\mu}}$-sequence in the presence of an infinite lazy reduction sequence, since there may be only finitely many non-empty $\to_{\widetilde{\mu}}$-reductions in the simulating sequence.

Example 3.5 *Consider the TRS $\mathcal{R}$*

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

with a replacement map $\mu(f) = \emptyset$. The transformed system $\widetilde{\mathcal{R}}$ is

$$a \to f(a) \qquad f(x) \to f_1(x) \qquad f_1(b) \to b$$

with $\widetilde{\mu}(f) = \emptyset$, $\widetilde{\mu}(f_1) = \{1\}$. We have the following lazy reduction sequence

$$\underline{a^e} \xrightarrow{LR}_\mu f^e(\underline{a^l}) \xrightarrow{LR}_\mu f^e(\underline{a^e}) \xrightarrow{LR}_\mu \ldots$$

which corresponds to the context-free sequence

$$a \to f(a) \to f(f(a)) \to \ldots.$$

Consider a corresponding sequence in the system $\widetilde{\mathcal{R}}$,

$$a \to_{\widetilde{\mu}} f(a).$$

Then we could activate a in $f(a)$ according to rule 2 of the transformed system. However, it is a priori not clear whether such a step should be performed when trying to simulate an infinite reduction sequence. The following example illustrates the potential problems.

Example 3.6 *Consider the non-terminating TRS $\mathcal{R}$*

$$f(g(x)) \to f(g(x)) \qquad g(a) \to g(b) \qquad a \to c$$

with a replacement map $\mu(f) = \{1\}$ and $\mu(g) = \emptyset$. The transformed system $\widetilde{\mathcal{R}}$ is

$$\begin{array}{cc} f(g(x)) \to f(g(x)) & g(x) \to g_1(x) \\ g_1(a) \to g(b) & a \to c \end{array}$$

with $\widetilde{\mu}(f) = \widetilde{\mu}(g_1) = \{1\}$ and $\widetilde{\mu}(g) = \emptyset$. Consider the following context-free reduction sequence.

$$f(g(a)) \to f(g(c)) \to f(g(c)) \to \ldots$$

If we activate position 1.1 in $f(g(a))$ in the simulating $\to_{\widetilde{\mu}}$-sequence, we cannot further simulate the sequence, i.e. we get

$$f(g(a)) \to_{\widetilde{\mu}} f(g_1(a)) \to_{\widetilde{\mu}} f(g_1(c)),$$

but the term $f(g_1(c))$ is a $\to_{\widetilde{\mu}}$-normal form.

The crucial difference why the activation of a subterm is essential in Example 3.5 and unnecessary in Example 3.6 is that in the former example the activated subterm itself initiates an infinite lazy reduction sequence. This observation will be used in the second part of the soundness proof (cf. Theorem 3.2).

When constructing an infinite reduction sequence in the transformed system corresponding to an infinite lazy sequence in the soundness proof, we will identify those activations that activate a non-terminating subterms s_{inf} and simulate them by activating the corresponding subterm s'_{inf} in the simulating sequence. Afterwards, we will focus only on an infinite lazy reduction sequence initiated by s_{inf}. This way the simulated activation, i.e., the introduction of a function symbol of the new signature, is of no relevance for the further simulation as it happened outside of s'_{inf}.

With the following definition we intend to identify labelled terms in an infinite lazy reduction sequence with non-terminating proper subterms that have possibly been activated. For such terms t, the predicate $mininf(t)$ does not hold.

Definition 3.7 *Let Σ be a signature and μ be a replacement map for Σ. A labelled term t is said to be* minimal non-terminating *if it admits an infinite lazy reduction sequence and for each eager labelled proper subterm $t|_p$ of t, either $t|_p$ does not initiate an infinite lazy rewrite sequence, or position p is eager in the term $label_\mu(erase(t))$. We write $mininf(t)$ if t has this property.*

Definition 3.8 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with replacement map μ. Let t be a labelled term t and $t \overset{LR}{\rightarrow}_\mu s$ be an activation step. This activation step is called* inf-activating *(thus it is an* inf-activation step*) if and only if $mininf(t)$ but* not *$mininf(s)$.*

It is easy to see that whenever $mininf(t)$ holds for a labelled term t, there is no active position $p \in Act(t)$ which is non-active in $label_\mu(erase(t))$, such that $t|_p$ initiates an infinite lazy reduction sequence.

In the second part of the soundness proof we will show that each infinite lazy reduction sequence contains either an inf-activation step or an active rewrite step $s \overset{LR}{\rightarrow}_\mu t$ at position p such that p is μ-replacing in $erase(s)$. Furthermore, such steps result in non-empty simulations by the $\rightarrow_{\widetilde{\mu}}$-sequence.

Lemma 3.3 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with replacement map μ. Let t be a labelled term satisfying $mininf(t)$. Then we have:*

1. *If $t \overset{LR}{\rightarrow}_\mu s$ with an inf-activation step at position q_1 activating position q_2 and $q_1 < p \leq q_2$ is the maximal (w.r.t. $\leq$) eager position in s which does initiate an infinite reduction sequence s.t. $t|_p$ does not, then we have $mininf(s|_p)$.*

2. *If $t \overset{LR}{\rightarrow}_\mu s$ with any other step than in (i) (i.e., activation steps which are not inf-activating, or active rewrite steps), then $mininf(s)$.*

Lemma 3.4 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with a replacement map μ. Let $t \in \mathcal{T}(\Sigma, V)$ be an unlabeled term. If t initiates an infinite context-free reduction sequence with infinitely many root reduction steps, then a labelled term t' initiates an infinite lazy reduction sequence if $erase(t') = t$ and t' has an eager root label.*

The next lemma characterizes infinite lazy reduction sequences starting from minimal non-terminating labelled terms. It states that in such an infinite lazy reduction sequence after finitely many steps there is either an active rewrite step $s_i \overset{LR}{\rightarrow}_\mu s_{i+1}$ at some position p which is active in $label_\mu(erase(s_i))$ or there is an inf-activation step. We know from Lemma 3.2 that active rewrite steps at such positions can be simulated by a non-empty sequence in the transformed system (remember that the active rewrite steps of a lazy reduction sequence correspond to a context-free derivation). In Theorem 3.2 we will prove that the same is true for inf-activation steps.

