HAL
open science

Decidable properties and optimal normalization in persistent term rewriting systems

Zurab Khasidashvili

To cite this version:

Zurab Khasidashvili. Decidable properties and optimal normalization in persistent term rewriting systems. [Research Report] RR-1837, INRIA. 1993. inria-00074835

HAL Id: inria-00074835

https://inria.hal.science/inria-00074835v1

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Decidable properties and optimal normalization in persistent term rewriting systems

Zurab KHASIDASHVILI

N° 1837
Janvier 1993

PROGRAMME 2

Calcul Symbolique,
Programmation
et Génie lociciel

Rapport de recherche

1993

Decidable Properties and Optimal Normalization in Persistent Term Rewriting Systems

Zurab Khasidashvili
INRIA-Rocquencourt, 78153, France
Zurab.Khasidashvili@inria.fr

Abstract

We introduce *Persistent* Term Rewriting Systems (PTRSs) by restricting ways of redex-creation during reductions in Orthogonal Term Rewriting Systems (OTRSs,[4], [7]). In particular, Recursive (Applicative) Program Schemes (RPSs, [3]), considered as TRSs, are persistent. We establish criteria for weak and strong normalization of terms in PTRSs and prove that they are decidable. The decidability of weak normalization implies immediately that RPSs do not have full computational power. We find a decidable, necessary, and sufficient condition for *syntactical* equivalence of PTRSs. We also prove that the *reducibility* problem is decidable in PTRSs. Finally, we design an optimal, sequential normalizing strategy for PTRSs. In particular, if t is a normalizable term in an RPS, then the number of steps in the optimal reduction of t to normal form coincides with the minimum number of redex families (in the sense of [2]) contracted in reductions of t to normal form.

Propriétés décidables et normalisation optimale dans les systèmes de réécriture persitents

Résumé

Dans cet article, nous introduisons la notion nouvelle de système de réécriture de termes *persistent*. La sous-classe des systèmes de réécriture persistents s'obtient à partir de la classe générale des systèmes de réécriture orthogonaux, en restreignant les possibilités de création de radicaux. Les schémas de programmes récursifs, par exemple, sont des systèmes persistents. Nous établissons ici une caractérisation des propriétés de normalisation faible puis forte dans le cas des systèmes persistents et prouvons que ces propriétés sont décidables. La décidabilité de la normalisation faible suffit à montrer que les systèmes persistents n'ont pas la puissance des machines de Turing. Nous présentons d'autres résultats de décidabilité dans les systèmes persistents : l'équivalence *syntaxique* de deux systèmes est décidable (et nous en donnons un critère simple), ainsi que le problème de la *réductibilité*. Enfin, nous construisons effectivement une stratégie séquentielle normalisante et optimale qui s'applique à tous les systèmes persistents. En outre, pour tout terme t normalisable, le nombre d'étapes de la réduction optimale qui calcule la forme normale de t est égal au nombre minimal de familles de radicaux (au sens de [2]) contracté par une réduction normalisante issue du terme t.

1 Introduction

The complexity of a rewrite system can be characterized by relating left-hand-sides and right-hand-sides of rewrite-rules rather than by the structure of left-hand-sides only, as specified by left-linearity, left-normality of O'Donnell [11], and strong sequentiality of Huet-Lévy [5]. We suggest classifying orthogonal TRSs (which do not have critical pairs, i.e., are left-linear and non-overlapping) depending on

(a) whether creation of a redex depends only on a rewrite rule or whether the arguments of the contracted redex and the context in which it is contracted also take part in the creation, and

(b) what kind of absorption of redexes that are outside of the contracted one is possible within arguments of created redexes.

The simplest TRSs are "development-like" systems, which we call *non-creating*, since no creation of new redexes is possible. The *Recursive Program Schemes* (RPSs) fall in the class of *persistent* systems (PTRSs), where the creation of a redex depends only on the rewrite rule; actual arguments and context do not matter. Therefore RPSs are also *inside-creating*, i.e., each created redex is inside the contractum of the contracted redex, and *non-absorbing*, i.e., new redexes cannot absorb subterms (in particular redexes) that are outside the contracted redex. In this classification, left-normal systems (where function symbols precede variables in left-hand-sides or rules), such as the λ-calculus and Combinatory Logic, fall into *non-left-absorbing* systems, where the created redex cannot absorb a subterm to the left of the contracted redex. One can define *non-absorbing* and *non-right-absorbing* systems as well. In *outside-creating* systems, each created redex contains a symbol above the contracted redex. These classes are all different in terms of the normalization and termination behaviour of terms. For example, non-creating and outside-creating systems are strongly normalizable (i.e., all reductions are terminating); for non-absorbing systems, any outermost strategy is normalizing; in non-left-absorbing systems, the leftmost-outermost strategy is normalizing, and so on. Here we study PTRSs, since they have very rich syntactical theory.

Huet-Lévy's [5] idea for constructing normalizing reductions is to find the redexes that *need* to be contracted in order to reach a normal form [5] or, according to Maranget [9], that cannot be erased by contraction of other redexes only. Our approach [6] is to determine which subterms of a term (redexes in particular) are *essential* in the sense that they contribute to the finite or infinite result of computation. Essential subterms can be characterized using a *descendant* relation, which allows the tracing of all subterms (in particular redexes after their contraction). We define the descendant of a contracted redex and all of its *pattern-subterms*, which are rooted at non-variable positions of corresponding rewrite rules, to be the contractum. Descendants of other subterms, which are in arguments of the contracted redex or outside it, are defined in the usual way. Thus a subterm is essential if it has a descendant under any reduction and is inessential otherwise.

The main feature of PTRSs is that any *free* subterm, i.e., a subterm that is not a pattern-subterm of a redex, remains free under any reduction. Thus free subterms never take part in the redex-creation. This implies that whether an argument of a redex is essential or not depends only on the rewrite rule. Thus each rule has its *essentiality indicator*, indicating which arguments are essential. Therefore a subterm is inessential iff it is in an inessential argument of a redex, i.e., there is a redex that makes the subterm inessential. Using these properties, we establish criteria for strong normalization and for weak normalization (i.e., existence of a normal form). For example, a term is normalizable iff all of its essential free subterms are normalizable. This holds iff all of its essential redexes are *semi-finite*, meaning that left-hand-sides of corresponding rules

are normalizable. Fortunately, all the needed information can be extracted from *essential rule-trees*, which are trees with rules as nodes and redexes corresponding to successors have essential occurrences in right-hand-sides of ancestors. For example, redex is semi-finite iff the corresponding essential rule-tree is finite. But to construct the essential rule-trees, we must be able to find all the essential redexes in a given term. For this we need to know the essentiality indicators of the redexes, which in turn can be found after accomplishing the construction of essential trees. However, there is a way out of this circle, since we can stop the construction of an essential rule-tree as soon as we detect a cycle in it (i.e., two occurrences of the same rule one above another), since in this case the tree is obviously infinite and the corresponding redexes are not semi-finite. The bulk of section 6 is devoted to the algorithms for constructing essential rule trees, finding all essential redexes in a given term, finding essentiality indicators of rules, and decision algorithms for weak and strong normalization, which does not need to make actual transformation of the input term.

