Higher order recursive program schemes are Turing incomplete

Z. Khasidashvili

Computer Science/Department of Software Technology

Report CS-R9348 July 1993
CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications. SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.
Higher Order Recursive Program Schemes are Turing Incomplete

Zura Khasidashvili
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract

We define Higher Order Recursive Program Schemes (HRPSs) by allowing metasubstitutions (as in the \(\lambda\)-calculus) in right-hand sides of function and quantifier definitions. To study reductions in a HRPS we split it into "first order part" and "pure substitution part". A study of pure substitutions and several kinds of similarity of redexes makes it possible to lift properties of (first order) Recursive Program Schemes to the higher order case. The crucial properties are that corresponding arguments of essentially similar redexes are either both essential or both inessential, and that essentially similar redexes create essentially similar redexes. The main result is the decidability of weak normalization in HRPSs, which immediately implies that HRPSs do not have full computational power. We analyze the structural properties of HRPSs and introduce several kinds of persistent higher order rewrite systems that enjoy similar properties. For uniformly persistent systems, we design an efficient optimal sequential normalizing strategy.

AMS Subject Classification (1991): 68Q42.
Keywords & Phrases: orthogonal combinatory reduction systems, higher order recursive program schemes, strong normalization, weak normalization, reduction strategies.
Note: Part of this work was completed during a visit of the author at CWI in the summer of 1993.

1. Introduction

Higher Order Recursive Program Schemes (HRPSs) are recursive definitions of functions, predicates, and quantifiers, considered as rewriting systems. Similar definitions are used when one extends a theory by introducing new symbols. For example, the existential quantifier \(\exists\) and the quantifier \(\exists!\) for "there is exactly one" can be defined using Hilbert's operator (sign) \(\tau\) as follows:

\[\exists a A \leftrightarrow (\tau a A/a) A\]

\[\exists! a A \leftrightarrow \exists a A \land \forall a \forall b (A \land (b/a) A \Rightarrow a = b)\]

The definitions of new symbols are added to the theory as axioms, and weak normalization of the system of definitions considered as a rewriting system ensures that the enriched theory is a conservative extension of the original one. A study of definitions as contraction schemes is made in [18].

The main result of the paper — decidability of weak normalization (that is, existence of a normal form) in HRPSs — implies that HRPSs do not have full computational power: the existence of an interpreter for, say \(\lambda\)-calculus, in an HRPS would imply decidability of weak normalization for \(\lambda\)-terms, which is not valid [3]. Note that rewrite rules for the conditional cannot be used for evaluating expression on the syntactic level, since they do not fall into the scope of our HRPSs.
1. Introduction

Indeed, it is well-known that addition of \textit{if} – \textit{then} – \textit{else} yields full computational power (see for example Courcelle [4]). The deep reason for Turing incompleteness of HRPSs is the restricted possibility for redex creation compared to languages with full computational power, the \( \lambda \)-calculus and Combinatory Logic, for example. This paper is a part of a general study of how various kinds of redex creation are reflected in syntactic properties of rewriting systems such as normalization, perpetuality, decidability of weak and strong normalization (that is, termination of all reductions), expressive power, etc. Some results in this direction are obtained in [10].

In [9], we introduced a formalism for higher order rewriting (i.e., term rewriting systems with bound variables and substitution mechanism) which is close to Combinatory Reduction Systems (CRSs) of Klopf [11]. Our syntax is more close to the syntax of \( \lambda \)-calculus and First Order Logic. For example, the \( \beta \)-rule is written as

\[
\beta : A p(\lambda a A, B) \rightarrow (B/a)A,
\]

where \( a \) is to be instantiated by a variable and \( A \) and \( B \) are to be instantiated by terms. The expression \((B/a)A\) is a metasubstitution, and its instance \((t/x)s\) denotes the result of substitution of the term \( t \) for \( x \) in the term \( s \). To express “pure” substitutions syntactically (instead of metasynthetically) we introduce \( S \)-reduction rules

\[
S^{n+1}a_1 \ldots a_nA_1 \ldots A_nA_0 \rightarrow (A_1/a_1, \ldots, A_n/a_n)A_0, \quad n = 1, 2, \ldots
\]

The \textit{substitution operator} \( S^{n+1} \) binds free variables only in the last argument. The difference with \( \beta \)-rules is that \( S \)-reductions can only express \( \beta \)-developments of \( \lambda \)-terms [8]. Therefore \( S \)-reductions have more simple properties and their study enables us to construct an interpretation of HRPSs in (first order) Recursive Program Schemes (RPSs).

To this end, we split an HRPS \( R \) into a “first order part” \( R_f \) and a “substitution part” \( S \). (The method was originally used by Klopf [11] to prove the Church-Rosser Property. Klopf used \( \beta \)-reductions instead of \( S \)-reductions.) The \( R_f \)-rules are obtained from \( R \)-rules by replacing in right-hand sides all metasubstitutions of the form \((A_1/a_1, \ldots, A_n/a_n)A_0 \) by \( S a_1 \ldots a_nA_1 \ldots A_nA_0 \), respectively, and the \( S \)-rules have the form

\[
S^{n+1}a_1 \ldots a_nA_1 \ldots A_nA_0 \rightarrow (A_1/a_1, \ldots, A_n/a_n)A_0, \quad n = 1, 2, \ldots
\]

So \( S \)-rules are just \( S \)-rules: we only need to distinguish \( S \)-redexes that are created during \( R_f \)-reduction steps. For example, a \( \beta \)-reduction step \((\lambda xt)s \rightarrow (s/x)t\) expands to a \( \beta \)-reduction step \((\lambda x^t)s \rightarrow S x t \) and an \( S \)-step \( S x t \rightarrow (s/x)t \). Now in the promised interpretation of HRPSs in RPSs, \( S \)-reductions play the role of projection functions. Ignoring some subtleties, the interpretation can be expressed by the following Representation Lemma: for any \( R \)-reduction \( P : t \rightarrow s \) there is an \( R_f \)-reduction \( P_f : t \rightarrow o \) such that \( s \) is obtained from \( o \) by any normalizing \( S \)-reduction \( P_S \); and conversely (see Lemma 3.9 for the precise formulation). Moreover, \( P \) is \textit{strictly equivalent} to the concatenation \( P_f + P_S \) of \( P_f \) and \( P_S \) in the following sense: \textit{descendants} of any subterm of \( t \) under \( P \) and \( P_f + P_S \) are the same occurrences in \( s \). The notion of descendant is a generalization of the usual notion of \textit{residuals of redexes} to all subterms in a way that makes it possible to trace contracted redexes: the descendent of a contracted redex is its contractum, while it does not have residuals. For the case of TRSs, this notion is studied in [10].

As in the case of orthogonal Term Rewriting Systems (TRSs), we say that a subterm \( s \) of a term \( t \) in a CRS \( R \) is \textit{inessential} if there is a reduction \( P \) starting from \( t \) such that \( s \) does not have descendants under \( P \); we call \( s \) \textit{essential} otherwise. Now an immediate corollary of the
1. Introduction

Representation Lemma is that $s$ is $R$-inessential in $t$ iff there is an $R$-reduction $P : t \rightarrow e$ such that each $P$-descendant of $s$ is $S$-inessential in $e$. Since $S$-reductions (like $\beta$-developments) are terminating, it is decidable whether a subterm is $S$-inessential.

We can now introduce the key concepts of the paper: several kinds of similarity of redexes. We explain the notion informally in terms of $\beta$-redexes. Let us consider the redexes $u = (\lambda x . x) s$, $v = (\lambda x . (\lambda y . z) x) o$, $w = (\lambda x . x z) e$, and $w' = (\lambda x . x y) t$. We call $u$ and $v$ similar (written $u \sim v$) because the binding variable $x$ occurs free both in the bodies of $u$ and $v$. In contrast, we have $u \neq u'$. We call $u$ and $v$ essentially similar (written $u \approx w$) because the binding variable $x$ has essential occurrences in bodies of both redexes. Similarly, $v \approx u'$, because $x$ occurs in the body $(\lambda y . z) x$ of $v$, but the occurrence is inessential — it does not have descendants in the contractum of $(\lambda y . z) x$. In general, let us call a $\beta$-redex $(\lambda x . e) o$ an $E$-redex if $x$ has an essential occurrence in $e$. (Recall that $(\lambda x . e) o$ is an $I$-redex if $x$ occurs free in $e$ and is a $K$-redex otherwise.) Then $I$-redexes are similar, $K$-redexes are both similar and essentially similar, and $E$-redexes are essentially similar. An important property of $E$-redexes is that they stay $E$-redexes under any reduction. Further refinement of the notion of similarity takes into account also the number of occurrences and essential occurrences of a binding variable in corresponding arguments. These similarity notions are formalized by special characteristic systems. For example, the essential characteristic system of a redex $\sigma x_1 \ldots x_n A_1 \ldots A_m$ corresponding to a rewrite rule $\sigma a_1 \ldots a_n A_1 \ldots A_m \rightarrow B$ is the set of all pairs $(a_i, A_j)$ such that $t_j$ is in the scope of $\sigma$ and $x_i$ has an essential occurrence in $t_j$. We call a pair $(\sigma, ECS(\sigma))$, where $\sigma$ is a rule and $ECS(\sigma)$ is an essential characteristic system for some $\sigma$-redex, an essentially characterized rule ($EC$-rule for short).