Lemma 3.5 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS with a replacement map μ. Let t_0 be a labelled term with the property $mininf(t_0)$. Let $P : t_0 \overset{LR}{\rightarrow}_\mu t_1 \overset{LR}{\rightarrow}_\mu \ldots \overset{LR}{\rightarrow}_\mu t_n \overset{LR}{\rightarrow}_\mu \ldots$ be an infinite lazy reduction sequence starting from t_0. Then, either there is an active rewrite step $t_i \overset{LR}{\rightarrow}_\mu t_{i+1}$ at position p, where p is active in $label_\mu(erase(t_i))$, or there is an* inf-activation *step in P.*

Theorem 3.2 *Let $(\mathcal{R} = (\Sigma, R), \mu)$ be a left-linear TRS with replacement map and let $(\widetilde{\mathcal{R}} = (\widetilde{\Sigma}, \widetilde{R}), \widetilde{\mu})$ be the system obtained by the transformation of Definition 3.5. If $\widetilde{\mathcal{R}}$ is $\widetilde{\mu}$-terminating, then $\mathcal{R}$ is $LR(\mu)$-terminating.*

Proof: We will show that the existence of an infinite lazy reduction sequence $P : t_0 \overset{LR}{\to}_\mu t_1 \overset{LR}{\to}_\mu \dots$ (where t_0 is canonically or more liberally labelled) implies the existence of an infinite reduction sequence in the transformed system. The following invariant will be maintained for every labelled term t_i of an infinite reduction sequence P. Let $s_0 \to_{\widetilde{\mu}} s_1 \to_{\widetilde{\mu}} \dots$ be the simulating reduction sequence we are going to construct:
There is a s_j and a position o such that

$$s_j|_o \to^*_{\widetilde{\mu}^c} erase(t_i|_o) \wedge mininf(t_i|_o)$$

and position o is $\widetilde{\mu}$-replacing in s_j and active in t_i. Furthermore, $t_i|_o$ is at least as eager as its canonically labelled version (i.e., whenever $label_\mu(erase(t_i|_o))$ has an eager label at some position q, then the label of $t_i|_o$ is eager at that position, too). Note that the latter condition is trivially fulfilled by all terms t_i in P, and thus by all subterms, since no "deactivations" are possible in lazy rewriting and active rewrite steps only introduce canonically labelled terms.
We show that a finite initial subsequence of P implies the existence of a non-empty reduction sequence in the transformed system which preserves the invariant. As each term $t_i|_o$ itself initiates an infinite lazy reduction sequence, this suffices to show that there is an infinite reduction sequence in the transformed system.
In order to apply Lemma 3.5, we assume $mininf(t_0)$. This minimality constraint can be satisfied, as w.l.o.g. we can find a t_0 such that *each* proper subterm of t_0 with an eager label does not initiate an infinite lazy reduction sequence.
The infinite reduction sequence we are going to construct in the transformed system starts with the term $s_0 = erase(t_0)$. We have $s_0 \to^*_{\widetilde{\mu}^c} erase(t_0)$.
Lemma 3.5 states that in the lazy reduction sequence starting from t_0 there is either an active rewrite step $t_i \overset{LR}{\to}_\mu t_{i+1}$ at position p such that p is active in $label_\mu(erase(t_i))$, or there is an inf-activation step. Let $t_j \overset{LR}{\to}_\mu t_{j+1}$ be the first such step.
The goal is to show that the reduction sequence $t_0 \overset{LR}{\to}^*_\mu t_{j+1}$ can be simulated by a sequence $s_0 \to^+_{\widetilde{\mu}} s_i$ such that $s_i|_o \to^*_{\widetilde{\mu}^c} erase(t_{j+1}|_o)$ and $mininf(t_{j+1}|_o)$ holds for some position o which is active in t_{j+1} and replacing in s_i. We make a case distinction on whether the step $t_j \overset{LR}{\to}_\mu t_{j+1}$ is an active rewrite step or an *inf-activation* step.
(i). Assume the step $t_j \overset{LR}{\to}_\mu t_{j+1}$ is an active rewrite step at position p such that p is active in $label_\mu(erase(t_j))$. We have (according to Lemma 3.2) $s_0 \to^*_{\widetilde{\mu}} s_i$ and $s_i \to^*_{\widetilde{\mu}^c} erase(t_j)$. If position p is active in $label_\mu(erase(t_j))$, then p is replacing in s_i (note that $s_i \in \mathcal{T}(\Sigma, V)$). Thus, with Lemma 3.2 we have $s_0 \to^*_{\widetilde{\mu}} s_i \to^+_{\widetilde{\mu}} s_{i+1}$ and $s_{i+1} \to^*_{\widetilde{\mu}^c} erase(t_{j+1})$. Furthermore, we have $mininf(t_{j+1})$ according to Lemma 3.3.