Note that decidability of weak normalization for PTRSs implies immediately that persistent languages, for example RPSs, do not have full computational power. Indeed, the existence of an interpretor for, say λ-calculus, in a PTRS would imply decidability of weak normalization for λ-terms, which is not valid [1].

Further, we study the question of optimal normalization of terms in PTRSs. The Berry-Lévy [2] approach to the question for RPSs is to formalize a *sharing mechanism*, which was proposed by Vuillemin [12] and which leads to a graph representation of RPSs [10], by the *family* relation. They suggest to reduce all redexes in the leftmost family in one step. This approach makes it possible to avoid reductions of useless redexes and manages the duplication of redexes by counting the contractions of redexes "of the same origin" as 1 in their definition of the cost of reductions. The merit of this approach is that it is applicable also for the λ-calculus [8]. However it is still possible to have a feasible, sequential, optimal strategy for PTRSs. In particular, we show that contraction of innermost essential redexes leads to normal form in the least number of steps and that all essential redexes can be found efficiently. We also design an algorithm for finding lengths of optimal reductions.

Since weak normalization is decidable, for any PTRS R one can construct corresponding *irreducible* PTRS $IR(R)$ (with right-hand-sides of its semi-finite rules in normal form), which is *syntactically equivalent* to R, meaning that any term t has R-normal form iff it has an $IR(R)$-normal form and they coincide. Further, we show that syntactically equivalent PTRSs may differ only with right-hand-sides of non-semi-finite rules. This criterion leads to a decision algorithm for syntactical equivalence of PTRSs. Finally, we show that the *reducibility* problem is decidable in PTRSs, i.e. given two terms t and s, it is decidable whether t is reducible to s or not.

2 Properties of essential and independent subterms

We refer to [4] and [7] for introductions in TRS theory.

Notation We use R for *TRSs*; t, s, e, o for *terms*; r for *rewrite rules*; u, v, w for *redexes*; P, Q for *reductions*. The *left-hand-side* (resp. the *right-hand-side*) of a rewrite rule r is denoted by $LHS(r)$ (resp. $RHS(r)$). We write $s \in t$ if s is a *subterm* of t. A *one-etep reduction*, in which a redex u in a term t is contracted, is written as $t \xrightarrow{u} s$ or $t \rightarrow s$. We write $P : t \twoheadrightarrow s$ if P denotes a *reduction* of t to s, comprising 0 or more steps. $\emptyset_t$ or simply $\emptyset$ denotes the empty reduction of a term t, although $\emptyset$ is also used to denote the empty set. A *normal form*, nf for short, of t is denoted by $!t$.

If the last term of P coincides with the initial term of Q, then $P+Q$ denotes the *concatenation* of P and Q. $|P|$ denotes the *length*, i.e., the number of steps, of P. WN stands for *(weak) normalization* (i.e., existence of a nf) and SN stands for *strong normalization* (i.e., termination of all reductions).

Definition 2.1 (1) Let $r : t \to s$ be a rewrite rule in an OTRS R and v be an r-redex. Subterms that correspond to variables of t are the *arguments* of v and the rest is the *pattern* of v. Subterms of v rooted at the pattern are called the *pattern-subterms* of v. The *arguments*, *pattern* and *pattern-subterms* are defined analogously in the contractum of v.

(2) Let $t' \xrightarrow{u} s'$ and e be the contractum of u in s. For each argument o of u there are 0 or more arguments of e. We call them *(u-)descendants* of o. Correspondingly, subterms of o have 0 or more *descendants*. By definition, the *descendant* of each pattern-subterm of u is e. It is clear what is to meant under *descendants* of subterms that are not in u. The notion of *descendant* extends naturally to arbitrary reductions.

(3) Let $t' \xrightarrow{u} s'$. Descendants of all redexes of t' except u are also called *residuals*. A redex w in s' is called a *new* redex or a *created* redex, if it is not a residual of a redex of t'.

Notation If F is a set of redexes in t and $P : t \twoheadrightarrow s$, then F/P denotes the set of all residuals of redexes from F in s. If $F = \{u\}$, then we write u/P for $\{u\}/P$. In the following, F will also denote a *complete F-development*, where the residuals of redexes from F are contracted as long as possible. Similarly, if $u \in t$, then u will also denote the reduction $t \xrightarrow{u} s$.

Definition 2.2 Let $Q : t \twoheadrightarrow s$ and $t \xrightarrow{u} e$. Then the *residual Q/u of Q by u* is defined by induction on $|Q|$ as follows. If $Q = \emptyset_t$, then $Q/u = \emptyset_e$. If $Q = Q_1 + v$, then $Q/u = Q_1/u + v/(u/Q_1)$.

Definition 2.3 Let $P : t \twoheadrightarrow s$ and $Q : t \twoheadrightarrow e$. Then the *residual P/Q of P by Q* and the *residual Q/P of Q by P* are defined by induction on $|P|$ as follows.

(1) If $P = \emptyset_t$, then $P/Q = \emptyset_e$ and $Q/P = Q$.

(2) If $P = P_1 + u$, then $P/Q = P_1/Q + u/(Q/P_1)$ and $Q/P = (Q/P_1)/u$.

We write $P \sqcup Q$ for $P + Q/P$.

Definition 2.4 We call the reductions $P : t \twoheadrightarrow t'$ and $Q : s \twoheadrightarrow s'$ *strictly equivalent* (written $P \approx Q$) if $t = s$, $t' = s'$, and P-descendants and Q-descendants of any subterm of t are the same occurrences in t' and s'.

Theorem 2.1 (Strict Church-Rosser Theorem) Let P and Q be any coinitial reductions in an OTRS R. Then $P \sqcup Q \approx Q \sqcup P$.

Proof Similar to the ordinary case. One has only to check that reduction steps u and v in a term "strictly commute": $u + v/u \approx v + u/v$.

Definition 2.5 We call a subterm s in t *essential* (written $ES(s,t)$) if s has at least one descendant under any reduction starting from t and call it *inessential* (written $IE(s,t)$) otherwise.

Lemma 2.1 Let $s_0, \ldots, s_k \in t$ such that $IE(s_i, t)$ for all $i = 0, \ldots, k$. Then there exists a reduction P starting from t such that none of the subterms $s_0, \ldots, s_k$ have P-descendants.

Proof It follows from Theorem 2.1 that, for any reductions Q_1 and Q_2 starting from t, if $s \in t$ does not have a Q_1-descendant or a Q_2-descendant, then s also does not have descendants under $Q_1 \sqcup Q_2$ and $Q_2 \sqcup Q_1$. Thus, if P_i is a reduction starting from t such that s_i does not have P_i-descendants (P_i exists since $IE(s_i, t)$), then we can take $P = (\ldots(P_1 \sqcup P_2) \sqcup \ldots \sqcup P_n)$.