Let a term $t$ be obtained from $s$ by replacing some subterms. We call $t$ and $s$ essentially similar if corresponding redexes in $t$ and $s$ (that are outside of replaced subterms) are essentially similar. The crucial property of essentially similar terms is the following Essential Similarity Lemma: corresponding subterms of essentially similar terms are either both essential or both inessential. A consequence is that corresponding arguments of essentially similar redexes are either both essential or both inessential. Hence each $EC$-rule $(\sigma, ECS(\sigma))$ has an essentiality indicator which indicates which arguments of a redex with the essential characteristic system $ECS(\sigma)$ ($(\sigma, ECS(\sigma))$-redex for short) are essential. Another consequence of the Essential Similarity Lemma is that essentially similar redexes create essentially similar redexes.

Now, the strategy for proving the decidability of weak normalization in HRPSs is similar to the first order case [10]. We first establish a criterion for weak normalization. We define an essential chain to be a sequence of $EC$-rules $(\sigma_0, ECS(\sigma_0)), (\sigma_1, ECS(\sigma_1)), \ldots$ such that any $(\sigma_i, ECS(\sigma_i))$-redex creates an $(\sigma_{i+1}, ECS(\sigma_{i+1}))$-redex ($i = 0, 1, \ldots$). We show that a term in an HRPS $R$ is weakly normalizable iff any essential chain, starting from any $EC$-rule $(\sigma, ECS(\sigma))$ such that a $(\sigma, ECS(\sigma))$-redex has an essential occurrence in $t$, is finite. So we need an algorithm for finding all essential (and inessential) subterms in a given term, which also will enable us to find essential characteristic systems of all redexes in $t$, and an algorithm for finding essentiality indicators of all $EC$-rules. Complete descriptions of the algorithms are presented in the body of the paper; we only note that our algorithm can also find essentiality indicators of redexes that do not have normal forms. As a corollary, we have an algorithm for optimal $S$-reductions and, in general, algorithms for optimal developments in orthogonal CRSs.

The above properties of HRPSs are due to the fact that each subterm $s$ of a term $t$ is free: if $s$ is inside a redex $u$, then it is either inside an argument of $u$ or coincides with $u$. We introduce persistent CRSs (PCRSs), where not all subterms are free but all free subterms remain free under any reduction. This persistency property of free subterms is due to the fact that PCRSs are
the systems where only a special kind of redex creation — generation — is possible: all new redexes are actually presented in right-hand sides of rewrite rules. For example, in the step $\sigma xx \rightarrow f((\sigma A/a)x) = f(\sigma xx)$, which corresponds to a rule $\sigma A \rightarrow f((\sigma A/a)A)$ and an assignment $\theta(a) = \theta(A) = x$, the created redex $\sigma xx$ is an instance of the metaterm $\sigma A$ presented in the right-hand side of the rewrite rule. We have generation here because $\theta(a)$ has an occurrence in $\theta(A)$. Therefore, the generation is not uniform — uniformly generated redexes are presented in right-hand sides of rules outside of “mobile” arguments of metasubstitutions. During the reduction step $\sigma xx \rightarrow f((d/x)x) = f(d)$ in the system $\{r_1 : \sigma A \rightarrow f((d/A)A), r_2 : f(d) \rightarrow c\}$, where $d$ and $c$ are constants, we have quasi-generation in the sense that all symbols needed to create the new redex $f(d)$ are presented in the right-hand side of $r_1$.

The Essential Similarity Lemma remains valid in PCRSs if the replaced subterms are free. Therefore, all the results obtained for HRPSs, the decidability of weak normalization in particular, remain valid for PCRSs in general. Similar reasoning shows that strong normalization also is decidable in PCRSs. For uniformly persistent systems, we show that the strategy that in each step contracts an innermost essential redex is optimal. For PCRSs in general, sharing of redexes of the same origin [14] is necessary, and this can be implemented in the framework of Interaction Systems [2]. However, the above results can not be generalized to quasi-persistent CRSs, where quasi-generation of redexes is possible.

2. ORTHOGONAL COMBINATORY REDUCTION SYSTEMS

Combinatory Reduction Systems have been introduced in Klop [11] to provide a uniform framework for reductions with substitutions, as in the $\lambda$-calculus and its extensions [3]. Different formalisms are proposed in Kennaway and Sleep [7] ((Functional) Combinatory Reduction Systems), Khasidashvili [8] (Expression Reduction Systems), and Nipkow [16] (Higher-order Rewrite Systems). They are extensions of Term Rewriting Systems [5, 12] by means of variable binding and substitution mechanisms. Restricted notions of CRs were first introduced in Pkhakadze [18] and Aczel [1]. A comparison of some formalisms of rewriting systems with bound variables and substitution mechanism (referred to also as higher order rewrite systems) can be found in van Oostrom and van Raamsdonk [17]. A survey paper is Klop et al. [13]. Here we describe a system for higher order rewriting as defined in Khasidashvili [9]; it is based on the syntax of [18].

Definition 2.1 (1) Let $\Sigma$ be an alphabet, comprising variables $v_0, v_1, \ldots$; function symbols, also called simple operators; and operator signs or quantifier signs. Each function symbol has an arity $k \in N$, and each operator sign $\sigma$ has an arity $(m, n)$ with $m, n \neq 0$ such that, for any sequence $x_1, \ldots, x_m$ of pairwise distinct variables, $\sigma x_1 \ldots x_m$ is a compound operator or a quantifier with arity $n$. Occurrences of $x_1, \ldots, x_m$ in $\sigma x_1 \ldots x_m$ are called binding variables. Each quantifier $\sigma x_1 \ldots x_m$, as well as corresponding quantifier sign $\sigma$ and binding variables $x_1 \ldots x_m$, has a scope indicator $(k_1, \ldots, k_l)$ to specify the arguments in which $\sigma x_1 \ldots x_m$ binds all free occurrences of $x_1, \ldots, x_m$. Terms are constructed from variables using functions and quantifiers in the usual way.

(2) Metaterms are constructed from terms, term metavariables, which range over terms, and object metavariables, which range over variables. Apart from the usual rules for term-formation, one is allowed to have metasubstitutions — expressions of the form $(A_1/a_1, \ldots, A_n/a_n)A_0$, where $a_i$ are object metavariables and $A_j$ are metaterms. Metaterms that do not contain metasubstitutions are called simple metaterms. An assignment maps each object metavariable to a variable and each term metavariable to a term over $\Sigma$. If $t$ is a metaterm and $\theta$ is an assignment, then the $\theta$-instance $\theta t$ of $t$ is the term obtained from $t$ by replacing metavariables with their values under $\theta$, and by replacement of subterms of the form $(t_1/x_1, \ldots, t_n/x_n)t$ by the result of substitution of
terms $t_1, \ldots, t_n$ for free occurrences of $x_1, \ldots, x_n$ in $t$.

**Definition 2.2** (1) A *Combinatory Reduction System* (CRS) is a pair $(\Sigma, R)$, where $\Sigma$ is an alphabet, described in Definition 2.1, and $R$ is a set of rewrite rules $r : t \rightarrow s$, where $t$ and $s$ are metaterms such that $t$ is a simple metaterm and is not a metavariiable, and each term metavariable which occurs in $s$ occurs also in $t$. Further,

(a) The metaterms $t$ and $s$ do not contain variables, and each occurrence of object metavariable in $t$ and $s$ is bound.

(b) An occurrence of a term-metaveriable $A$ in $s$ is in the scope of an occurrence of an object metaveriable $a$ in $s$ iff any occurrence of $A$ in $t$ is in the scope of an occurrence of $a$ in $t$.

(2) An assignment $\theta$ is *admissible* for a rule $r : t \rightarrow s$ if occurrences of binding variables in $s\theta$ corresponding to object metavariables of $s$ not occurring in $t$ do not bind variables in subterms corresponding to term metavariables of $s$. For any admissible assignment $\theta'$, $t\theta'$ is an $r$-redex, and $s\theta$ is the *contractum* of $t\theta$. Redexes that are instances of the left-hand side of the same rule are called *weakly similar*.

(3) $R$ is *simple* if right-hand sides of $R$-rules are simple.

**Remark 2.1** Terms $a$ and $e$ are called *congruent*, notation $a \equiv e$, if $a$ is obtained from $e$ by renaming bound variables. The conditions in Definition 2.2 imply that, for any rule $r : t \rightarrow s$, if $\theta, \theta' \in AA(r)$, then $t\theta \equiv t\theta'$ implies $s\theta \equiv s\theta'$. Below we identify all congruent terms.

**Notation** We use $a, b$ for object metavariables, $A, B$ for term metavariables, $c, d$ for constants, $t, s, e, a$ for terms and metavariables, $u, v, w$ for redexes, $\sigma$ for operators and operator signs, and $P, Q$ for reductions. We write $s \subseteq t$ if $s$ is a subterm of $t$. A one-step 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 $P : t \rightarrow s$ if $P$ denotes a reduction of $t$ to $s$. $|P|$ denotes the length, i.e., the number of steps, of $P$. If the last term of $P$ coincides with the initial term of $Q$, then $P + Q$ denotes the concatenation of $P$ and $Q$. $\emptyset_t$, or simply $\emptyset$, denotes the empty reduction of a term $t$; the symbol $\emptyset$ is also used to denote the empty set.

**Definition 2.3** A term $t$ in a CRS $R$ is said to be in *normal form* (nf) or to be a nf if it does not contain redexes. If $s \rightarrow t$ and $t$ is a nf, then $t$ is called a normal form of $s$. A term is called *weakly normalizable* if it has a nf and is called *strongly normalizable* if it does not possess an infinite reduction. A CRS $R$ is called *weakly normalizing* (resp. *strongly normalizing*) if each term in $R$ is weakly normalizable (resp. strongly normalizable).