(ii). Assume the step $t_j \overset{LR}{\to}_\mu t_{j+1}$ is an inf-activation step. Again by Lemma 3.2 we have $s_0 \to^*_{\widetilde{\mu}} s_i$ and $s_i \to^*_{\widetilde{\mu}^c} erase(t_j)$ ($s_i \in \mathcal{T}(\Sigma, V)$). The matching modulo laziness of a rule with t_j was established at a position q_{inf} which is active in $label_\mu(erase(t_j))$. The reason is that otherwise in t_j there would be a non-terminating active subterm which is non-active in $label_\mu(erase(t_j))$. Thus, $mininf(t_j)$ would not hold, which contradicts Lemma 3.3.
The fact that the activation step from t_j to t_{j+1} is inf-activating implies that there is a unique maximal active subterm $t_{j+1}|_p$ of t_{j+1} which is non-active in $label_\mu(erase(t_{j+1}))$ and initiates an infinite lazy reduction sequence. For this position p we have $p \leq q$ where q is the position that is activated in the inf-activation step: If we had $p > q$ or $p||q$, then $t_j|_p = t_{j+1}|_p$. Furthermore, as $label_\mu(erase(t_j)) = label_\mu(erase(t_{j+1}))$, this would contradict $mininf(t_j)$.
Note that, since the position q_{inf} where the matching modulo laziness was established in t_j is active in $label_\mu(erase(t_j))$, we have that $q_{inf} < p \leq q$.
In the simulating sequence we will activate position p in the term s_i. We note that p is non-replacing in s_i (as it is non-active in $label_\mu(erase(t_j))$), but it is not necessarily minimal non-replacing. Thus, in order to activate position p in s_i, we possibly need to activate positions $o < p$ in s_i first.
Let $o < p$ be the minimal non-replacing position in s_i. According to Lemma 3.1 we can activate o in s_i yielding s_i'. Note that as t_j is at least as eager as $label_\mu(erase(t_j))$, we have $orig(root(s_i|_{q_{inf}.q'})) = root(erase(t_j|_{q_{inf}.q'}))$ for every replacing position $q_{inf}.q'$ of s_i and $root(erase(t_j|_{q_{inf}.q'})) = l|_{q'}$ for some rule $l \to r$. Then, as $s_i \to^*_{\widetilde{\mu}^c} erase(t_j)$, we have $s_i' \overset{\geq o}{\to}^*_{\widetilde{\mu}} s_i''$ such that $s_i''|_o \to^*_{\widetilde{\mu}^c} erase(t_j|_o)$ according to Definition 3.6. Position o is replacing in s_i''. If there is still a non-replacing position $o' < p$ in s_i'', it is again activated and s_i'' is reduced to a term s_i''' such that $s_i'''|_{o'} \to^*_{\widetilde{\mu}^c} erase(t_j|_{o'})$. This construction is repeated until position p is replacing in a term s_i^* and we have $s_i^*|_p \to^*_{\widetilde{\mu}^c} erase(t_j|_p)$.
Note that $s_i^*|_p$ is a term over the original signature Σ, so it does not contain any function symbols introduced by the transformation. Clearly, we have that $s_i^*|_p \to^*_{\widetilde{\mu}^c} erase(t_{j+1}|_p)$, since $erase(t_j) = erase(t_{j+1})$. Finally, according to Lemma 3.3 we have $mininf(t_{j+1}|_p)$.

Now given an infinite lazy reduction sequence P starting from a labelled term t_0 and a term s_0 with $s_0 \to^*_{\widetilde{\mu}^c} erase(t_0)$, we have shown that a *finite* subsequence $t_0 \overset{LR+}{\to}_\mu t_i$ of a special shape implies the existence of a non-empty sequence $s_0 \to^+_{\widetilde{\mu}} s_j$ such that $s_j|_p \to^*_{\widetilde{\mu}^c} erase(t_i|_p)$, where p is active in t_i and replacing in s_j, $mininf(t_i|_p)$ holds and $t_i|_p$ initiates an infinite lazy reduction sequence. Thus, again the infinite lazy sequence starting at $t_i|_p$ has a finite subsequence, that can be simulated by a non-empty reduction sequence in the transformed system. By repeating this construction, we get an infinite reduction sequence in the transformed system starting at s_0. ■

4 Experiments

Whenever developing methods for proving termination of a certain kind of TRSs it is important to know how well the approach performs in practice. A series of tests were performed to answer this question for our lazy TRSs. As this is the first experimental termination analysis of lazy TRSs, neither lazy test TRSs nor benchmarks of other methods for proving lazy termination were readily available. Thus, we used the context-sensitive TRSs from the [TPDB] and interpreted these systems as lazy ones. Out of 53 lazy systems that were tested, lazy termination of 35 (with a time limit of 10 seconds) and 37 (with a time limit of 120 seconds) could be shown. The tests were performed using *AProVE* ([GTSK06]) for proving context-sensitive termination of the CSRSs obtained by our transformation. When interpreting these results one has to keep in mind that the example TRSs considered were actually supposed to be considered context-sensitive by their authors. Thus, in many cases the changes made by our transformation were only minimal. Figure 2 shows some interesting results of our experiments. Note that there were only 5 context-sensitively terminating systems which could not be shown terminating in the lazy case. As one can see in Figure 2 these systems are `Ex14_AEGL02`, `Ex1_GL02`, `Ex1_Zan97`, `Ex24_Luc06` and `Ex9_Luc06`. However, at least 4 of these systems, namely `Ex14_AEGL02`, `Ex1_Zan97`, `Ex24_Luc06` and `Ex9_Luc06` actually become non-terminating in the lazy case.[4]

5 Discussion

In the proof of Theorem 3.2 we saw that the CSRS obtained by our transformation cannot simulate lazy reduction sequences in a one-to-one fashion. When simulating infinite lazy reduction sequences, after every inf-activation step in the lazy reduction an entirely new infinite lazy sequence was considered, namely the one initiated by the activated subterm. Thus, the question arises whether we can define a transformation from lazy rewrite systems into context-sensitive ones, such that the transformed system is able to fully simulate the lazy reduction system. We conjecture that this is indeed possible ([Sch07]), but would render termination proofs more difficult. The reason is that such a transformation would need to introduce even more rules that alter the status (i.e., lazy or eager) of positions in lazy terms (to be more precise, more activation rules would be needed).

Regarding the size of the transformed system, we have that the number of rules created by our transformation is in general exponentially higher than the number of lazy non-variable subterms in left-hand sides of rules of the lazy system. However, we found that in our test series that was taken from the termination problem database ([TPDB]), this number was not too high in most of the systems. This indicates that in many practical cases the analysis of

[4] See `http://www.logic.at/people/schernhammer/lazy_rewriting/experiments.html` for a more detailed description of the experiments.