Lemma 2.2 Let $t \xrightarrow{u} t'$ and $s \in t$. Then $IE(s,t)$ iff, for every u-descendant s' of s, $IE(s',t')$.
Proof ($\Rightarrow$) Let $IE(s,t)$. Then s does not have descendants under a reduction P starting from t. By Theorem 2.1, $P+u/P \approx u+P/u$. Hence, s' does not have P/u-descendants, i.e., $IE(s',t')$. ($\Leftarrow$) If all u-descendants of s are inessential in t', then, by Lemma 2.1, there is a reduction P starting from t' in which none of them have descendants. Thus s does not have $u+P$-descendants, i.e., $IE(s,t)$.

Corollary 2.1 Let $P : t \twoheadrightarrow t'$ and $s \in t$. Then $IE(s,t)$ iff all P-descendants of s are inessential in t'. In particular, if t' is a normal form, then $ES(s,t)$ iff s has a P-descendant.

Lemma 2.3 Let $t \xrightarrow{u} t'$ and $e \in s \in t$, and let e' be a u-descendant of e. Then s has a u-descendant that contains e'.
Proof By considering all relative positions of u and e, using Definition 2.1.

Lemma 2.4 Let $e \in s \in t$ and $ES(e,t)$. Then $ES(s,t)$.
Proof By Lemma 2.3.

Lemma 2.5 Let s be a pattern-subterm of a redex $u \in t$. Then $ES(u,t)$ iff $ES(s,t)$.
Proof Let $t \xrightarrow{u} t'$. The contractum of u is the unique descendant of u, as well as of s. Hence the lemma follows from Corollary 2.1.

Notation We write $t = (t_1//e_1, ..., t_k//e_k)e$ if $e_1, ..., e_n$ are non-overlapping proper subterms in e and t is obtained from e by their replacement with $t_1, ..., t_n$ respectively.

Definition 2.6 We call a subterm s in t *free* (written $F(s,t)$) if s is not a proper pattern-subterm of a redex in t. We call $s \in t$ *independent* (written $I(s,t)$) if for any reduction P starting from t any P-descendant of s is free.

Definition 2.7 Let $t = (t_1//s_1, \ldots, t_k//s_n)s$ and let $P : s = e_0 \xrightarrow{v_0} e_1 \xrightarrow{v_1} \ldots$ Let us construct the reduction $P\|(t) : t = o_0 \xrightarrow{u_0} o_1 \xrightarrow{u_1} \ldots$ as follows. If the pattern of v_0 does not overlap with $s_1, \ldots, s_n$, then the corresponding subterm o_0' of v_0 in o_0 is a redex, and we take $u_0 = o_0'$. Otherwise $u_0 = \emptyset$. In the first case, o_1 is obtained from e_1 by replacing the descendants of $s_1, \ldots, s_n$ with $t_1, \ldots, t_n$ respectively, and in the second case by replacing the contractum of v_0 with o_0'. Hence, in o_1 we can choose the redex u_1 analogously, and so on.

Remark 2.1 The reduction $P\|(t)$ depends not only on P and t, but also on the choice of $s_1, \ldots, s_k$, but the notation will not give rise to ambiguity.

Lemma 2.6 Let $t = (t_1//s_1, \ldots, t_n//s_n)s$, the subterms $s_1, \ldots, s_k$ be independent and essential in s, and $s_{k+1}, \ldots, s_n$ be inessential in s. Further, let $P : s \twoheadrightarrow s'$ and $Q = P\|(t) : t \twoheadrightarrow t'$. Then

(1) The term t' can be obtained from s' by replacing the descendants of $s_1, \ldots s_k$ with $t_1, \ldots, t_k$ respectively, and by replacing the descendants of $s_{k+1}, \ldots, s_n$. The replaced occurrences of $t_1, \ldots, t_k$ in t' are the only descendants of $t_1, \ldots, t_k \in t$.

(2) If e and o are corresponding subterms in s and t, then e has P-descendants iff o has Q-descendants; each Q-descendant of o is in the subterm that corresponds to some P-descendant of e.

(3) If P is normalizing, then, for all $j = 1, \ldots, k$, s_j has a descendant in s', which is a nf of s_j.

(4) If P' is a reduction starting from t in which infinitely many contracted redexes are outside the descendants of $s_1, \ldots s_k$ and do not contain them as pattern-subterms, then $P\|(t)$ is infinite.
Proof (1)-(3) can be proved by induction on $|P|$, and (4) follows from Definition 2.7.

Corollary 2.2 Let $t = (t_1//s_1, \ldots, t_n//s_n)s$, where $s_1, \ldots, s_k$ are essential and independent in s, subterms $s_{k+1}, \ldots, s_n$ are inessential in s, and $t_1, \ldots, t_k$ are independent in t.

(1) If s' and t' are any corresponding subterms in s and t, then $ES(s', s)$ iff $ES(t', t)$.

(2) If s is weakly normalizable, then s_i is weakly normalizable for all $i = 1, \ldots, k$.

(3) If s is weakly normalizable and t_i is weakly normalizable for all $i = 1, \ldots, k$, then t is weakly normalizable and the normal form $!t$ of t can be obtained from $!s$ by replacing the descendants of $s_1, \ldots, s_k$, which are occurrences of $!s_1, \ldots, !s_k$, with $!t_1, \ldots, !t_k$ respectively.

(4) If s is strongly normalizable, $k = n$, and t_i is strongly normalizable for all $i = 1 \ldots n$, then t is strongly normalizable.

Lemma 2.7 Let t be a term in an OTRS, let $I(s,t)$, $P : t \twoheadrightarrow t'$ and let s have a P-descendant $s' \in t'$. Then there exists a reduction $Q : s \twoheadrightarrow s'$ such that any subterm $e \in s$ has a P-descendant in $s' \in t'$ iff e has a Q-descendant in s'.
Proof Easy induction on $|P|$.

Corollary 2.3 Let t be a term in an OTRS, let $ES(e,s)$, $ES(s,t)$, and $I(s,t)$. Then $ES(e,t)$.

3 Criteria for strong and weak normalization

Definition 3.1 (1) Let $t \xrightarrow{u} s$ in an OTRS R, and let $v \in s$ be a new redex. We call v *generated* if its pattern is in the pattern of the contractum of u.

(2) We call an OTRS R *persistent* if for each reduction step in R any created redex is generated.

Lemma 3.1 Let t be a term in a PTRS, let e be free in t, and let $t \xrightarrow{u} s$. Then any descendant of e in s is free, i.e., e is free iff it is independent.
Proof By considering all relative positions of e and u.

Definition 3.2 Let r be a rule in a PTRS. We call the sequence of numbers of essential arguments of $LHS(r)$ the *essentiality indicator* of r (written $EI(r)$). By definition, any r-redex has the same *essentiality indicator.*

It follows from Corollary 2.2.(1) that if $i \in EI(r)$, then the ith argument of any r-redex u is essential in u.

Theorem 3.1 Let t be a term in a PTRS.

(1) Each outermost redex in t is essential.

(2) If s is a free essential subterm of t and e is essential in s, then e is essential in t as well.

(3) A subterm e is inessential in t iff e is in an inessential argument of an essential redex of t.

Proof (1) Immediate.

(2) By Lemma 3.1 and Corollary 2.3.