**Definition 2.4** Let $t \rightarrow s$ be a rule in a CRS $R$ and $\theta$ be an assignment. Subterms of a redex $v = t\theta$ that correspond to term metavariables 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 $v$. If $R$ is a simple CRS, then arguments, pattern, and pattern-subterms are defined analogously in the contractum $s\theta$ of $v$.

**Definition 2.5** A rewrite rule $t \rightarrow s$ in a CRS $R$ is *left-linear* if it is linear, i.e., no term metavariable occurs more than once in $t$. $R$ is *left-linear* if each rule in $R$ is so. $R$ is *non-ambiguous* or *non-overlapping* if no term redex-patterns can overlap. $R$ is *orthogonal* (OCRS) if it is left-linear and non-overlapping.

**Definition 2.6** The CRS $S$ comprises rules of the form

$$\text{S}^{n+1} a_1 \ldots a_n A_1 \ldots A_n A \rightarrow (A_1/a_1, \ldots, A_n/a_n)A, \ n = 1, 2, \ldots,$$

where $\text{S}^{n+1}$ is the *operator sign of substitution* with arity $(n, n + 1)$ and scope indicator $(n + 1)$, and $c_1, \ldots, a_n$ and $A_1, \ldots, A_n, A$ are pairwise distinct object and term metavariables, respectively.
Definition 2.7 Let \( R = \{r_i : t_i \rightarrow s_i | i \in I\} \) be a CRS.

1. \( R_f =_{\text{def}} \{r'_i : t_i \rightarrow s'_i | i \in I\} \), where \( s'_i \) is obtained from \( s_i \) by replacing all metasubstitutions of the form \( t_1/a_1, \ldots, t_n/a_n \) t with \( S_{n+1} a_1 \cdots a_n t_1 \cdots t_n t \), respectively (\( S_{n+1} \) is a fresh symbol with the same arity and scope indicator as \( S^n \)).

2. If \( R \) is simple, then \( R_{FS} =_{\text{def}} R_f =_{\text{def}} R \). Otherwise \( R_{FS} =_{\text{def}} R_f \cup S \), where \( S \)-rules are obtained from \( S \)-rules by underlining the \( S \)-symbols.

3. For each step \( e = C[t_1, \theta] \xrightarrow{u} C[s, \theta] = e \) in \( R \) (corresponding to the rule \( r_i \) and an admissible assignment \( \theta \)) there is a reduction \( P : e = C[t_1, \theta] \rightarrow C[s, \theta] \rightarrow C[s, \theta] = e \in R_{FS} \), where \( C[s', \theta] \rightarrow C[s, \theta] \) is the rightmost innermost normalizing \( S \)-reduction. We call \( P \) the expansion of \( u \) and denote it by \( Ex(u) \). The notion of expansion generalizes naturally to \( R \)-reductions with 0 or more steps.

Definition 2.8

1. Let \( t \xrightarrow{u} s \) in a simple OCRS and \( e \) be the contractum of \( u \) in \( s \). For each argument \( t^* \) of \( u \) there are 0 or more arguments of \( e \). We call them \((u-)descendants of \( t^* \). Correspondingly, subterms of \( t^* \) have 0 or more descendants. The descendant of each pattern-subterm of \( u \) that is not a variable is \( e \). (We do not define descendants of "variable pattern-subterms", which are binding variables.) It is clear what is to be meant under descendants of subterms that are not in \( u \). The notion of descendant extends naturally to arbitrary reductions in simple OCRSs.

2. Let \( t \xrightarrow{u} s \), where \( u = S x_1 \cdots x_n t_1 \cdots t_n t_0 \), and let \( e \) be the contractum of \( u \) in \( s \). For each mobile argument \( t_i \) of \( u \) \((i = 1, \ldots, n) \) there are substituted occurrences of \( t_i \) in \( e \). We call them \( u\)-descendants of \( t_i \). By definition, they also are \( u\)-descendants of corresponding free occurrences of the variable \( x_i \) in \( t_0 \). Subterms in \( t_i \) have the same number (possibly none) of descendants in \( s \). The descendant of \( u \) is \( e \). It is clear what is to be meant under descendants of subterms that are not in \( u \) or are in \( t_0 \) and are not free occurrences of variables \( x_1, \ldots, x_n \). The notion of descendant extends naturally to \( S \)-reductions with 0 or more steps.

3. Let \( P : t \rightarrow s \) in an OCRS \( R \) and let \( Q = Ex(P) \). It is clear from (1) and (2) what is to be meant under \( Q \)-descendants of subterms in \( t \). We call a subterm \( o \in t \) a \( P \)-descendant of a subterm \( o \in t \) if \( o \) is a \( Q \)-descendant of \( o \), and call \( o \) in this case a \( P \)-ancestor of \( o \).

4. Let \( t \xrightarrow{u} s \). Descendants of all redexes of \( t \) except \( u \) are also called residuals. By definition, \( u \) does not have residuals in \( s \). A redex \( v \subseteq s \) is a \((u-)new redex or a created redex if it is not a residual of a redex in \( t \). The notion of residual of redexes extends naturally to reductions with 0 or more steps.

Definition 2.9 We call the co-initial reductions \( P : t \rightarrow s \) and \( Q : t \rightarrow e \) strictly equivalent (written \( P \approx_{st} Q \)) if \( s = e \) and \( P \)-descendants and \( Q \)-descendants of any subterm of \( t \) are the same in \( s \) and \( e \).

Notation If \( F \) is a set of redexes in \( t \) and \( P : t \rightarrow s \), 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 2.10 Let \( Q : t \rightarrow s \) and \( t \xrightarrow{u} e \). Then the residual \( Q/u \) of \( Q \) by \( u \) is defined modulo permutation of non-overlapping steps by induction on \(|Q|\) as follows. If \( Q = \emptyset \), then \( Q/u = \emptyset_e \). If \( Q = Q' + v \), then \( Q/u = Q'/u + v/(u/Q') \).
Definition 2.11 Let $P : t \rightarrow s$ and $Q : t \rightarrow e$. Then the residual $P/Q$ of $P$ by $Q$ and the residual $Q/P$ of $Q$ by $P$ are defined modulo permutation of non-overlapping steps by induction on $|P|$ as follows.

1. If $P = \emptyset$, then $P/Q = \emptyset$ and $Q/P = Q$.
2. If $P = P' + u$, then $P/Q = P'/Q + u/(Q/P')$ and $Q/P = (Q/P')/u$.

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

Theorem 2.1 (Strict Church-Rosser [9]) Let $R$ be an OCRS, and $P$ and $Q$ be co-initial reductions in $R$. Then $P \cup Q \approx_{st} Q \cup P$.

3. SIMILARITY OF REDEXES

In this section, we study some properties of substitutions and similarity of terms. In particular we prove the Replacement Lemma, the Essential Similarity Lemma, and the Uniform Generation Lemma.

Definition 3.1 We call an OCRS $R$ a Higher Order Recursive Program Scheme (HRPSs) if, for any rule $t \rightarrow s$, pattern of $t$ consists of one operator, i.e., $t$ has the form $\sigma a_1 \ldots a_m A_1 \ldots A_n$, where $\sigma$ is an operator sign with arity $(m, n)$ ($\sigma$ is a function if $m, n = 0$).

Definition 3.2 Let $t$ be a term in an OCRS $R$. 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.

The notion of essentiality is a generalization of the notion of neededness [6, 15] in a way that it works for all subterms, bound variables in particular. (See [10] for the precise relationship between the notions). The following two lemmas are valid for all OCRSs; the proofs are similar to the case of orthogonal TRSs [10].

Lemma 3.1 Let $s_0, \ldots, s_k \subseteq t$ be 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 Let $P_i$ be a reduction starting from $t$ such that $s_i$ does not have $P_i$-descendants ($P_i$ exists since $IE(s_i, t)$). Then, by Theorem 2.1, one can take $P = (P_1 \cup P_2 \cup \ldots \cup P_n)$.

Lemma 3.2 Let $P : t \rightarrow t'$ and $s \subseteq t$. Then $IE(s, t)$ if and only if $P$-descendant $s'$ of $s$ is inessential in $t'$. In particular, if $t'$ is a normal form, then $ES(s, t)$ if $s$ has a $P$-descendant.

Proof ($\Rightarrow$) Let $IE(s, t)$. Then there is some reduction $Q$ starting from $t$ such that $s$ does not have $Q$-descendants. By Theorem 2.1, $P + Q/P \approx_{st} Q + P/Q$. Hence, $s'$ does not have $P/Q$-descendants, i.e., $IE(s', t')$. ($\Leftarrow$) If all $v$-descendants of $s$ are inessential in $t'$, then, by Lemma 3.1, there is some reduction $P'$ starting from $t'$ under which none of them have descendants. Thus $s$ does not have $P + P'$-descendants, i.e., $IE(s, t)$.

Definition 3.3 We call a CRS $R'$ a subsystem of a CRS $R$ if the alphabet of $R'$ is a subset of the alphabet of $R$ and the set of $R'$-rules is a subset of the set of $R$-rules.