	Termination analysis succeeded...		
System	lazy (10 sec)	lazy (120 sec)	cs (120 sec)
Ex1_2_AEL03	yes	yes	yes
Ex1_2_Luc02c	yes	yes	yes
Ex14_AEGL02	no	no	yes
Ex15_Luc06	yes	yes	yes
Ex1_GL02a	no	no	yes
Ex1_GM99	no	no	no
Ex1_Zan97	no	no	yes
Ex24_GM04	no	no	no
Ex24_Luc06	no	no	yes
Ex49_GM04	yes	yes	yes
Ex4_DLMMU04	no	no	no
Ex6_15_AEL02	no	no	no
Ex6_GM04	yes	yes	yes
Ex9_Luc04	no	no	no
Ex9_Luc06	no	no	yes

Figure 2: Termination analysis of lazy TRSs

lazy termination with our approach may well be feasible and a priori not too hard. Furthermore, lazy termination analysis can greatly benefit from ongoing research in the field of context-sensitive termination.

Acknowledgements We would like to thank the anonymous reviewers of the previous workshop submission for numerous useful hints and criticisms.

References

[AEGL03] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. On-demand strategy annotations revisited. Technical Report DSIC–II/18/03, UPV, Valencia, Spain, 2003.

[BN98] F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[FKW00] W. Fokkink, J. Kamperman, and P. Walters. Lazy rewriting on eager machinery. *ACM Transactions on Programming Languages and Systems*, 22(1):45–86, 2000.

[FW76] D. P. Friedman and D. S. Wise. CONS should not evaluate its arguments. In S. Michaelson and R. Milner, eds., *Proc. 3rd ICALP*, pp. 257–284. Edinburgh University Press, 1976.

[GTSK06] J. Giesl, R. Thiemann, and P. Schneider-Kamp. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In Ulrich Furbach and Natarajan Shankar eds., *Proc. IJCAR'06,* LNCS 4130, pp. 281–286, 2006.

[GSST06] J. Giesl, S. Swiderski, R. Thiemann, and P. Schneider-Kamp. Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages In F. Pfenning, ed., *Proc. RTA'06,* LNCS 4098, pp. 297–312. Springer, 2006.

[GTS05] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and Disproving Termination of Higher-Order Functions In B. Gramlich, ed., *Proc. FROCOS'05,* LNAI 3717, pp. 216–231. Springer, 2005.

[HFA+96] P. H. Hartel, M. Feeley, M. Alt, L. Augustsson, P. Baumann, M. Beemster, E. Chailloux, C. H. Flood, W. Grieskamp, J. H. G. v. Groningen, K. Hammond, B. Hausman, M. Y. Ivory, R. E. Jones, J. Kamperman, P. Lee, X. Leroy, R. D. Lins, S. Loosemore, N. Röjemo, M. Serrano, J. Talpin, J. Thackray, S. Thomas, P. Walters, P. Weis, and P. Wentworth. Benchmarking implementations of functional languages with 'pseudoknot', a float-intensive benchmark. *Journal of Functional Programming*, 6(4):621–655, January 1996.

[HMJ76] P. Henderson and J. H. Morris Jr. A lazy evaluator. In *Conference Record of the Third ACM Symp. on Principles of Programming Languages, Atlanta, Georgia, Jan. 1976*, pp. 95–103, 1976.

[HO82] C. M. Hoffmann and M. J. O'Donnell. Programming with equations. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(1):83–112, January 1982.

[Ing61] P. Z. Ingerman. Thunks: A way of compiling procedure statements with some comments on procedure declarations. *Communications of the ACM*, 4:55–58, 1961.

[Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), 1998.

[Luc01] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In H. Sondergaard, ed., *Proc. PPDP'01*, pp. 82–93, 2001. ACM Press, New York.

[Luc02b] S. Lucas. Lazy rewriting and context-sensitive rewriting. *In M. Hanus, ed., Proc. WFLP'01,* ENTCS 64, Elsevier, 2002.

[Luc06] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(1):1782–1846, 2006.

[Ngu01] Q. H. Nguyen. Compact normalisation trace via lazy rewriting. In B. Gramlich and S. Lucas, eds., *Proc. WRS'01,* ENTCS 57, 2001.

[Pv93] M. J. Plasmeijer and M. C. J. D. van Eekelen. *Functional programming and parallel graph rewriting.* Addison-Wesley, 1993.

[Str89] R. Strandh. Classes of equational programs that compile into efficient machine code. In N. Dershowitz, ed., *Proc. RTA'89*, LNCS 355, pp. 449–461. Springer, 1989.

[Sch07] F. Schernhammer. On context-sensitive term rewriting. Master's thesis, TU Wien, 2007.

[SG07] F. Schernhammer and B. Gramlich. Termination of lazy rewriting revisited. Technical Report E1852-2007-01 (available at `http://www.logic.at/people/schernhammer/papers/`), TU Wien, 2007.

[TPDB] Termination Problem Database. Available at `http://www.lri.fr/~marche/tpdb`.

Appendix: Missing Proofs

Proof of Proposition 3.1

The result is an immediate consequence of Definition 3.5, as no new function symbol is introduced unless there is none with the same restrictions and the same original symbol. ■

Proof of Proposition 3.2

Clearly, steps 1 and 3 of Definition 3.5 terminate and produce only finitely many new rules. In step 2.a one rule is replaced by finitely many new rules. Thus, all the rules created in step 2 can be viewed as a (finitely branching) tree. To show that the iteration in step 2 of this definition terminates it is thus sufficient to show that there is no infinite path in this tree. Whenever a rule is replaced in step 2.a, the number of non-replacing non-variable positions is smaller in the new rules than in the original one. As this number is greater than or equal to zero in every term, there cannot be an infinite path. Thus, the construction of activation rules terminates for every TRS and every replacement map. ■