(3) ($\Rightarrow$) Since $IE(e,t)$, e is inside an outermost, hence essential, redex. Let u be the innermost among essential redexes that contain e. It follows from Lemma 2.5 that e is in an argument s of u. Since there is no redex in s that contains e, we have $ES(e,s)$. But u is free in t and s is free in u. Thus, by (2), $IE(e,u)$ and $IE(s,u)$. ($\Leftarrow$) By Corollary 2.3.

Definition 3.3 (1) We call a redex in a PTRS R *trivial* if it is a LHS of a rewrite rule in R. We call two redexes u and v *similar* (written $u \sim v$) if they are instances of the same rule.

(2) We call a redex u *finite* if its similar trivial redex is SN, and call *infinite* otherwise. We call a rule $r \in R$ *finite* if the trivial r-redex is finite and call *infinite* otherwise.

(3) We call a redex u *semi-finite* if its similar trivial redex is WN and call *strictly infinite* otherwise. We call a rule $r \in R$ *semi-finite* if the trivial r-redex is semi-finite and call *strictly infinite* otherwise.

Theorem 3.2 Let t be a term in a PTRS. Then t is strongly normalizable iff each redex in t is finite.

Proof ($\Rightarrow$) Obviously, any redex u in t is SN. Thus, by Corollary 2.2.(4), the trivial redex similar to u is SN as well. ($\Leftarrow$) By induction on the number n of redexes in t. The case $n = 0$ is trivial. So let $n > 0$, u be an innermost redex in t, and $s = (x//u)t$. By the induction assumption, s is SN. Since arguments of u are in nf and u is a finite redex, Corollary 2.2.(4) implies that u is SN. Since $t = (u//x)s$, we have again by Corollary 2.2.(4) that t is SN.

Theorem 3.3 Let t be a term in a PTRS. Then t is weakly normalizable iff each essential redex in t is semi-finite.

Proof ($\Rightarrow$) By Corollary 2.2.(2), any essential redex u in t and all essential independent subterms of u are WN. Thus, by Corollary 2.2.(3), the trivial redex similar to u is WN as well. ($\Leftarrow$) By induction on the number n of essential redexes in t. If $n = 0$, then, by Theorem 3.1.(1), t is a nf and hence t is WN. So suppose that $n > 0$. Let u be an innermost among essential redexes in t and v be the trivial redex similar to u. By the assumption, v is WN. It follows from Theorem 3.1.(1)-(2) that essential arguments of u are in nf. Thus, by Corollary 2.2.(3), u is WN. Let $t' = (!u//u)t$. By Corollary 2.2.(1), t' contains $n - 1$ essential redexes. Thus, by the induction assumption, t' is WN. Hence t is WN.

Definition 3.4 Let R be a PTRS and $r \in R$.

(1) The *r-tree* (resp. the *essential r-tree*) is the maximal tree with rules as nodes and r as the root such that a redex corresponding to a node has an occurrence (resp. an essential occurrence) in RHSs of its ancestor node.

(2) We call R *hierarchic* (resp. *essentially hierarchic*) if there exists a function $f : R \to N$ such that if $r' \in R$ is a successor (resp. an essential successor) of an $r'' \in R$, then $f(r'') > f(r')$. (Under "r' is an essential successors of r''" we mean that an r'-redex has an essential occurrence in $RHS(r'')$). We call such an f a *hierarchic* (resp. an *essentially hierarchic*) enumeration of R.

Theorem 3.4 Let R be a PTRS, $r \in R$ and u be an r-redex. Then u is a finite redex iff the r-tree is finite.

Proof ($\Rightarrow$) If the r-tree has an infinite branch $r_0, r_1, \ldots$, then one can construct infinite reduction of the trivial r-redex $v \xrightarrow{v} t_1 \xrightarrow{v_1} t_2 \xrightarrow{v_2} \ldots$, where v_1 is an r_1-redex created by v, v_2 is an r_2-redex created by v_1, and so on. ($\Leftarrow$) By induction on the height n of the r-tree. If $n = 0$, then contractum s of the trivial r-redex v is nf. If $n > 1$, then by the induction assumption, each redex in s is finite. Hence, by Theorem 3.2, s is SN. Thus, v is also SN, i.e., u is finite.

Theorem 3.5 Let R be a PTRS. Then the following are equivalent:

(1) R is strongly normalizable;

(2) all trees in R are finite;
(3) R is hierarchic;
(4) all redexes in R are finite.

Proof ((1)⇒(2)) As we have seen in the proof of Theorem 3.4, existence of an infinite branch would imply existence of infinite reduction.

((2)⇒(3)) For any $r \in R$, let $f(r)$ be the height of the r-tree. It is easy to see that f is a hierarchic enumeration of R.

((3)⇒(4)) Let f be a hierarchic enumeration of R and $r \in R$. Let us prove by induction on $n = f(r)$ that the trivial r-redex v is SN. If $n = 0$, then the contractum s of v is a nf and so v is SN. If $n > 1$, then by the induction assumption each essential redex in s is semi-finite. Hence, by Theorem 3.2, s is SN. Thus v is SN, i.e., any r-redex is finite.

((4)⇒(1)) By Theorem 3.2.

Lemma 3.2 Let r' be a successor of r in the essential r-tree. If u is an essential r-redex in t and $t \xrightarrow{u} s$, then s contains an essential r'-redex created by u.

Proof By Definition 3.4, the contractum e of u contains an essential r'-redex v created by u. By Corollary 2.1, $ES(e, s)$. By Lemma 3.1, e is free in s. Thus by Theorem 3.1.(2), $ES(v, s)$.

Theorem 3.6 Let R be a PTRS, $r \in R$, and u be an r-redex. Then u is a semi-finite redex iff the essential r-tree is finite.

Proof (⇒) If the essential r-tree has an infinite branch $r_0, r_1, \ldots$, then by Lemma 3.2, we can construct infinite essential reduction of the trivial r-redex $v \xrightarrow{v} t_1 \xrightarrow{v_1} t_2 \xrightarrow{v_2} \ldots$, where v_1 is an essential r_1-redex created by v, v_2 is an essential r_2-redex created by v_1, and so on. (⇐) By induction on the height n of the essential r-tree. If $n = 0$, then contractum s of the trivial r-redex v is a nf. If $n > 1$, then by the induction assumption, each essential redex in s is semi-finite. Hence, by Theorem 3.3, s is WN. Thus v is WN, i.e., u is semi-finite.

Theorem 3.7 Let R be a PTRS. Then the following are equivalent:
(1) R is weakly normalizable;
(2) all essential trees in R are finite;
(3) R is essentially hierarchic;
(4) all redexes in R are semi-finite.

Proof ((1)⇒(2)) As we have seen in the proof of Theorem 3.6, existence of an infinite essential branch would imply existence of an infinite essential reduction.

((2)⇒(3)) For any $r \in R$, let $f(r)$ be the height of the essential r-tree. It is easy to see that f is an essential hierarchic enumeration of R.