Notation Below $EFV_R(t)$ denotes the set of variables having $R$-essential free occurrences in $t$ and $FV(s)$ denotes the set of variables having free occurrences in $s$. We write $t = (e_1/t_1, \ldots, e_k/t_k)e$ if $t$ is obtained from $e$ by replacing non-overlapping proper subterms $e_1, \ldots, e_k$ in $e$ with $t_1, \ldots, t_k$, respectively. For any $s \subseteq t$, $BV_R(s)$ (resp. $EBV_R(s)$) denotes the set of free occurrences (resp. $R$-essential free occurrences) of $s$ bound by quantifiers belonging to patterns of $R$-redexes that are outside $s$. For any subsystems $R_1$ and $R_2$ of $R$, $EBV_{R_1R_2}(s)$ is the set of $R_2$-essential free occurrences of $s$ that are bound by quantifiers belonging to patterns of $R_1$-redexes that are outside $s$. (Note that $EBV_{R,R}(s) = BV_R(s)$ and $EBV_{R,R}(s) = EBV_R(s)$.)
Definition 3.4 (1) Let \( \sigma a_1 \ldots a_n A_1 \ldots A_m \) be the left-hand side of a rewrite rule \( r \) in an HRPS \( R \) and let \( \tau x_1 \ldots x_n t_1 \ldots t_m \) be an \( r \)-redex. The characteristic system of \( u \) (written \( CS(u) \)) is the set of pairs \((a_j, A_i)\) such that \( t_i \) is in the scope of \( a_j \) and \( x_j \in FV(t_i) \) \((i = 1, \ldots, m, j = 1, \ldots, n)\). In this case, \( u \) is an \((r, CS(u))\)-redex. A characterized rule (\( C \)-rule for short) is a pair \((r, CS(r))\), where \( CS(r) \) is a characteristic system for some \( R \)-redex. For any subsystem \( R' \) of \( R \), an \( R'-essential \) characteristic system of \( u \) (written \( ECS_{R'}(u) \)) is the set of pairs \((a_j, A_i)\) such that \( t_i \) is in the scope of \( a_j \) and \( x_j \in FV_{R'}(t_i) \). In this case, \( u \) is an \((r, ECS_{R'}(u))\)-redex. An \( R'-essentially \) characterized rule (\( EC \)-rule for short) is a pair \((r, ECS_{R'}(r))\), where \( ECS_{R'}(r) \) is an \( R'-essential \) characteristic system for some \( R \)-redex. (We omit the subscript \( R' \) if \( R = R' \).)

(2) Let \( t = (t_1/e_1, \ldots, t_k/e_k) e \). We write \( e \prec_R t \) (resp. \( e \prec_R t \)) if \( BV_R(e_i) \subseteq BV_R(t_i) \) (resp. \( EBV_R(e_i) \subseteq EBV_R(t_i) \)) for all \( i = 1, \ldots, k \). We call the terms \( t \) and \( e \) \( R \)-similar (resp. \( R \)-essentially similar), written \( e \sim_R t \) (resp. \( e \approx_R t \)), if \( s \prec_R t \) and \( t \prec_R s \) (resp. \( s \prec_R t \) and \( t \prec_R s \)). We write \( e \prec_{R_1, R_2} t \) if \( EBV_{R_1, R_2}(e_i) \subseteq EBV_{R_1, R_2}(t_i) \) for all \( i = 1, \ldots, k \), and write \( e \sim_{R_1, R_2} t \) if \( e \prec_{R_1, R_2} t \) and \( t \prec_{R_1, R_2} e \). In the latter case, we call \( e \) and \( t \) \((R_1, R_2)\)-essentially similar. (Note that \( \prec_R \) coincides with \( \prec_{R, R} \) and \( \prec_R \) coincides with \( \prec_{R, R} \); therefore, \( R \)-similarity coincides with \((R, \emptyset)\)-similarity and \( R \)-essentially similarity coincides with \((R, R)\)-similarity.)

Of course, (essential) similarity of terms depends on the specification of the replaced subterms. Note that, for any weakly similar \( R \)-redexes \( v \) and \( u \), if one specifies replaced subterms to be the arguments of the redexes, then \( v \prec u \) (resp. \( v \prec u \)) iff \( CS(v) \subseteq CS(u) \) (resp. \( ECS_{R}(v) \subseteq ECS_{R}(u) \)). Below, when we speak of (essential) similarity of redexes without specifying replaced subterms, we mean that the replaced subterms are the arguments of the redexes; we write \( v \prec u \), \( v \prec u \), \( v \approx u \), and \( v \approx u \) for \( v \prec u \), \( v \prec u \), \( v \sim u \), and \( v \approx u \), respectively, to stress the fact that the replaced subterms are specified to be arguments of the redexes.

3.4 The Replacement Lemma

Definition 3.5 Let \( u = S x_1 \ldots x_n t_1 \ldots t_n t_0 \) and \( t_0' \) be an \( S \)-normal form of \( t_0 \). A subterm \( e \) in \( u \) is called \( \nu \)-inessential (written \( IE(u; e) \)) if \( e \) is in \( t_i \) for some \( 1 \leq i \leq n \) and \( x_i \not\in FV(t_0') \).

Lemma 3.3 Let \( u = S x_1 \ldots x_n t_1 \ldots t_n t_0 \subseteq t \). Then \( IE_S(u; t_0) \iff x_i \not\in EFV_S(t_0) \).

Proof By Lemma 3.2, if \( t_0' \) is the \( S \)-normal form of \( t \), then \( EFV_S(t_0) = FV(t_0') \).

Lemma 3.4 Let \( s \subseteq t \). Then \( IE_S(s; t) \iff IE_S(u; s) \) for some \( S \)-redex \( u \) in \( t \).

Proof sketch One can take for \( u \) the redex whose residual erases all descendants of \( s \) in the rightmost innermost normalizing \( S \)-reduction.

Lemma 3.5 Let \( e \subseteq s \subseteq t \). Then \( ES_S(e; t) \iff ES_S(e; s) \) and \( ES_S(s; t) \).

Proof (\( \Rightarrow \)) From Definition 3.2. (\( \Leftarrow \)) By Lemma 3.4, the redex that would make \( e \)-inessential can neither occur in \( s \) nor contain \( e \) in its argument.

Lemma 3.6 (Replacement Lemma) Let \( s = (s_1/\ldots, s_n/\ldots t_n) t \), where \( s_i \) and \( t_i \) do not contain \( S \)-redexes, and let \( t \prec_S s \). Further, let \( s' \) and \( t' \) be any corresponding subterms in \( s \) and \( t \) that are not in replaced subterms. Then \( IE_S(s', s) \Rightarrow IE_S(t', t) \).

Proof By induction on the length of \( s \). If \( t \) and \( s \) are not \( S \)-redexes, then the lemma follows easily from Lemma 3.4 and the induction assumption. So suppose that \( t = S x_1 \ldots x_m e_1 \ldots e_m e_0 \), \( s = S x_1 \ldots x_m o_1 \ldots o_m o_0 \), \( s' \subseteq o_1 \), and \( IE_S(s', s) \). If \( IE_S(s', o_1) \), then by the induction assumption \( IE_S(t', e_1) \) and hence \( IE_S(t', t) \). Otherwise, by Lemma 3.5, we have \( IE_S(o_1, s) \). Hence, by Lemma 3.4, \( IE_S(s; o_1) \). Thus, by Lemma 3.3, \( x_i \not\in EFV_S(o_1) \). Let us show that \( x_i \not\in EFV_S(e_0) \).
By the induction assumption, if \( z_i \) has an \( S \)-essential occurrence in \( e_0 \) outside of replaced subterms, then the corresponding occurrence of \( z_i \) in \( \alpha_0 \) is \( S \)-essential. If \( z_j \) has an \( S \)-essential occurrence in a subterm \( t_j \subseteq e_0 \), then by Lemma 3.5, \( ES_S(t_j, \alpha_0) \). By the induction assumption, \( ES_S(s_j, \alpha_0) \). Since \( t_j \prec_S s_j \), \( z_i \) has a free occurrence in \( s_j \). Since \( s_j \) does not contain \( S \)-symbols, it follows from Lemma 3.4 that this occurrence is \( S \)-essential in \( s_j \) and hence, by Lemma 3.5 and \( ES_S(s_j, \alpha_0) \), is \( S \)-essential in \( \alpha_0 \). Hence \( x_i \notin EFV_S(e_0) \) and, by Lemma 3.3, \( IE_S(t; e_i) \). Therefore, by Definition 3.5 and Lemma 3.4, \( IE_S(t; t') \) and \( IE_S(t'; t) \).

### 3.2 The Essential Similarity Lemma

In this subsection we only consider HRPSs.

**Definition 3.6** Let \( t = (t_1/s_1, \ldots, t_k/s_n) s \) and let \( P : s = e_0 \xrightarrow{u_0} c_1 \xrightarrow{u_1} \ldots \in R_f \). Let us construct the reduction \( P(||(t)) : t = e_0 \xrightarrow{u_0} c_1 \xrightarrow{u_1} \ldots \in R_f \) as follows. If \( u_0 \) is outside \( s_1, \ldots, s_n \), then \( u_0 \) is its corresponding redex in \( \alpha_0 \). Otherwise \( u_0 = \emptyset \). Since \( c_1 \) is obtained from \( c_1 \) by replacing the descendants of \( s_1, \ldots, s_n \) with \( t_1, \ldots, t_n \), respectively, in \( c_1 \) we can choose the redex \( v_1 \) analogously, and so on.

**Lemma 3.7** Let \( t = (t_1/s_1, \ldots, t_n/s_n) s \), let \( P : s \rightarrow s' \) in \( R_f \), and let \( Q = P(||(t)) : t \rightarrow t' \). Then \( t' \) can be obtained from \( s' \) by replacing the descendants of \( s_1, \ldots, s_n \) with \( t_1, \ldots, t_n \), respectively, and if \( e \) and \( o \) are corresponding subterms in \( s \) and \( t \), then the \( P \)-descendants and \( Q \)-descendants of \( e \) and \( o \) are the corresponding subterms of \( s' \) and \( t' \).