Proof of Proposition 3.3

Consider the set

$$Symb(p, l) = \{root(r|_p) \mid l' \to r' \in A(l) \wedge p \in Pos_\Sigma(r)\}$$

for some $l \to r \in R$ and some position $p \in Pos_\Sigma(l)$. We will show that if there are two symbols f and g from the set $Symb(p, l)$, such that $\overline{\mu}(f)$ and $\overline{\mu}(g)$ are incomparable, then either f or g is not maximal (i.e. there is a symbol $h \in Symb(p, l)$ such that $\overline{\mu}(h) \supsetneq \overline{\mu}(f)$ or $\overline{\mu}(h) \supsetneq \overline{\mu}(g)$. First, note that $\overline{\mu}(orig(f)) \subseteq \overline{\mu}(f)$ (resp. $\subseteq \overline{\mu}(g)$). W.l.o.g. we may assume $i \in \overline{\mu}(f)$ and $i \notin \overline{\mu}(g)$. Then $p.i$ is non-variable in l. Thus, for the rule $l \to r$ from $A(l)$ containing g at position p of the right-hand side, we have $\mathcal{I}(p) \neq \emptyset$. Hence, there is another rule $l' \to r' \in A(l)$ such that $r'|_p = h$ with $orig(g) = orig(h)$ and $\overline{\mu}(h) = \overline{\mu}(g) \cup \{i\}$. So g is not a maximal symbol in $Symb(p, l)$. ■

Proof of Lemma 3.1

We will prove the existence of an activation rule $l^* \to l^q$ in the transformed system which activates q in t. To this end, we will consider the non-variable positions of l and look at how they are altered during the transformation. To do this, the positions of t are partitioned into 4 classes:

1. the position q which is subject to activation,
2. the minimal positions P^l of t which are non-replacing in t and non-variable in l without q,
3. the positions P^e of t which are replacing in t and non-variable and lazy in $label_\mu(l)$, and
4. all other the positions of t.

First we consider the positions of P^l: P^l can be extended to a set P^l_α by adding all positions $p \in Pos_\Sigma(l)$ which are below some position of P^l or q such that p is lazy in $label_\mu(l)$. Now consider the transformation of the rule α. If $p \in P^l_\alpha$ is a maximal (i.e. innermost) position of P^l_α, then α is replaced by a set of initial activation rules containing

$$l[x]_p \to r\,.$$

In the next step of the transformation this rule is replaced by a set of rules containing (if p' is minimal in $P^l_\alpha \setminus \{p\}$)

$$l[x]_p[y]_{p'} \to r'\,.$$

This way, after a certain number of transformation steps we obtain an activation rule

$$\bar{l} \to \bar{r}$$

such that all positions $o \in P^l$ are variable positions in $\bar{l}$ (note that $\bar{l}$ is linear). Then position q is innermost non-variable lazy in $\bar{l}$ (note that minimal non-replacing positions of t are lazy in $label_\mu(l)$). Thus in the next step of the transformation the following rule is created:

$$\bar{l}[x]_q \to \bar{l}'[x]_q\,,$$

where $\bar{l}'$ is the same as $\bar{l}$ except that position q is $\overline{\mu}$-replacing (cf. Definition 3.5 Item 2).
Now consider the positions P^e. Let p be a maximal (i.e., innermost) position of P^e. Then the previous rule is replaced by a set of rules containing

$$l_1 \to l_2$$

where l_1 is the same as $\bar{l}[x]_q$ (resp. l_2 is the same as $\bar{l}'[x]_q$) except that position p is $\overline{\mu}$-replacing (cf. Definition 3.5 Item 2).
Again by considering all positions of P^e we deduce that there is a rule

$$l^* \to l^q$$

where l^* is the same as $\bar{l}[x]_q$ (resp. l^q is the same as $\bar{l}'[x]_q$) except that all positions of P^e are $\overline{\mu}$-replacing (cf. Definition 3.5 Item 2).
As there are no non-variable lazy positions in l^*, this rule is actually in the transformed system. Now we have the following: For all non-variable positions o of l^* we have that $orig(root(t|_o)) = orig(root(l^*|_o))$. Furthermore, a position o of l^* is $\overline{\mu}$-replacing *if and only if* it is $\overline{\mu}$-replacing in t (at this point it is crucial to demand precondition 2). The latter statement is not trivial. Positions $o \in Pos_\Sigma(l^*)$ are $\overline{\mu}$-replacing in t as well as in l^*. The remaining positions $o \in Pos_V(l^*)$ are divided into two classes. First consider the positions $o \in Pos_V(l^*)$, such that $o \notin Pos_V(l)$. These positions are non-replacing in l^* and non-replacing in t, because otherwise $l|_o$ would not have been replaced by a variable in l^* (these are positions which were in the set P^l). On the other hand consider positions

$o \in Pos_V(l^*) \wedge o \in Pos_V(l)$. Let $o = q.i$, then according to precondition 2 we have $root(l^*|_q) = root(t|_q)$. Furthermore, position q is $\overline{\mu}$-replacing in both t and l^*. Thus, $l|_o$ is $\overline{\mu}$-replacing if and only if $t|_o$ is $\overline{\mu}$-replacing. We have the following property for all function symbols $f, g \in \overline{\Sigma}$ according to Proposition 3.1:

$$orig(f) = orig(g) \wedge \overline{\mu}(f) = \overline{\mu}(g) \Rightarrow f = g.$$

Thus, we have that $root(l^*|_o) = root(t|_o)$ for all $o \in Pos_\Sigma(l^*)$, and as l^* is linear we get $t = \sigma l^*$. Thus we obtain $t \to_{\overline{\mu}} t'$ and q is active in t'. ■