((3)⇒(4)) Let f be an essential hierarchic enumeration of R and $r \in R$. Let us prove by induction on $n = f(r)$ that the trivial r-redex v is WN. If $n = 0$, then by Theorem 3.1.(1) the contractum s of v is a nf, and so v is WN. If $n > 1$, then by the induction assumption, each essential redex in s is semi-finite. Hence, by Theorem 3.3, s is WN. Thus v is WN, i.e., any r-redex is semi-finite.

((4)⇒(1)) By Theorem 3.3.

4 Optimal reductions

Lemma 4.1 Let $t \xrightarrow{u} s$, $t' \xrightarrow{u'} s'$, $u \sim u'$, $ES(u,t)$, $ES(u',t')$, and let e and e' be corresponding pattern-subterms in contractums o and o' of u and u' respectively. Then $ES(e,s)$ iff $ES(e',s')$.
Proof It follows from $ES(u,t)$, $ES(u',t')$, and Corollary 2.1 that $ES(o,s)$ and $ES(o',s')$. By Lemma 3.1 and Corollary 2.2.(1), $ES(e,o)$ iff $ES(e',o')$. But by Lemma 3.1, o and o' are free. Thus by Theorem 3.1.(2), $ES(e,s)$ iff $ES(e',s')$.

Definition 4.1 We call a redex $u \in t$ *innermost essential* if u is an essential redex that does not contain essential redexes.

Theorem 4.1 Let t be a term in a PTRS. Contraction of innermost essential redexes gives a reduction of t to normal form with the least number of steps, whenever the normal form exists.
Proof Let $P : t = t_0 \xrightarrow{u_0} \ldots \rightarrow t_n$ be a normalizing reduction starting from t and $Q : t = s_0 \xrightarrow{v_0} s_1 \xrightarrow{v_1} \ldots$ be an innermost essential reduction. It is enough to assign a number $n_i < n$ to each $i < |Q|$ in such a way that $i \neq j$ implies $n_i \neq n_j$. We show by induction on i that there is a number $n_i < n$ such that $ES(u_{n_i}, t_{n_i})$ and $u_{n_i} \sim v_i$.

(1) Let $i = 0$. Since v_0 is essential in t and t_n is a nf, it follows from Corollary 2.1 that at least one essential residual of v_0 is contracted in P. Thus we can take as n_0 a number such that u_{n_0} is an essential residual of v_0. Obviously, $v_0 \sim u_{n_0}$.

(2) Suppose that for each $i < m$, we have already defined $n_i < n$ such that $ES(u_{n_i}, t_{n_i})$ and $u_{n_i} \sim v_i$, and assume that $i = m$. Consider two possible cases: (a) v_m is a residual of a redex $v' \in t$. It follows from $ES(v_m, s_m)$ and Corollary 2.1 that $ES(v', t)$. Thus, as before, there is a number n_m such that u_{n_m} is an essential residual of v'. (b) There is a number k such that v_m is a residual of a redex $v^* \in s_k$ generated by v_{k-1}. By the induction assumption, $u_{n_{k-1}}$ is essential and $u_{n_{k-1}} \sim v_{k-1}$. Let $l = n_{k-1} + 1$ and $w^* \in t_l$ be the redex generated by $u_{n_{k-1}}$ that corresponds to v^*. By Lemma 4.1, $ES(v^*, s_k)$ iff $ES(w^*, t_l)$. But by Corollary 2.1, $ES(v_m, s)$ implies $ES(v^*, s_k)$. Thus $ES(w^*, t_l)$. Hence at least one essential residual of w^* is contracted in P, i.e., there is a number n_m such that u_{n_m} is an essential residual of w^*. Thus $u_{n_m} \sim v_m$ and $ES(u_{n_m}, t_{n_m})$. Hence n_i can be constructed for each $i < |Q|$. Since Q is innermost essential, any essential redex in s_i has at most one residual contracted in Q. Hence, $i \neq j$ implies $n_i \neq n_j$ and this completes the proof.

Definition 4.2 Let R be a PTRS. Let us define numbers $\|r\|$ and $\|u\|$ for any semi-finite rule $r \in R$ and any r-redex u by induction on the height of the essential r-tree as follows. Let $u_1, \ldots, u_m$ be all essential occurrences of redexes in $RHS(r)$. Then $\|r\| = \sum_{j=1}^{m} \|u_i\| + 1$ and $\|u\| = \|r\|$.

Theorem 4.2 Let R be a PTRS, t be a normalizable term in R, and $u_1, \ldots, u_n$ be all essential occurrences of redexes in t. Then the greatest lower bound $l(t)$ of lengths of normalizing R-reductions starting from t coincides with $\sum_{i=1}^{n} \|u_i\|$.
Proof It can be proved by induction on the height of the essential r-tree that for any r-redex u with arguments in nf the length of the rightmost innermost essential normalizing reduction starting from u is $\|u\|$. Thus the length of the rightmost innermost essential normalizing reduction starting from t is $\sum_{i=1}^{n} \|u_i\|$ and the theorem follows from Theorem 4.1.

Corollary 4.1 Let R be a non-creating OTRS, i.e., such an OTRS in which no redexes are created during reduction steps, and let t be a term in R. Then the greatest lower bound $l(t)$ of lengths of normalizing R-reductions starting from t coincides with the number of essential redexes in t. (Developments can be represented as reductions in non-creating OTRSs).

5 Syntactical equivalence

Definition 5.1 ([7]). A reduction P is called *outermost-fair* if P does not contain a term with an outermost redex which infinitely long stays an outermost and which is never contracted.

We shall use the following theorem of Klop [7].

Theorem 5.1 ([7]). Outermost-fair reductions are normalizing in OTRSs.

Definition 5.2 Let R and R' be the OTRSs with the same alphabet. We call R and R' *syntactically equivalent* (written $R \simeq R'$) if a term t has an R-nf iff it has an R'-nf and these nf's coincide.

Lemma 5.1 Let $R = \{r_i : t_i \to s_i | i \in I\}$ be an OTRS, $s_i \twoheadrightarrow e_i$ in R and let $R' = \{r'_i : t \to e_i | i \in I\}$. Further, let u be an R-redex and $u \xrightarrow{u} o$ in R'. Then $u \twoheadrightarrow o$ in R.
Proof Let u be an r_j-redex, $j \in I$. Then the reduction obtained from the R-reduction $t_j \to s_j \twoheadrightarrow e_j$ by replacing variables with corresponding arguments of u is a reduction $u \twoheadrightarrow o$ in R.

Theorem 5.2 Let $R = \{r_i : t_i \to s_i | i \in I\}$ be an OTRS, $s_i \twoheadrightarrow e_i$ in R and let $R' = \{r'_i : t \to e_i | i \in I\}$. Then $R \simeq R'$. In particular, $R \simeq IR(R)$.
Proof Let $P : t = t_0 \to t_1 \to \ldots$ be an outermost-fair reduction in R'. Then, by Lemma 5.1, there is a reduction $Q : t = s_0 \twoheadrightarrow s_{k_1} \twoheadrightarrow \ldots \twoheadrightarrow s_{k_i} \twoheadrightarrow \ldots$ in R, where $s_{k_i} = t_i$ $(i = 1, 2, \ldots)$ and $Q_i : s_{k_i} \twoheadrightarrow s_{k_{i+1}}$ is constructed as in Lemma 5.1. Let $u_m \in s_m$ and $k_i \leq m < k_{i+1}$. Since P is outermost-fair, none of outermost residuals of u_m in $s_{k_{i+1}}$ (if any) stay outermost in Q, i.e., Q is outermost-fair as well. Now the theorem follows from Theorem 5.1.