**Proof** By induction on \( |P| \).

**Notation** Recall that, if \( F \) is a set of \( R \)-redexes in a term \( t \), then a complete \( F \)-development of \( t \) is a reduction that contracts residuals of redexes from \( F \) as long as possible. Below \( F_R \) denotes a complete \( F \)-development in \( F \), \( F_f \) denotes a complete \( F \)-development in \( R_f \), and \( F_S \) denotes a complete development of the set of \( S \)-redexes created during \( F_f \). If \( F \) consists of one redex \( u \) only, then we write \( u_R, u_f \), and \( u_S \) for \( F_R, F_f \), and \( F_S \), respectively.

**Lemma 3.8** Let \( F \) be a set of \( R \)-redexes in a term \( t \). Then \( F_R \approx_s t F_f + F_S \).

**Proof** By induction on number \( n \) of redexes in \( F \). The case \( n = 0 \) is trivial. So let \( F = F' \cup \{ u \} \), where \( u \) is an outermost among redexes in \( F \). Then \( F_R \approx_s F'_R + u_R' \) (where \( u' = u/F_R \)) \( \approx_s \) by the induction assumption\( F'_f + F'_S + u'_R \approx_s t F'_f + u'_R + F'_S/u'_R \) (where \( u'' \) is the unique ancestor of \( u' \)) \( \approx_s \) \( F'_f + u'' + F'_S/u'' + F'_S/u'_R \approx_s t F'_f + u'' + F'_S/u_R \approx_s t F'_f + F'_S/u_R \).

![Diagram](image_url)

**Lemma 3.9** (Representation Lemma) Let \( R \) be an HRPS.

1. For any \( R \)-redaction \( P : t \rightarrow_R s \) there are \( Q : s \rightarrow_R e \) and \( P' : t \rightarrow_{R_f} o \) such that \( P + Q \approx_s P' + P'' \), where \( P'' \) is a normalizing \( S \)-reduction starting from \( o \).

2. If \( P_f : t \rightarrow_{R_f} s \) and \( P_S \) is a normalizing \( S \)-reduction starting from \( s \), then there is an \( R \)-reduction \( P_R \) such that \( P_f + P_S \approx_{st} P_R \).
3. Similarity of redexes

Proof (1) By induction on $|P|$. Let $P = P^* + u$. By the induction assumption, there are $R$-reduction $Q^*$, $R_f$-reduction $P_1$, and a normalizing $S$-reduction $Q_1$ such that $P^* + Q^* \approx_{st} P_1 + Q_1$. Let $F$ be the set of all $Q^*$-residuals of $u$ and let $F'$ be the set of their $Q_1$-ancestors for which redexes from $F$ are residuals ($S$-reduction steps do not create new redexes). Then, $F'_R + Q_1/F'_R \approx_{st} Q_1 + F'_R$. By Lemma 3.8, $F'_R = F'_S + F'_R$. Thus $P_1 + P'_f + F'_S + Q_1/F'_R \approx_{st} P_1 + Q_1 + F'_R \approx_{st} P^* + Q^* + F'_R \approx_{st} P^* + u + Q^*/u = P + Q^*/u$ and we can take $Q = Q^*/u$, $P' = P_1 + F'_f$, and $P'' = F'_S + Q_1/F'_R$.

(2) By induction on $|P_f|$. The case $|P_f| = 1$ is immediate. So let $P_f : t \to e^u s$ and let $P'_f : t \to e$ be the initial part of $P_f$. By the induction assumption, there is an $R$-reduction $P_R$ such that $P'_R \approx_{st} P'_f + F'_S$. Hence $P_f + F_S \approx_{st} P_f + F'_S = (P'_f + F'_S + F_f + F_S)$ (where $F$ is the set of residuals of $u$ under $P'_S$) $\approx_{st} (P'_R + F_f + F_S) \approx_{st} (P'_R + F_R)$, so we can take $P_R = P'_R + F_R$.

Remark 3.1 It is easy to see that the reduction $Q$ in the Representation Lemma contracts the redexes that belong to the “families” [14] of redexes contracted by $P$.

Corollary 3.1 (1) (Erasure Lemma) Let $R$ be an HRPS. Then $IE_R(s, t)$ iff there is an $R_f$-reduction $P : t \to e$ such that all $P$-descendants of $s$ in $t'$ are $S$-inefficient.

(2) Let $t \to e$ in $R_f$. If $ES_R(s, t)$, then $s$ has an $R$-essential descendant in $e$.

Lemma 3.10 Let $s = (s_1/t_1, \ldots, s_n/t_n)t$, let $t \leftarrow_R s$, and let $s'$ and $t'$ be any corresponding subterms of $s$ and $t$ that are outside of replaced subterms. Then $IE_R(s', s) \Rightarrow IE_R(t', t)$.

Proof Let $IE_R(s', s)$. By the Erasure Lemma, there is an $R_f$-reduction $P : s \to e$ such that all $P$-descendants of $s'$ are $S$-inefficient. Let $P|(t) : t \to o$ in $R_f$. Further le: $s_1^i, \ldots, s_k^i$ be the enumeration of all $P$-descendants of $s_i$ from left to right and let $t_1^i, \ldots, t_k^i$ be the enumeration of all $P|(t)$-descendants of $t_i$ from left to right ($i = 1, \ldots, n$). By Lemma 3.1, $o$ is obtained from $e$ by
replacing $s^i_j$ with $t^i_j$ (subterms $s^i_j$ and $t^i_j$ are disjoint, $t^i_j = t_i$, and $s_i \rightarrow s^i_j$). By Lemma 3.1, there are $R$-reductions $s^i_j \rightarrow s^{i+1}_j$ and $t^i_j \rightarrow t^{i+1}_j$ such that none of $R$-inessential free variables of $s^i_j$ and $t^i_j$ that are bound by (outside) $S$-operators have descendants in $s^{i+1}_j$ and $t^{i+1}_j$. Let $e'$ and $o'$ be obtained from $e$ and $o$ by replacing $s^i_j$ and $t^i_j$ with $s^{i+1}_j$ and $t^{i+1}_j$, respectively. Since $t \ll_R s$, it follows from the conditions (a)-(b) in the Definition 2.2 that $o \ll_{R[S,R]} e$; in particular, $o \ll_{S[R]} e$. Hence, it follows from $EBV_{S,R}(s^i_j) = EBV_{S,R}(s^{i+1}_j) = BV_{S}(s^{i+1}_j)$ and $EBV_{S,R}(t^i_j) = EBV_{S,R}(t^{i+1}_j) = BV_{S}(t^{i+1}_j)$ that $o' \ll_{S,R} e'$, i.e., $o' \ll_{S} e'$. Since $BV_{S}(s^{i+1}_j) \subseteq BV_{S}(s^i_j)$, we have also that $e' \ll_{S} e'$. Thus, by the Replacement Lemma, all descendants of $s'$ in $e'$ are $S$-inessential and hence all descendants of $t'$ in $o'$ are $S$-inessential. Hence, again by the Erasure Lemma, $IE_R(t', t)$.

**Corollary 3.2 (Essential Similarity Lemma)** Let $s = (s_1//t_1, \ldots, s_n//t_n)t$, $t \approx_R s$, and let $s'$ and $t'$ be any corresponding subterms of $s$ and $t$. Then $IE_R(s', s)$ if and only if $IE_R(t', t)$.

**Corollary 3.3** Corresponding arguments of essentially similar redexes are either both essential or both inessential.

### 3.3 The Uniform Generation Lemma

In this subsection we prove the Uniform Generation Lemma; the Essential Generation Lemma and the Generation Lemma are its immediate corollaries. The lemma establishes relation between similarity of redexes and generation of redexes in HRPSs.

**Lemma 3.11** Let $t \overset{w}{\rightarrow} s$ in $R$ and let $t \subseteq s$ be a residual of a redex $u \subseteq t$. Then $u \approx_R v$.

**Proof** It is easy to see that if $u \subseteq w$, then $v = w \theta_1 \ldots \theta_n$ for some substitution $\theta_1 \ldots \theta_n$ such that $s_i = a_i \theta_i$ for some substitution $\theta_i$, then $s_i \subseteq a_i \theta_i$ for some substitution $\theta_i$, and the descendants of $e_0$ in $s_i$ and $a_0$ are corresponding occurrences. Hence, it follows from the Essential Similarity Lemma that in this case $u \approx_R v$. If $u$ and $w$ do not overlap, then $u = v$. Otherwise, if $w \subseteq u$, then $u \approx_R v$ follows from Lemma 3.2.

**Lemma 3.12** Let $e_0 \subseteq s_0 \subseteq t_0$, $P : t_0 \overset{\eta_0}{\rightarrow} t_1 \overset{\eta_1}{\rightarrow} \ldots \rightarrow t_n$, and let $s_{i+1} \subseteq t_{i+1}$ be a $u_i$-descendant of $s_i \subseteq t_i$ ($i = 0, \ldots, n - 1$). Then there is a reduction $Q : t_0 \overset{\eta_0}{\rightarrow} t_1 \overset{\eta_1}{\rightarrow} \cdots \rightarrow t_n$ such that $a_i = a_i \theta_i$ and the descendants of $e_0$ in $s_i$ and $a_i$ are corresponding occurrences.

**Proof** By induction on $|P|$. The case $|P| = 0$ is obvious. Suppose that $a_i \theta_k = s_k$. If $u_k$ and $s_k$ do not overlap, then we take $v_k = \theta_k$ and $\theta_k = \theta_{k+1}$ (since $s_k = s_{k+1}$). If $u_k$ is in a substituted subterm of $s_k$, then again we take $v_k = \theta_k$, and take a corresponding substitution $\theta_{k+1}$. If $u_k$ contains $s_k$, then again $v_k = \theta_k$ and $\theta_{k+1} = \theta_k \theta$ for some substitution $\theta$. Finally, if $u_k \subseteq s_k$ and is outside of the substituted subterms, then we take as $v_k$ the corresponding redex of $u_k$ in $a_k$, and take $\theta_k = \theta_{k+1}$.