Proof of Lemma 3.2

First assume that position p is in $Pos^{\overline{\mu}}(s)$ and $t|_p = \sigma l$. As $s \to^*_{\overline{\mu}^c} t$, $t|_p = \sigma l$ and s does not contain any function symbols of $\overline{\Sigma} \setminus \Sigma$, we can apply Lemma 3.1 in order to activate one minimal non-replacing position $p.q$ of s (q non-variable in l) if there is such a position. From Definition 3.6 we know that there is a $\overline{\mu}$-rewrite sequence $s \overset{\geq p.q}{\to}_{\overline{\mu}} s'$ such that $s'|_{p.q} \to^*_{\overline{\mu}^c} t|_{p.q}$ and $s'|_{p.q} \in \mathcal{T}(\Sigma, V)$. Thus, we can activate position $p.q$ in s and reduce it to s', where $p.q$ is replacing in s' and $root(s'|_{p.q}) = root(t|_{p.q})$ because $s'|_{p.q} \to^*_{\overline{\mu}^c} t|_{p.q}$. This sequence of activation and simulation of subterms at positions $p.q'$, where q' is non-replacing in (a successor of) s and non-variable in l, is repeated until we arrive at a term $\overline{s}$ ($s \to^*_{\overline{\mu}} \overline{s}$), such that all positions $p.o$ where o is non-variable in l are replacing. Furthermore, we have for all these positions that $orig(root(\overline{s}|_{p.o})) = root(l|_o)$. Thus, $\overline{s}|_p = \sigma' l'$ (where $l' \to r$ is transformed version of $l \to r$) and we have $\sigma' x \to^*_{\overline{\mu}} s^x \to^*_{\overline{\mu}^c} \sigma x$, for each variable x. The reason is that positions $p.o'$ where $o' \in Pos_V(l) (= Pos_V(l'))$ are replacing or minimal non-replacing in $\overline{s}$ because all positions of $\overline{s}$ which correspond to non-variable positions of l' are replacing. Besides, for each $p.o'$ there is a position $p' \leq p.o'$ such that $\overline{s}|_{p'} \to^*_{\overline{\mu}^c} t|_{p'}$. As $\overline{s}|_p = \sigma' l'$ and p is $\overline{\mu}$-replacing in $\overline{s}$, a rewrite step $\overline{s} \to_{\overline{\mu}} \overline{s}'$ is possible. Finally, for all replacing positions $p.q$ where $q \in Pos_V(r)$ and $r|_q = x$ (note that these positions are parallel) the sequences $\overline{s}'|_{p.q} = \sigma' x \to^*_{\overline{\mu}} s^x$ are performed yielding the term s^* with $s^* \to^*_{\overline{\mu}^c} t^*$. Note that at least one step, namely the step from $\overline{s}$ to $\overline{s}'$, was performed in the reduction sequence from s to s^*, thus we have $s \to^+_{\overline{\mu}} s^*$.

Secondly, assume that position p is non-replacing in s. Then we will show that $s^* = s$ and $s^* \to^*_{\overline{\mu}^c} t^*$. We know that

$$s \to^*_{\overline{\mu}^c} t \xrightarrow{p} t^*$$

and we want to show that $s \to^*_{\overline{\mu}} s^* \to^*_{\overline{\mu}^c} t^*$. Let $q \leq p$ be the (unique) minimal non-replacing position of s. We know that $s|_q \to^*_{\overline{\mu}} t'$ with

$$t' \to^*_{\overline{\mu}^c} t|_q \xrightarrow{p'} t^*|_q$$

where $(p = q.p')$ and in order to show that $s^* = s$ and $s^* \to^*_{\overline{\mu}^c} t^*$, it suffices to show that $t' \to^*_{\overline{\mu}} t'' \to^*_{\overline{\mu}^c} t^*|_q$. If p' is replacing in $t|_q$ (and thus in t'), the first part of the proof can be used to show $t' \to^*_{\overline{\mu}} t'' \to^*_{\overline{\mu}^c} t^*|_q$. Otherwise, we

consider the minimal non-replacing position q' of $t|_q$ (resp. t') and reduce the problem as before to the following. Given $t'|_{q'} \to^*_{\overline{\mu}} \overline{t}'$ with

$$\overline{t}' \to^*_{\overline{\mu}^c} t|_{q.q'} \stackrel{p''}{\to} t^*|_{q.q'} ,$$

we want to show that $\overline{t}' \to^*_{\overline{\mu}} \overline{t}'' \to^*_{\overline{\mu}^c} t^*|_{q.q'}$. Here, $p' = q'.p''$.
This way, the problem is reduced to subterms of t_o and t^*_o until the step from t_o to t^*_o is in the replacing part of t_o. Then part one of this proof can be applied to show that $s^* = s$ and $s^* \to^*_{\overline{\mu}^c} t^*$.

■

Proof of Lemma 3.3

ad(i): For positions $p < o \leq q_2$ it holds that if $s|_o$ initiates an infinite lazy reduction sequence, so does $t|_o$, because p was chosen to be the maximal position such that $s|_o$ initiates an infinite lazy reduction sequence where $t|_o$ does not. Furthermore, if such a position is also lazy in $label|_\mu(erase(s))$, it is lazy in $label|_\mu(erase(t))$ and this contradicts $mininf(t)$. So, let o be a position such that $o > q_2$. For each such position we have $t|_o = s|_o$. Furthermore, as $label_\mu(erase(s)) = label_\mu(erase(t))$, if $s|_o$ is eager and initiates an infinite lazy reduction sequence, then the same is true for $t|_o$, thus the statement holds.
ad(ii): First consider an active rewrite step at a position p. For all positions $q \leq p$ we have that the root labels of $s|_q$ are the same as in $t|_q$ (and are also equal in $label_\mu(erase(s))$ and $label_\mu(erase(t))$). Furthermore, if $s|_q$ does initiate an infinite reduction sequence, so does $t|_q$, as p is active in t, and thus $t|_q \stackrel{LR}{\to}_\mu s|_q$. Root labels of terms $s|_{p.o}$ where $o \in O(r)$ (if $l \to r \in R$ is the active rewrite rule which has been applied) are the same as root labels of terms $label_\mu(erase(s))|_{p.o}$ as $s = t[\sigma label_\mu(r)]_p$. So if $s|_{p.o}$ is eager, $label_\mu(erase(s))|_{p.o}$ is eager as well.
Thirdly, consider positions $p.o$ where $o \notin O(r)$. Then a proper superterm of $s|_{p.o}$ was bound to a variable in the matching σ. Thus, this superterm occurred already in t. After the rewrite step only the root label of this superterm changed. Thus, if $p.o$ were an eager position of s which is lazy in $label_\mu(erase(s))$ and $s|_{p.o}$ initiated an infinite reduction sequence, then there would be a corresponding position in t having the same properties, which contradicts the assumption that there are no such positions in t. Finally, for positions $o||p$ we have that $t|_o = s|_o$, thus, as the demanded properties hold for $t|_o$, they hold for $s|_o$, too.
If an activation step which is not an *inf-activation step* has been applied to t, then we have $mininf(s)$ according to Definition 3.8. ■