Definition 5.3 We call a semi-finite rule $r : t \to s$ in a PTRS R *irreducible* if s is in R-nf. $IR(r) : t \to !s$ is the *irreducible rule corresponding to* r. R is *irreducible* if each semi-finite rule in R is irreducible. The irreducible PTRS obtained from R by replacing all semi-finite rules with the corresponding irreducible rules is denoted by $IR(R)$.

Theorem 5.3 Let R and R' be irreducible PTRSs with the same alphabet. Then $R \simeq R'$ iff R and R' are identical except for the RHSs of strictly infinite rules.
Proof ($\Rightarrow$) Let $r : t \to s$ be a rule in R with s in R-nf. Since $R \simeq R'$, t has the same nf s in R'. By Theorem 3.1.(1), t contains an essential R'-redex u. By Theorem 3.3, u is semi-finite in R'. Since $R \simeq R'$, u is not in R-nf and it contains an R-redex v. Thus the trivial R-redex t contains an R-redex v. By the orthogonality of R, v must coincide with t, and hence $t = u = v$. Thus t is also an R'-redex. Let $t = C[e_1, \ldots, e_k]$, where $C[\,]$ is the pattern of t in R', and let $o = C[x_1, \ldots, x_k]$. Since o is an R'-redex, we can prove by analogy that o is also an R-redex. Since t is a trivial R-redex, it follows from the orthogonality of R that $o = C[x_1, \ldots, x_k] = C[e_1, \ldots, e_k] = t$. i.e., t is a trivial R'-redex. Hence, $r \in R'$. Now consider a strictly infinite R-rule $r' : t' \to s'$. One can prove by analogy that t' is the LHS of an R'-rule and s' does not have an R'-nf, since it does not have an R-nf.

($\Leftarrow$) By Theorem 4.1, if a term t has an R-nf, then an innermost essential R-reduction $P : t \xrightarrow{u_0} t_1 \xrightarrow{u_1} \ldots \xrightarrow{u_{n-1}} t_n$ leads to its R-nf t_n. By Theorem 3.3, u_i is semi-finite in R. Since the sets of semi-finite rules in R and R' coincide and the sets of R-redexes and R'-redexes are the same, P is also a normalizing R'-reduction. Thus if t has an R-nf, then it has the same R'-nf. Since the converse is also valid, we have $R \simeq R'$.

6 Decidable properties in PTRSs

Definition 6.1 Let $\alpha = r_0, ..., r_m$ and $\beta = r'_0, ..., r'_n$ be paths of a tree with rules as nodes. We call α and β *similar* if $m = n$ and r_i and r'_i are occurrences of the same rule, $i = 0, ..., m$. We call α a *cycle* if $r_0, ..., r_{m-1}$ are occurrences of pairwise different rules and r_0 and r_m are occurrences of the same rule.

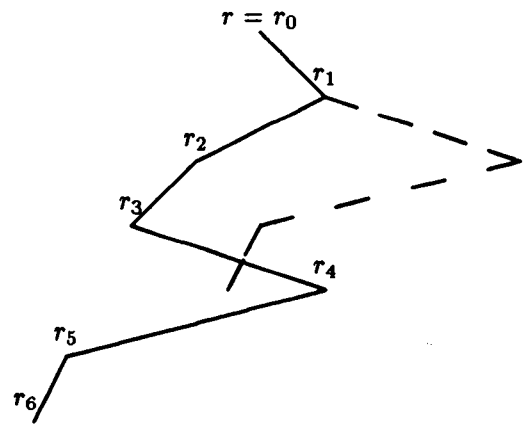
Lemma 6.1 Let R be a PTRS containing n rules, and let $r, r' \in R$. Further, let T be the r-tree (resp. the essential r-tree) and T_n be the top-subtree of T of the height n.

(1) Each rule that occurs in T occurs also in T_n.

(2) If T contains a cycle, then T_n also contains a cycle.

(3) If r'-tree is finite and occurs in T, then it occurs also in T_n.

Proof (1) Let us show by induction on k that $(a)_k$: a rule having occurrence of the depth k in T occurs also in T_n. If $k \leq n$, then $(a)_k$ is obvious. Suppose that $(a)_k$ is valid for any $k \leq m$. Let r^* be a successor of a node r'' of the depth m in T. By the induction assumption, (the rule corresponding to the occurrence) r'' has an occurrence r_d of the depth $d \leq n$ in T_n. If $d < n$, then obviously r^* has an occurrence of the depth $d + 1 \leq n$ in T_n. If $d = n$ and $\beta = r_0, ..., r_d$ is a path from the root $r_0 = r$ to r_d, then β contains at least two occurrences of some rule. Let r_i and r_j be such occurrences, $0 \leq i < j \leq n = d$. Then the path that is similar to $r_j, r_{j+1}, ..., r_d$ and starts from r_i is in the top-subtree of T of height $n - 1$. Hence r'' occurs in T_n as a node of depth less or equal to $n - 1$. Thus r^* has an occurrence in T_n, i.e., $(a)_{m+1}$ holds and (1) is proved (see the picture below, where $i = 1$, $j = 3$ and $m = n = 6$).



(2) Similarly to the proof of (1) one can show that the upmost cycle is inside T_n. Otherwise, the path from the root to the top of the cycle would contain occurrences of the same rule.

(3) Similar to (2).

Proposition 6.1 Let R be a PTRS containing n rules and $r \in R$. For each rule in the r-tree we can establish whether it is finite or not as follows.

Algorithm 6.1 Construct the top-subtree T_n of height n of the r-tree T. If the height of T_n is less than n, then each rule in T is finite. Otherwise, r is infinite and a rule r' from T is finite iff r'-tree occurs in T_n.

Proof From Theorem 3.4 and Lemma 6.1.

Proposition 6.2 Let R be a PTRS, $r \in R$, $r_1 \ldots r_k$ be all essential successors of r, and let the essentiality indicators $EI(r_1), ..., EI(r_k)$ be known. Then $EI(r)$ can be found as follows.

Algorithm 6.2 Let $r : t \rightarrow s$ and x_i be i-th argument of t. Check for each i whether x_i has an occurrence in s that is not in an inessential argument of an r_j-redex, $j = i, ..., k$. Then $i \in EI(r)$ iff x_i has such an occurrence in s.

Proof From Corollary 2.1 and Theorem 3.1.

Proposition 6.3 Let t be a term in a PTRS; let corresponding rules of essential redexes of t be semi-finite and their essentiality indicators and corresponding irreducible rules be known. Then we can (a) find all essential redexes in t and (b) find the normal form $!t$ of t using the innermost essential strategy as follows.