**Lemma 3.13** (1) Each outermost redex in a term $t$ is essential.

(2) $ES(e, t)$ iff $ES(s, t)$ and $ES(e, s)$.

(3) $IE(e, t)$ iff $e$ is in an inessential argument of an essential redex $u$ of $t$.

**Proof** (1) From Definition 3.2.

(2) ($\Rightarrow$) From Definition 3.2. ($\Leftarrow$) By Lemma 3.12, $IE(e, t)$ and $ES(s, t)$ imply $IE(e, s)$, a contradiction.

(3) ($\Rightarrow$) Since $IE(e, t)$, $e$ is inside an outermost, hence essential, redex. Let $u$ be the innermost essential redex containing $e$ in an argument $s$. Then $ES(s, u)$ imply by (2) that $IE(e, s)$, and the outermost redex of $s$ that contains $e$ would be essential in $t$ by (2), contrary to the choice of $u$. ($\Leftarrow$) By (2).
Lemma 3.14 (Uniform Generation Lemma) Let $R$ be an HRPS, $R'$ be a subsystem of $R$, $u$ and $v$ be $R'$-redexes, $u \xrightarrow{R'} v, u \xrightarrow{R} o$, and $v \xrightarrow{R} e$. If there is an $R'$-essential $R$-redex $w$ in $o$ created by $u$, then $e$ contains an $R'$-essential $R$-redex $w'$, created by $v$, such that $w \xrightarrow{R} w'$.

Proof Let $o'$ and $e'$ be contracta of $u$ and $v$ in $R'$. Then $e'$ is obtained from $o'$ by replacing the descendants of arguments of $u$ with corresponding descendants of arguments of $v$. Since the replaced subterms do not contain $S$-redexes (and $R'$-essentiality coincides with $R' \cup S$-essentiality), it follows from conditions (a)-(b) of Definition 2.2 that $o' \xrightarrow{S \cup R} e'$. Let $w_1$ be the ancestor of $w$ in $o'$ and $w_2$ be the corresponding redex of $w_1$ in $e'$. By Lemma 3.10 and $o' \xrightarrow{S \cup R} e'$, $ES_{S \cup R}(w_1, o')$ implies that $ES_{S \cup R}(w_2, e')$. Therefore, $w_2$ has an $R'$-essential residual $w'$ in $e'$ (where again $R'$-essentiality coincides with $R' \cup S$-essentiality).

Let $w_1 = \sigma z_1 \ldots z_k a_1 \ldots a_m$ and $w_2 = \sigma y_1 \ldots y_k e_1 \ldots e_m$. Let $o_j$ be in the scope of $\sigma$ and $x_i'$ be an occurrence of $x_i$ in $o_j$ that is $R' \cup S$-essential in $o_j$. If $x_i'$ is outside the replaced subterms, then its corresponding occurrence of $y_i$ in $e_j$ is $R' \cup S$-essential in $e_j$ by Lemma 3.10. If $x_i'$ is in a replaced subterm $t^*$, then, by Lemma 3.13, $t^*$ is $R' \cup S$-essential in $o_j$, its corresponding replaced subterm $s^*$ is $R' \cup S$-essential in $e_j$ by Lemma 3.10, and, again by Lemma 3.13, each occurrence of $y_i$ in $s^*$ that is $R' \cup S$-essential in $s^*$ (there exists one because $o' \xrightarrow{S \cup R} e'$) is $R' \cup S$-essential in $e_j$ as well. Thus $w_1 \xrightarrow{S \cup R} w_2$, Hence, by Lemma 3.11, $w \xrightarrow{R} w'$ (since, in $e$ and $o$, $R'$-essentiality coincides with $R' \cup S$-essentiality).

Corollary 3.4 (Essential Similarity Lemma) Let $R$ be an HRPS, $u \xrightarrow{S \cup R} v, u \xrightarrow{R} o$, and $v \xrightarrow{R} e$. Then there is an $R$-essential redex $w$ in $o$ created by $u$ if and there is an $R$-essential redex in $e$. $R$-essentially similar to $w$, created by $v$.

Remark 3.2 If $s = (s_1/t_1, \ldots, s_n/t_n)t$ and $s \approx t$, then it follows from the Essential Similarity Lemma that corresponding redexes of $s$ and $t$ (that are outside of the replaced subterms) are essentially similar (when replaced subterms are specified to be the arguments of the redexes). The converse is not true in general: consider the redexes $u = \sigma x(g(y, x))$ and $v = \sigma x(g(x, x))$ in a system $R = \{\sigma a A \rightarrow \ldots\}$; then $u \not\xrightarrow{R} v$ when $v = (x/y)u$, while $u \xrightarrow{S \cup R} v$. However, it follows easily from Lemma 3.13 and Corollary 3.2 that the Essential Similarity Lemma remains valid if the condition $t \approx s$ is replaced by a weaker condition that, for any corresponding redexes $w$ and $w'$ in $t$ and $s$, $w \approx w'$. Similar remark is valid for Lemma 3.10.

4. DECIDABILITY OF WEAK NORMALIZATION IN HRPSs

In this section we further study (only) HRPSs. In particular, we construct a decision algorithm for weak normalization. The following definition is crucial; its correctness follows from Corollaries 3.3 and 3.4.

Definition 4.1 (1) We call a sequence of EC-rules $(r_0, ECS(r_0)), (r_1, ECS(r_1)), \ldots$ an essential $(r_0, ECS(r_0))$-chain if an essential $(r_{i+1}, ECS(r_{i+1}))$-redex is created by contraction of any $(r_i, ECS(r_i))$-redex. For any $(r_0, ECS(r_0))$-redex $u$, we also call an essential $(r_0, ECS(r_0))$-chain an essential $u$-chain.

(2) Let $(r, ECS(r))$ be an EC-rule and $u$ be an $(r, ECS(r))$-redex. We call the sequence of numbers of essential arguments of $u$ the essentiality indicator of $(r, ECS(r))$ and of $u$.

Lemma 4.1 A term $t$ is weakly normalizable if any essential chain of any essential redex in $t$ is finite.

Proof ($\Rightarrow$) Consider an infinite essential chain $(r_0, ECS(r_0)), (r_1, ECS(r_1)), \ldots$ of an essential redex in $t$. We show by induction on $i$ that $(\alpha_i)$: for any reduction $P : t \approx t_0 \approx t_1 \approx \ldots$ and for
each \( i < |P| \) there is a \( k_i \) such that \( t_i \) contains an essential \((r_{k_i}, ECS(r_{k_i}))\)-redex. \((\alpha)_0\) is obvious. If \( u_i \) is an essential \((r_{k_i}, ECS(r_{k_i}))\)-redex, then it follows from Lemmas 3.2 and 3.13.(2) that \( t_{i+1} \) contains an essential \((r_{k_i+1}, ECS(r_{k_i+1}))\)-redex. Otherwise, by Lemmas 3.2 and 3.11, any essential \((r_{k_i}, ECS(r_{k_i}))\)-redex in \( t_i \) has an essentially similar essential residual in \( t_{i+1} \). Thus \((\alpha)_i\) holds for all \( i \), and \( t \) is not normalizable. \((\Leftarrow\Rightarrow\) For any \((r, ECS(r))\)-redex \( u \), let the weight of \( u \) be the length of a longest essential \((r, ECS(r))\)-chain. Since the contraction of any essential redex creates essential redexes only with smaller weights, a reduction that in each step contracts an innermost redex among essential ones with maximal weight terminates in a normal form.

**Lemma 4.2** Let \( t \) be a term in an HRPS \( R \) and essentiality indicators of all \( EC \)-rules in \( R \) be known. Then one can find all inessential subterms in \( t \) using the following

**Algorithm 4.1** Choose in \( t \) an innermost redex; find its essential characteristic system (which coincides with its characteristic system); underline its inessential arguments; and mark the redex itself. Now choose in \( t \) an unmarked redex that only contains marked redexes; find its essential characteristic system (only occurrences that are in the underlined subterms are inessential); underline its inessential arguments; mark the redex itself; and so on, as long as possible. Then exactly occurrences that are in underlined subterms are inessential in \( t \).

**Proof** From Lemma 3.13.

**Definition 4.2** Let \((r, ECS(r))\) be an \( EC \)-rule, let \( r : t = \alpha_1 \ldots \alpha_n A_1 \ldots A_m \rightarrow B\), and let \( \theta \) be an admissible assignment such that \( A_i \theta \) is in \( R \)-normal form and \( ECS(t\theta) = ECS(r) \). Then we call \( r\theta : t\theta \rightarrow s\theta \) a trivial \( ECS(r)\)-instance of \( r \).