Proof of Lemma 3.4

We introduce a relation $\to^{*LR}_{\mu^c}$: $\mathcal{T}(\Sigma', V') \times \mathcal{T}(\Sigma, V)$ similar to $\to^*_{\overline{\mu}^c}$, where Σ' and V' are the labelled versions of Σ resp. V:

$$t' \to^{*LR}_{\mu^c} t \Leftrightarrow root(erase(t'|_p)) = root(t|_p) \wedge (t'|_q)^e \stackrel{LR*}{\to}_\mu t'', \text{ s.t. } t'' \to^{*LR}_{\mu^c} t|_q$$

for all $p \in Act(t')$ and all q in the set of minimal non-active positions. By $(t'|_q)^e$ we mean the term $t'|_q$ with an eager root label. Then we will show that

whenever $t' \to^{*LR}_{\mu^c} t$ and $t \to s$, there is a reduction sequence $t' \overset{LR*}{\to}_{\mu} \bar{t}$ such that $\bar{t} \to^{*LR}_{\mu^c} s$ and this sequence is not empty if the step $t \to s$ takes place at a position which is active in t'. Thus, we have

$$t' \to^{*LR}_{\mu^c} t \overset{p}{\to} s$$

and we want to show that there is a $\bar{t}$, such that $t' \overset{LR*}{\to}_{\mu} \bar{t} \to^{*LR}_{\mu^c} s$. We distinguish two cases according to the relative position of p.
First, assume that $p \notin Act(t')$ and let p' be the unique minimal non-active position in t'. As $t' \to^{*LR}_{\mu^c} t$ we know that $(t'|_{p'})^e \overset{LR*}{\to}_{\mu} t'' \to^{*LR}_{\mu^c} t|_{p'}$. Thus, we have

$$t'' \to^{*LR}_{\mu^c} t|_{p'} \overset{q}{\to} s|_{p'}$$

and as t and s differ only at positions below p', it is sufficient to show that there is a $\bar{t}'$, such that $t'' \overset{LR*}{\to}_{\mu} \bar{t}' \to^{*LR}_{\mu^c} s|_{p'}$ $(p = p'.q)$. Note that $p < q$ as we demanded that the root position of t' is eager. If q is non-active in t'', we can use the same reasoning to reduce the problem further to subterms of t and s. Eventually, we will arrive at the following situation. Given

$$\tilde{t} \to^{*LR}_{\mu^c} t|_{\tilde{p}} \overset{\tilde{q}}{\to} s|_{\tilde{p}},$$

we have to show that there is a $\tilde{t}'$, such that $\tilde{t} \overset{LR*}{\to}_{\mu} \tilde{t}' \to^{*LR}_{\mu^c} s|_{\tilde{p}}$ where $\tilde{q}$ is an active position in $\tilde{t}$. This case will be dealt with in the following.
Secondly, assume $t \overset{p}{\to} s$ with $p \in Act(t')$. $t = t[\sigma l]_p$ for the applied rule $\alpha : l \to r$. As $t' \to^{*LR}_{\mu^c} t$, l matches modulo laziness $t'|_p$. If this matching gives rise to an *essential* position q, this position can be activated in t' and as $t'|^e_q \overset{LR*}{\to}_{\mu} t''$ with $t'' \to^{*LR}_{\mu^c} t|_q$, we have $t' \overset{LR*}{\to}_{\mu} t'[t'']_q \to^{*LR}_{\mu^c} t$. This can be done for all subterms of t' which are at essential positions in the matching of l and $t'|_p$ yielding t^* $(t' \overset{LR*}{\to}_{\mu} t^*)$. Then in the matching (modulo laziness) of l and t^* at position p there are no essential positions, thus $t^* = t^*[\sigma' l]_p$ and furthermore $(\sigma' x)^e \overset{LR*}{\to}_{\mu} s^x$ with $s^x \to^{*LR}_{\mu^c} \sigma x$, since $t^* \to^{*LR}_{\mu^c} t$ and the minimal lazy positions of t^* are at or below the variable positions of l in $t^*|_p$. We can perform the step $t^* \overset{LR}{\to}_{\mu} s' = t^*[\sigma' r]_p$ (note that this step takes place in every case). Then, for all active positions $p.o$ where $o \in O_V(r)$, we can reduce $s'|_{p.o}$ to s_i with $s_i \to^{*LR}_{\mu^c} s|_{p.o}$. By performing these reductions in s' (note that the positions $p.o$ are parallel) we obtain a labelled term s'' $(s' \overset{LR*}{\to}_{\mu} s'')$ and we have $s'' \to^{*LR}_{\mu^c} s$.
Now consider a context-free reduction sequence starting from a term t containing infinitely many root reduction steps. We have $t' \to^{*LR}_{\mu^c} t$ for every term t' with $erase(t') = t$ (especially for terms with eager root label). Every step of the context-free sequence can be simulated by 0 or more steps in the lazy system. Every root reduction step is simulated by 1 or more steps. Thus, as there are infinitely many root reduction steps in the context-free system, an infinite lazy reduction sequence exists. ■

Proof of Lemma 3.5

We will show that if there is no active rewrite step $t_i \overset{LR}{\rightarrow}_\mu t_{i+1}$ at position p where p is active in $label_\mu(erase(t_i))$, then there must be an *inf-activation* step in P. The following observations are the starting point for the proof:

- Every infinite lazy reduction sequence contains infinitely many active rewrite steps. The reason is that between two active rewrite steps only finitely many activation steps are possible, as there are only finitely many positions in a labelled term which can be activated.
- Whenever a position p is active in a labelled term s, then p is active in every term s' with $s \overset{LR}{\rightarrow}_\mu s'$ (regardless of $\mathcal{R}$ and μ), unless the step from s to s' was an active rewrite step at a position $o < p$.
- We assume that no active reduction step in terms t_i at positions p are performed where p is active in $label_\mu(erase(t_i))$. Thus, we have that $Act(label_\mu(erase(t_0))) = Act(label_\mu(erase(t_i)))$ for all $i \geq 0$. Therefore, as there are infinitely many active rewrite steps, we can deduce that there are infinitely many rewrite steps below some position q which is minimal (w.r.t. $\leq$) non-active in $label_\mu(erase(t_0))$.