Algorithm 6.3 (a) Mark all outermost redexes in t; in essential arguments of marked redexes mark all outermost redexes and so on, as long as possible. The marked redexes are all the essential redexes in t.

(b) Choose an innermost marked redex in t and reduce it using appropriate irreducible rule. Reduce obtained term in the same way, and so on, as long as possible.

Proof (a) By Theorem 3.1. (b) By Corollary 2.1 and Theorems 3.3, 3.6 and 4.1.

Proposition 6.4 Let $r : t \rightarrow s$ be a rule in a PTRS, let $r_1 \ldots r_k$ be all its essential successors, and let $IR(r_1), \ldots, IR(r_k)$ and $EI(r_1), ..., EI(r_k)$ be known. Then $IR(r)$, $EI(r)$, and $EI(IR(r))$ can be found as follows.

Algorithm 6.4 Using Algorithm 6.3, find $!s$. $IR(r) : t \rightarrow !s$ is the irreducible r-rule. If x_i is the i-th argument of t, then $i \in EI(r) = EI(IR(r))$ iff x_i occurs in $!s$.

Proof $EI(r) = EI(IR(r))$ follows from Corollary 2.1. The rest follows from Proposition 6.3.

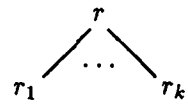
Proposition 6.5 Let R be a PTRS containing n rules and $r \in R$. We can establish whether r is finite and if so, for each rule in the r-tree find its essentiality indicator and corresponding irreducible rule as follows.

Algorithm 6.5 Construct the top-subtree T_n of height n of the r-tree. If it contains a branch of length n, then r is infinite and the algorithm stops. Otherwise, choose a leaf r_1 in T_n, find $IR(r_1)$ and $EI(r_1)$ using Algorithm 6.4, and mark r_1. Choose an unmarked leaf or an unmarked node all successors of which are marked and use Algorithm 6.4, and so on, as long as possible. The algorithm stops after finding $IR(r)$ and $EI(r)$.

Proof From Propositions 6.1 and 6.4.

Definition 6.2 (1) Let t be a term in a PTRS R, $F_R = \{r', ..., r^*\}$ be a set of R-rules, and let $EI(r'), \ldots, EI(r^*)$ be known. Let us define *F_R-essential redexes* in t as follows. Mark all outermost redexes in t; in essential arguments of marked $r'-, \ldots, r^*-$redexes mark all outermost redexes and so on, as long as possible. Then the marked redexes are the *F_R-essential redexes* in t.

(2) Let $r \in R$. A *local r-figure* is a tree of the form



(1)

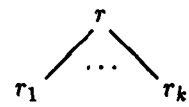
where $r_1 \ldots r_k$ are (not necessarily all) successors of r. We call (1) the *initial local r-figure* if $r_1, \ldots, r_k$ are all such rules that $r_1-, \ldots, r_k-$redexes have outermost occurrences in $RHS(r)$. If F_R is a set of R-rules such that F_R-essential redexes in $RHS(r)$ are exactly the occurrences of $r_1-, \ldots, r_k-$redexes, then we call (1) the *local r-figure corresponding to F_R* or just the *local F_R-r-figure.*

Remark 6.1 Obviously, the initial local r-figure is the $\emptyset$-r-figure.

Lemma 6.2 Let r be a rule in a PTRS, F_R be a set of its successors, (1) be the local F_R-r-figure, and let $\{r_1, ..., r_k\} \subset F_R$. Then (1) is the essential local r-tree.
Proof By Proposition 6.3.(a).

Proposition 6.6 Let R be a PTRS containing n rules and let $r \in R$. We can find a cycle in the essential r-tree if r is strictly infinite and otherwise construct the essential r-tree (the height of which is less then n) and find irreducible rules and essentiality indicators for rules in the essential r-tree as follows.

Algorithm 6.6 First step: Construct the initial r-figure T_1

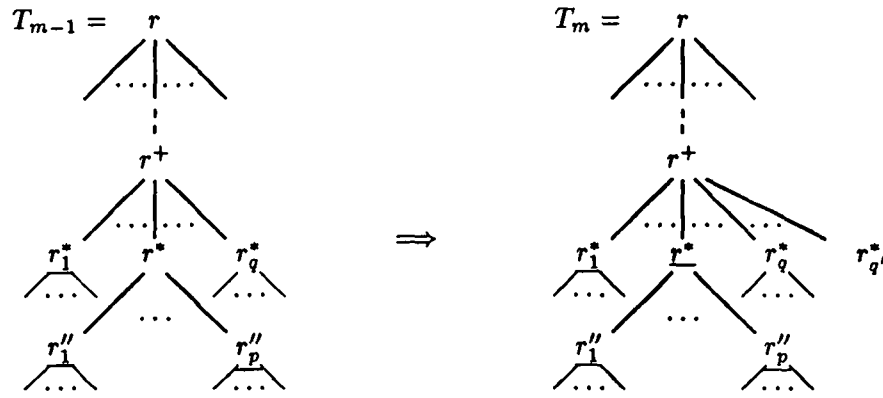


If $k = 0$ then find $IR(r)$ (it coincides with r) and $EI(r)$ using Algorithm 6.4. The essential r-tree is constructed. It is just r and the algorithm stops. If $k \neq 0$ and some r_i coincides with r, then the algorithm stops by finding the cycle r, r_i. Otherwise proceed by performing the second step.

m^{th} step ($m > 1$). Choose in the figure T_{m-1}, constructed in $(m-1)^{th}$ step, either any unmarked leaf r^* or any unmarked node r^* all successors of which are marked. If such a node does not exist, i.e., each node in T_{m-1} is marked, then the algorithm stops; T_{m-1} (without marks) is the essential r-tree, and essentiality indicators and irreducible rules for all rules in the essential r-tree are found. Otherwise:

(a) If r^* is a leaf and is not irreducible, then construct T_m from T_{m-1} by substituting the initial local r^*-figure for r^*.

(b) If r^* is not a leaf or is an irreducible leaf, then mark r^*, using Algorithm 6.4 find $IR(r^*)$ and $EI(r^*)$, and construct T_m from T_{m-1} by adding new brothers $r^*_{q+1}, ..., r^*_{q'}$ of r^*



where the local r^+-figure in T_m corresponds to the set of all marked successors of r^+ in T_{m-1}.

If in T_m there is a cycle, then the algorithm stops. Otherwise proceed by performing the next step.

Proof It follows from Lemma 6.2 that in the case (b) of the mth step $r_1'', \ldots r_p''$ are all essential successors of r^*. Hence Algorithm 6.4 is applicable and Algorithm 6.6 works. Since in each step of Algorithm 6.6 either new nodes are created or an unmarked node is marked, it terminates eventually by finding a cycle or by constructing the essential r-tree. Now the lemma follows from Theorem 3.6.

Theorem 6.1 Let R be a PTRS containing a finite number of rules and t be a term in R. Then strong normalization is decidable for R and t. The decision algorithms are as follows.

Algorithm 6.7 (a) Using Algorithm 6.1, for each rule in R establish whether it is finite or not. R is strongly normalizable iff each rule in R is finite.