**Lemma 4.3** Let \((r_1, ECS(r_1)), \ldots, (r_l, ECS(r_l))\) be all \( EC \)-rules in an HRPS \( R \) and \( r_i \theta_i : t_i \theta_i \rightarrow s_i \theta_i \) be a trivial \( ECS(r_i)\)-instance of \( r_i \) : \( t_i \rightarrow s_i \) \((i = 1, \ldots, l)\). Then the essentiality indicators of the \( EC \)-rules in \( R \) can be found using the following

**Algorithm 4.2** Find, for each \( i \), all arguments of \( t_i \theta_i \) that do not have descendants in \( s_i \theta_i \). Let the corresponding arguments of any \((r_i, ECS(r_i))\)-redex be 0-inessential. Let the 0-essentiality indicator of \((r_i, ECS(r_i))\) be the list of numbers of arguments of any \((r_i, ECS(r_i))\)-redex that are not 0-inessential. Apply Algorithm 4.1 to the right-hand sides of all \( EC \)-rules of \( R \) using 0-essentiality indicators of \( EC \)-rules instead of essentiality indicators. Let the 1-inessential arguments of \( t_i \theta_i \) be all arguments whose descendants in \( s_i \theta_i \) are in underlined subterms \((i = 1, \ldots, l)\). Let the corresponding arguments of any \((r_i, ECS(r_i))\)-redex be 1-inessential, and let the 1-essentiality indicator of \((r_i, ECS(r_i))\) be the list of numbers of arguments of any \((r_i, ECS(r_i))\)-redex that are not 1-inessential. Apply again Algorithm 4.1 to the right-hand sides of all \( EC \)-rules of \( R \) using 1-essentiality indicators of \( EC \)-rules instead of essentiality indicators; and so on. The algorithm stops after \( n_0 \) steps if \( n_0 \)-essentiality indicator of each \( EC \)-rule in \( R \) coincides with its \((n_0 - 1)\)-essentiality indicator. Let the \( n_0 \)-inessential arguments of \( R \)-redexes be inessential*. Then the essentiality indicator of \((r_i, ECS(r_i))\) coincides with the list of non-inessential* arguments of \( t_i \theta_i \).

**Proof** An easy induction on \( n \) shows that \( n \)-inessential arguments of each redex are inessential; the case \( n = 0 \) follows from Corollary 3.3, and the induction step follows from Lemmas 4.2 and 3.2. Now let us prove by induction on \( k \) that if, for an \( EC \)-rule \((r_i, ECS(r_i))\), there is an \((r_i, ECS(r_i))\)-redex \( u \) whose \( j \)-th argument \( s_j \) (is inessential and) does not have descendants under some reduction \( P \) with a length \( |P| \leq k \), then the \( j \)-th argument \( s_j \) of \( t_i \theta_i \) is inessential*. The case \( k = 1 \) is obvious. So suppose \( P : u \rightarrow t \rightarrow s \rightarrow e \). Let \( s'_j \) be a descendant of \( s_j \) in \( o \), and let \( u \) be the minimal redex that
contains $s_j'$ and has a descendant in $e$. Suppose that $v$ is an $(r_m, E\text{CS}(r_m))$-redex and $s_j'$ is in its $p$-th argument. By Lemma 3.12, there is a reduction $Q$ starting from $v$ with a length less than $k$, such that the $p$-th argument of $Q$ does not have $Q$-descendants. Hence, by the induction assumption, the $p$-th argument of $t\theta_m$ is inessential*. Similarly, any descendant of $s_j$ in $o$ is an inessential* argument of some redex. Since $CS(t\theta_i) = E\text{CS}(t\theta_i) = E\text{CS}(u) \subseteq CS(u)$, for any descendant $e_j'$ of $e_j$ there is a descendant $s_j'$ of $s_j$ in $o$ that is not in an inessential argument of a non-created redex and such that, for any $q = 1, \ldots, l$, $e_j'$ is in an argument of an $(r_q, E\text{CS}(r_q))$-redex if $s_j'$ is in the corresponding argument of a created $(r_q, E\text{CS}(r_q))$-redex. Hence, any descendant of $e_j$ in $s\theta_i$ is in an inessential* argument of some redex, and $e_j$ itself is inessential*.

Algorithm 4.2 can be illustrated by the following example.

**Example 4.1** Let us consider an HRPS $R$ with the following rules:

$$\sigma \alpha(A, B) \rightarrow (\epsilon\alpha A/a)B$$

$$\delta a A \rightarrow \sigma A, f(A)\)$$

$$f(A) \rightarrow g(A, A)$$

$$g(A, B) \rightarrow \text{const}$$

where $\alpha$ is an object metavariable, $A, B$ are term metavariables, $f, g$ are function symbols, $\sigma, \epsilon, \delta$ are quantifier signs with arities $(1, 2), (1, 1), (1, 1)$, and binding scopes $(1, 2), (1, 1)$, and $(1, 1)$, respectively. The rule for $\sigma$ has four trivial $E\text{CS}$-instances $r_1 : \sigma x(x, x) \rightarrow (\epsilon x x/x)x = \epsilon x x$, $r_2 : \sigma x(x, y) \rightarrow (\epsilon x x/x)y = y$, $r_3 : \sigma x(y, x) \rightarrow (\epsilon x y/x)x = \epsilon x y$, and $r_4 : \sigma x(y, y) \rightarrow (\epsilon x y/x)y = y$ with $E\text{CS}s \{(a, A), (a, B)\}, \{(a, A)\}, \{(a, B)\},$ and $\{\}$, respectively. Similarly, we can choose trivial $E\text{CS}$-instances of the rules for $\delta, f,$ and $g$: $r_5 : \delta x x \rightarrow \sigma x(x, f(x))$, $r_6 : \delta x y \rightarrow \sigma x(y, f(y))$, $r_7 : f(x) \rightarrow g(x, x)$, and $r_8 : g(x, y) \rightarrow \text{const}$. The result of Algorithm 4.2 is then as follows:

$$r_1 : \sigma x(x, x) \rightarrow (\epsilon x x/x)x = \epsilon x x$$

$$r_2 : \sigma x(\underbrace{x, y}_{0}) \rightarrow (\epsilon x x/x)y = y$$

$$r_3 : \sigma x(y, x) \rightarrow (\epsilon x y/x)x = \epsilon x y$$

$$r_4 : \sigma x(\underbrace{y, y}_{0}) \rightarrow (\epsilon x y/x)y = y$$

$$r_5 : \delta x \underbrace{x}_{0} \rightarrow \sigma x(x_{1,3}, f(x_{2,3}))$$

$$r_6 : \delta x \underbrace{y}_{2} \rightarrow \sigma x(y_{1,2,3}, f(y_{2,3}))$$

$$r_7 : f(\underbrace{x}_{1}) \rightarrow g(x_{1,2,3}, x_{1,2,3})$$

$$r_8 : g(\underbrace{x}_{0}, \underbrace{y}_{0}) \rightarrow \text{const}$$

where $\underbrace{i}$ indicates that corresponding subterms are $i$-inessential and subscripts $j$ in underlined subterms indicate that the underlining was made while running Algorithm 4.1 for the $j$-th time. The 3-essentially indicators of all the $E\text{C}$-rules in $R$ coincide with their 2-essentiality indicators, so
it is enough to run Algorithm 4.1 three times. Note that, for example, the \( ECS \) of the right-hand side \( cxx(x, f(x)) \) of \( r_5 \) used while running Algorithm 4.1 for the second time is a proper subset of \( ECS \) of \( px'(x, f(x)) \) used while running Algorithm 4.1 for the first time, and this makes it possible to determine that the argument \( x \) of the left-hand side \( \delta xx \) of \( r_5 \) is 2-inessential, and hence inessential. By Lemma 4.3, essentiality indicators of \( EC \)-rules in \( R \) coincide with the numbers of non-overbraced arguments of corresponding trivial instances.

**Theorem 4.1** Weak normalization is decidable in HRPs.

**Proof** From Lemmas 4.1, 4.2, and 4.3.

**Corollary 4.1** HRPs do not have full computational power.

**Proof** The existence of an interpreter for, say \( \lambda \)-calculus, in an HRP would imply decidability of weak normalization for \( \lambda \)-terms, which is not valid [5].

5. **Persistent systems**

Without restricting the class of OCRSs we can assume that in right-hand sides of rewrite rules the last argument of each metasubstitution is a term-metavari­able or a metasubstitution. For example, we can replace the metasubstitution \( f((B/a)g(A)) \) by the equivalent metasubstitution \( f(g((B/a)A)) \). Using this convention, we can define the following persistent systems.

**Definition 5.1** Let \( R \) be an orthogonal CRS.

1. Let \( t \rightarrow s \), let \( t \rightarrow t' \rightarrow s \) be its expansion, let \( o \subseteq t' \) be the contractum of \( u \) in \( R_f \), and let \( v \) be a new redex in \( s \). We call \( v \) **generated** if \( v \) is a residual of a redex \( w \) of \( t' \) whose pattern is in the pattern of \( o \). We call \( v \) **uniformly generated** if the pattern of \( w \) is in the pattern of \( o \) and is not inside an \( S \)-redex. We call \( v \) **quasi-generated** if any of its pattern-subterms is a descendant of a pattern-subterm of \( o \) that is not an \( S \)-redex.

2. We call \( R \) respectively **persistent** (PCRS), **uniformly persistent**, or **quasi-persistent** if, for any \( R \)-reduction step, each created redex is generated, uniformly generated, or quasi-generated.

3. We call \( R \) **strongly persistent** if \( R_{fs} \) is persistent.

A quasi-persistent CRS \( R = \{r_1 : aaA \rightarrow f((d/a)A), r_2 : f(d) \rightarrow c\} \) that is not persistent was considered in the introduction. The CRS \( R_1 = \{\exists aA \rightarrow g((\tau aA/a)A), f(\tau ac) \rightarrow d\} \), where \( c \) and \( d \) are constants, is an example of persistent system (no redex creation is possible in \( R_1 \)) that is not strongly persistent.

The following is a characterization of HRPs in terms of redex creation.

**Proposition 5.1** A non-simple OCRS \( R \) is strongly persistent iff it is an HRP.

**Proof** If \( R \) is not simple, then \( R_{fs} \) contains \( S \)-rules. If \( R \) is not an HRP, then there is a redex containing at least two function symbols in its pattern, and it can be created during an \( S \)-step. (For example, if \( u = f(g(a)) \) is a redex with \( f, g \) in the pattern, then \( Sx g(a) f(x) \rightarrow u \).) Hence any non-simple strongly persistent CRS must be an HRP. The converse is obvious.