Let q be such a position which is a minimal (w.r.t. $\leq$) non-active position in $label_\mu(erase(t_0))$ such that in the sequence P there are infinitely many active rewrite steps at positions $p \geq q$. Clearly, if an active rewrite step takes place at a position p in a labelled term, then p must be an active position in this term. Furthermore, it is easy to see that in an activation step, which is induced by a matching modulo laziness at position p_1 activating p_2, the "distance" between p_1 and p_2 (i.e. $|o|$ where $p_2 = p_1.o$) cannot be arbitrarily high. For a given TRS this distance cannot be higher than the maximal term depth of a left-hand side of a rule. We will prove that if there is no inf-activation step, an increasingly large part of the subterms $t_i|_q$ does not change anymore during the rest of P as i gets bigger. When the constant part is big enough no activation step induced by a matching modulo laziness at a position above q can alter a subterm at position q. Then we can show that some term $t_i|_q$ initiates an infinite lazy reduction sequence, which contradicts the non-existence of inf-activation steps in P.

Towards a contradiction we assume that there is no *inf-activation* step. First, there is a subsequence of P $t_0 \overset{LR_*}{\rightarrow}_\mu t_{i_1}$, such that q is active in t_{i_1}, because there are active rewrite steps in P which take place at positions below q. Furthermore, as there is no active rewrite step in P at a position $q' < q$, q is active in t_j for all $j \geq i_1$.

- As there is no *inf-activation* step in P, we have $mininf(t_{i_1})$. Thus, $t_{i_1}|_q$ does not initiate an infinite reduction sequence. From Lemma 3.4 we can therefore deduce that there are not infinitely many active rewrite steps in P at position q. Hence, we have $t_{i_1} \overset{LR_*}{\rightarrow}_\mu t_{i_2}$ (as subsequence of P), such that $mininf(t_{i_2})$ and the subsequence of P starting with t_{i_2} contains no active rewrite steps at positions $p \leq q$.

- Positions $q.i$ of t_{i_2} ($i \in \{1, ..., ar(root(erase(t_{i_2}|_q)))\}$) cannot be "deactivated" in the remainder of the sequence P (as there is no more outer active rewrite step). Hence, all such positions $q.i$ which get activated in P (actually the subsequence of P starting at t_{i_2}), are activated after finitely many steps. I.e. $t_{i_2} \overset{LR_*}{\to}_\mu t_{i_3}$, such that $mininf(t_{i_3})$ holds and all positions p which are activated in the subsequence of P starting at t_{i_3} are either parallel to q or below some $q.i$ ($p||q \vee p > p.i$ for some $i \in \{1, ..., ar(root(erase(t_{i_3}|_q)))\}$).

The function symbol $root(erase(t_{i_3}|_q))$ is the same as $root(erase(t_j|_q))$ for all $j \geq i_3$ and all labels at positions q and $q.i$ where $i \in \{1, ..., ar(root(erase(t_{i_3}|_q)))\}$ are the same in all labelled terms t_j for all $j \geq i_3$.
We show another step of this construction:

- There are not infinitely many active rewrite steps in P at positions $q.i$ for $i \in \{1, ..., ar(root(erase(t_{i_3}|_q)))\}$ (because of Lemma 3.4 and $mininf(t_{i_3})$). Thus, we have $t_{i_3} \overset{LR_*}{\to}_\mu t_{i_4}$, such that $mininf(t_{i_4})$ and there are is no active rewrite step in the subsequence of P starting at t_{i_4} at a position $p \leq q.i$ for $i \in \{1, ..., ar(root(erase(t_{i_4}|_q)))\}$.

- We know that positions $p.i.j$, where $j \in \{1, ..., ar(root(erase(t_{i_2}|_{q.i})))\}$, cannot be deactivated in the subsequence of P starting at t_{i_4}. Thus, we have $t_{i_4} \overset{LR_*}{\to}_\mu t_{i_5}$, with $mininf(t_{i_5})$, such that all positions p which are activated in the subsequence of P starting at t_{i_5} are either parallel to q or below one $q.i.j$. ($p||q \vee p > p.i.j$ for some $p.i.j$ where $i \in \{1, ..., ar(root(erase(t_{i_5}|_q)))\}$ and $j\{1, ..., ar(root(erase(t_{i_5}|_{q.i})))\}$).

Now we have that positions $q.i.j$ where $i \in \{1, ..., ar(root(erase(t_{i_5}|_q)))\}$ and $j \in \{1, ..., ar(root(erase(t_{i_5}|_{q.i})))\}$ are not altered in P by a rewrite step at position $o \leq q$.
This construction can be continued to prove that for an arbitrary number n there is a term t_{i_n} such that $mininf(t_{i_n})$ and positions $q.o$ are not altered by rewrite steps at positions at or above q where $|o| \leq n$. For a sufficiently high n this implies that the subterm $t_{i_n}|_q$ is not altered anymore by any step at a position at or above q in the remainder of P, because it could only be altered by an activation induced by a matching modulo laziness at a position $p \leq q$. This is impossible as after n exceeds the maximal term depth of a left-hand in R it would imply that some position $p.o$ is altered.
Thus, all rewrite steps in the subsequence of P starting at t_{i_n} below position q are also possible in $t_{i_n}|_q$. As we assumed that in P there are infinitely many rewrite steps below positions q, there are still infinitely many rewrite steps below q in the subsequence of P starting at t_{i_n}. Therefore, $t_{i_n}|_q$ initiates an infinite lazy reduction sequence. As position q is lazy in $label_\mu(erase(t_{i_n}))$, this implies that we do not have $mininf(t_{i_n})$. Thus, according to Lemma 3.3 there is an *inf-activation* step in P. ■