Algorithm 6.8 (b) Using Algorithm 6.1, for each rule from R whose corresponding redex occurs in t establish whether it is finite or not. Then t is strongly normalizable iff each such a rule is finite.

Proof By Theorem 3.2, Theorem 3.5, and Proposition 6.1.

Theorem 6.2 Let R be a PTRS containing a finite number of rules. Then it is decidable whether R is weakly normalizable or not. A decision algorithm is as follows.

Algorithm 6.9 Using Algorithm 6.6, establish for each rule in R whether it is semi-finite or not. R is weakly normalizable iff each rule in R is semi-finite.

Proof By Proposition 6.6 and Theorem 3.7.

Theorem 6.3 Let t be a term in a PTRS with a finite number of rules. We can establish whether t is weakly normalizable or not and in the case of positive answer find its normal form $!t$ in the least number of steps using the following algorithm:

Algorithm 6.10 Using Algorithm 6.6, for each rule whose corresponding redex occurs in t establish whether it is semi-finite or not, and in the case of positive answer find corresponding irreducible rules and essentiality indicators. If at least one strictly infinite redex is not in an inessential argument of a semi-finite redex, then t is not weakly normalizable and the algorithm stops. Otherwise replace all strictly infinite redexes by extra variables and using Algorithm 6.3 find its normal form. It coincides with $!t$.

Proof By Propositions 6.3 and 6.6, Corollary 2.2, and Theorems 3.3 and 5.2.(1).

Theorem 6.4 Let t be a term in a PTRS with a finite number of rules. Then the greatest lower bound $l(t)$ of lengths of normalizing reductions starting from t can be found using the following

Algorithm 6.11 Using Algorithm 6.6, for each rule whose corresponding redex occurs in t establish whether it is semi-finite or not and find essentiality indicators of semi-finite rules. If at least one strictly infinite redex is not in an inessential argument of a semi-finite redex, then $l(t)$ is infinite. Otherwise, using Algorithm 6.3, find all essential redexes $u_1, \ldots, u_k$ in t and using Definition 4.2 find numbers $\|u_1\|, \ldots, \|u_k\|$. Then $l(t) = \sum_{i=1}^{n} \|u_i\|$.

Proof By Propositions 6.6 and 6.3 and Theorem 4.2.

Theorem 6.5 Let R and R' be PTRSs with the same alphabet containing a finite number of rules. Then it is decidable whether $R \simeq R'$ or not. A decision algorithm is as follows.

Algorithm 6.12 For each rule $r : t \to s$ in R, establish using Algorithm 6.10 whether s is weakly normalizable or not and find R-nf of s if it exists. Replace all semi-finite rules in R by corresponding irreducible rules, constructing this way $IR(R)$. Do the same for R'. Then $R \simeq R'$ iff $IR(R)$ and $IR(R')$ are identical except for the RHSs of strictly infinite rules.

Proof By Theorems 5.3, 5.2, and 6.3.

Theorem 6.6 Let t and s be any terms in a PTRS R with a finite number of rewrite rules. Then it is decidable whether $t \twoheadrightarrow s$ or not.

Proof First observe that $t \twoheadrightarrow s$ iff there is a term t' obtained from t by an outermost reduction such that the top-symbol of t' coincides with the top-symbol of s and $t' \twoheadrightarrow s$. Now we prove the theorem by induction on the length of s.

(1) Suppose first that t and s are not redexes. If top-symbols of t and s coincide, then the decidability of $t \twoheadrightarrow s$ follows from the induction assumption. Otherwise t cannot be reduced to s.

(2) Let t be a redex and s be a non-redex. Further, let $P : t \to t_1 \to \ldots \to t_n$ be an outermost reduction, where n is the number of the rewrite rules in R. If at least one term t_i is not a redex and t_j is such a term with the minimum j, then $t \twoheadrightarrow s$ iff $t_j \twoheadrightarrow s$ and the decidability of $t_j \twoheadrightarrow s$ is established as in the case (1). Otherwise, contraction of outermost redexes gives a "cyclic" reduction containing only the terms that are already in P. Hence t cannot be reduced to a non-redex, in particular to s.

(3) Let both t and s be redexes. If among terms in P there is a redex similar to s, then we can use the induction assumption. If among terms in P there is no redex similar to s but there is a non-redex t_k, then t cannot be reduced to s. Otherwise, similarly to the last case in (2), all terms in the infinite outermost reduction starting from t are redexes that are not similar to s. Hence t cannot be reduced to s.

Acknowledgements

I enjoyed discussions with H. Barendregt, P.-L. Curien, J. W. Klop, J.-J. Lévy, A. Middeldorp, G. Mints, Sh. Pkhakadze, Kh. Rukhaia and V. Sazonov. I thank H. Barendregt, P.-L. Curien, H. Ganzinger and J. W. Klop for organizing my talks at Nijmegen, Paris, Saarbrücken and Amsterdam. I also would like to thank Sh. Aditya, D. Doligez, I. Jacobs, J.-J.'Lévy, G. Tagviashvili, and K. Urbaitis for their help in the preparation of this paper.

References

[1] Barendregt H.P. The Lambda Calculus, its Syntax and Semantics. 2nd ed. North-Holland, 1984.

[2] Berry G, Lévy J.-J. Minimal and optimal computations of recursive programs. JACM 26, 1979, p.148-175.

[3] Courcelle B. Recursive Applicative Program Schemes. In: J.van Leeuwen ed. Handbook of Theoretical Computer Science, Chapter 9, vol.B, 1990, p. 459-492.

[4] Dershowitz N, Jouannaud J.-P. Rewrite Systems. In: J.van Leeuwen ed. Handbook of Theoretical Computer Science, Chapter 6, vol.B, 1990, p. 243-320.

[5] Huet G, Lévy J.-J. Computations in Orthogonal Rewriting Systems, in Computational Logic, Essays in Honor of Alan Robinson, ed. by J.-L. Lassez and G. Plotkin, MIT Press, 1991.

[6] Khasidashvili Z. Minimal Normalizing Strategy for Orthogonal and Persistent Term Rewriting Systems (short version). Proceedings of I.Vekua Institute of Applied Mathematics, vol.36, 1990. p.189-199.

[7] Klop J.W. Term Rewriting Systems. In: Handbook of Logic in Computer Science (eds. S.Abramsky, D.Gabbay and T.Maibaum). Oxford University Press, 1990.

[8] Lévy J.-J., Optimal Reduction in the Lambda-Calculus. In: To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, J. P. Seldin and J. R. Hindley editors, Academic Press, 1990.

[9] Maranget L. "La stratégie paresseuse", These de l'Universite' de PARIS VII, 1992.

[10] Montangero C, Pacini G, Turini F. Graph representation and computation rules for typeless recursive program schemes. Springer LNCS, 1974.

[11] O'Donnell M.J. Computing in systems described by equations. Springer LNCS 58, 1977.

[12] Vuillemin J. Proof techniques for recursive programs. Ph.D. Thesis, Stanford University, Stanford, 1973.

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399

RR_1837