We call a subterm \( s \) in \( t \) **free** if \( s \) is not a proper pattern-subterm of a redex in \( t \). It is easy to see that all free subterms remain free under any reduction in PCRSs. This fails already for quasi-persistent systems: in the quasi-persistent CRS \( R = \{r_1 : aaA \rightarrow f((d/a)A), r_2 : f(d) \rightarrow c\} \), the argument \( x \) is free in the redex \( \sigma xx \), while its descendant \( d \) under the contraction of \( \sigma xx \) is a pattern-subterm of the created redex \( f(d) \). All properties of similar and essentially similar terms in HRPs are valid because all subterms are free in HRPs. Therefore, it is not difficult to check that this properties remain valid for all PCRSs if in the definition of similarity and essential
similarity one requires the replaced subterms to be free. Since in PCRSs arguments of all redexes are free, one can analogously define essentiality indicators and essential chains for EC-rules in PCRSs. Algorithms 4.1 and 4.2 work also in PCRSs. Hence we have the following theorem.

**Theorem 5.1** Weak normalization is decidable in persistent CRs.

**Lemma 5.1** Let $R$ be a persistent CRS, $u \xrightarrow{r} v$, $u \xrightarrow{u} o$, and $v \xrightarrow{v} e$. If there is a redex $w$ in $o$ generated by $u$, then $e$ contains a redex $w'$, generated by $v$, such that $w \xrightarrow{r} w'$.

**Proof** The proof of Lemma 3.14 remains valid (after some minor changes) for the case of persistent CRs. Hence it is enough to take $R' = \emptyset$.

**Corollary 5.1** (Generation Lemma) Let $u$ and $v$ be similar redexes in a persistent CRS. Then $u$ and $v$ generate similar redexes.

**Definition 5.2** We call a sequence of C-rules $(r_0, CS(r_0)), (r_1, CS(r_1)), \ldots$ an $(r_0, CS(r_0))$-chain if an $(r_{i+1}, CS(r_{i+1}))$-redex is generated by contraction of any $(r_i, CS(r_i))$-redex $(i = 0, 1, \ldots)$. For any $(r_0, CS(r_0))$-redex $u$, we also call an $(r_0, CS(r_0))$-chain a $u$-chain.

**Lemma 5.2** A term $t$ in a PCRS $R$ is strongly normalizable iff all chains of redexes in $t$ are finite.

**Proof** ($\Rightarrow$) Immediate. ($\Leftarrow$) Let $P : t = t_0 \xrightarrow{u_1} t_1 \xrightarrow{u_2} \ldots$ be an infinite reduction. Let us call a redex $u_i \subseteq t_n$ a son of $u_0 \subseteq t_0$ if there are redexes $u_i \subseteq t_i$ $(i = 1, \ldots, n - 1)$ such that either $u_{i+1}$ is a residual of $u_i$ or $u_i = u_i'$ and $u_i'$ is generated by $v_i$. Since in $P$ only redexes from $t$ and their sons are contracted, there is at least one redex $u_0 \subseteq t_0$ such that infinitely many sons of $u_0$ are contracted in $P$. Let $t_0$ be a residual of $u_0$ that is contracted in $P$ and infinitely many sons of which are contracted in $P$. Then $t_0$ generates a redex $u_1$ whose infinitely many sons are contracted in $P$. By analogy, $u_2$ has a residual $v_1$ contracted in $P$ that generates a redex $u_2$ whose infinitely many sons are contracted in $P$, and so on. Obviously, $t_0 \xrightarrow{r} u_0$. Thus by Lemma 5.1, $u_0$ generates a redex $u_1$ such that $u_1 \xrightarrow{r} u_1$. Further, $u_1 \xrightarrow{r} u_2$. Thus, again by Lemma 5.1, $u_1$ generates a redex $u_2$ such that $u_2 \xrightarrow{r} u_2$. By analogy we can show that $u_2$ generates a redex $u_3$ such that $u_3 \xrightarrow{r} u_3$, and so on. Thus, $t_0$ has an infinite chain.

**Corollary 5.2** Strong normalization is decidable in persistent CRs.

Obviously, a PCRS $R$ is weakly (resp. strongly) normalizing iff all essential chains (resp. all chains) in $R$ are finite. Thus it is decidable whether $R$ is weakly (strongly) normalizing.

**Remark 5.1** Let $R$ be a persistent CRS and $R'$ be a subsystem of $R$. We call an $R$-redex $u$ $E_{CRS}$-complete if each binding variable of a quantifier belonging to the pattern of $u$ has an $R'$-essential free occurrence in all arguments that belong to its binding scope. It is easy to check that $E_{CRS}$-complete redexes generate $E_{CRS}$-complete redexes (this follows from conditions (a)-(b) of Definition 2.2 and Lemma 3.13). By Lemma 3.14, if $u$ and $v$ are weakly similar redexes such that $E_{CRS}((u) \subseteq E_{CRS}(v)$ and $u$ generates a redex $u'$, then $v$ generates a redex $v'$ such that $E_{CRS}(v') \subseteq E_{CRS}(u')$. Therefore, any $R$-essential chain (resp. any chain) of a redex $w$ is a subsequence of an $R$-essential chain (resp. chain) of an $E_{CRS}$-complete (resp. $E_{CRS}$-complete) redex that is weakly similar to $w$. Thus, in order to establish weak (strong) normalization of a persistent CRS, it is enough to check essential chains (chains) of $E_{CRS}$-complete ($E_{CRS}$-complete) rules only.
6. Optimal normalization in persistent CRSS

Theorem 6.1 Let t be a term in a uniformly persistent CRS. 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 \( P : t = t_0 \overset{s_0}{\Rightarrow} s_1 \overset{s_1}{\Rightarrow} \ldots \) be a normalizing reduction starting from \( t \) and \( Q : t = s_0 \overset{s_0}{\Rightarrow} s_1 \overset{s_1}{\Rightarrow} \ldots \) 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 \( ES(u_{n_i}, t_{n_i}) \), \( u_{n_i} \overset{r}{\Rightarrow} v_{n_i} \), and \( i \neq j \) implies \( n_i \neq n_j \).

1. Let \( i = 0 \). Since \( v_0 \) is essential in \( t \) and \( t_n \) is a normal form, it follows from Corollary 3.2 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 \( v_0 \). By Lemma 3.11, \( v_0 \overset{r}{\Rightarrow} u_{n_0} \).

2. Suppose that, for each \( i < m \), we have already defined \( n_i < n \) such that \( ES(u_{n_i}, t_{n_i}) \) and \( u_{n_i} \overset{r}{\Rightarrow} v_{n_i} \), and assume that \( i = m \). Consider two possible cases:

(a) \( u_m \) is a residual of a redex \( v' \subseteq t \).

It follows from \( ES(v_m, s_m) \) and Lemma 3.2 that \( ES(v', t) \). Thus, as before, there is a number \( n_m \) such that \( u_{n_m} \) is an essential residual of \( v' \), and \( u_{n_m} \overset{r}{\Rightarrow} u_{n_{m+1}} \) by Lemma 3.11. (b) There is a number \( k \) such that \( u_m \) is a residual of a redex \( v^* \subseteq s_k \) generated by \( k \). By Lemma 3.2, \( ES(v_m, s) \) implies \( ES(v^*, s_k) \). By the induction assumption, \( v_{n_{k+1}} \) is essential and \( u_{n_{k+1}} \overset{r}{\Rightarrow} v_{n_{k+1}} \). Let \( l = n_{k+1} + 1 \). By Corollary 3.4, there is an essential redex \( w^* \subseteq t_l \) generated by \( u_{n_{k+1}} \) that is essentially similar to \( v^* \). 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 \( ES(u_{n_m}, t_{n_m}) \), and by Lemma 3.11, \( u_{n_m} \overset{r}{\Rightarrow} v_{n_m} \). 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.

Thus one can construct optimal sequential normalizing reductions in uniformly persistent CRSS. For PCRSs in general, sharing of redexes of "the same origin" [14] is necessary. For example, one can check that all normalizing reductions starting from a term \( sxf(x) \) in the following PCRS \( R = \{saA \rightarrow (saA/a)A, saA \rightarrow (raI/a)A, f(A) \rightarrow g(A, A)\} \) must contract two copies of at least one generated redex. Optimal implementation of PCRSs is possible in the framework of Interaction Systems [2].

7. Further work

We believe that the Essential Similarity Lemma can be generalized to the case of the \( \lambda \)-calculus. The replaced subterms must be independent: a subterm is independent if it is free and remains free under any reduction; moreover, redex creation is not possible inside an independent subterm during contraction of a redexes outside it. In this case, independent subterms are indeed independent in the sense that their computation can be made in parallel.

One can also define a class of constructor CRSS (pattern-matching definitions) that are persistent and enjoy the decidability of weak normalization.

Forthcoming papers will be devoted to a classification of OCRSS based on the patterns of redex creation and to a study of normalization and perpetuity behaviour in several subclasses. In this context, it is interesting to investigate whether inside creating systems, where each created redex is inside the contractum of the creating redex, have full computational power. In inside-creating systems, all outermost redexes are essential and independent, hence any outside-in strategy is normalizing and non-overlapping redexes can be computed independently.

Acknowledgments I would like to thank G. Gonthier, J.-J. Lévy, and I. Maranget for many suggestions, and B. Pierce and F. J. de Vries for careful reading.

References
REFERENCES


