A Generalized Normalizing Strategy

Riad S. Jabri*

ABSTRACT

Terms rewriting systems (TRS) has an important modeling role in many aspects of computing systems. However, there is a need for efficient, safe and terminating reductions of such systems. Hence, the normalization and the reductions of TRS have gained an increasing attention from a theoretical as well as from a practical point of view. In this work, we have formalized a generic normalization strategy, based on proposed functional concepts for TRS. Two frameworks, called GNFR1 and GNFR2, have been proposed for implementing such strategy. The computational results of GNFR1 and GNFR2 have been compared to the publicly available results of similar approaches. They have demonstrated their correctness and generality, in addition to their superiority in terms of reducing the cardinality of the rewriting sequences and the number of the terminating reductions. Furthermore, based on proposed functional concepts for positional matching and its aggregation into terms matching, the proposed frameworks can be applied to other than the constructor-based TRS and according to different reducing strategies.

Keywords: Terms, Rewriting systems, Normalization, Reduction, Generic, Matching.

1. INTRODUCTION

Term rewriting systems (TRS) consists of set of rewrite rules \( R = \{ R_i | R_i : l_i \rightarrow r_i \} \), where \( l_i \) and \( r_i \) are terms built from function symbols in the set of ranked alphabet \( \Sigma \) and from variables in the countable infinite universe of variables \( \nu \). Given a term \( t \), such rules can be applied to replace the subterms \( \sigma(l) \) of \( t \) by the respective right hand sides \( \sigma(r) \), where \( \sigma \) represents same substitutions for variables in \( l \) and \( r \). The replaced subterms are called redexes of \( t \) and the replacement process is called reduction. This process can be performed repeatedly until an irreducible term is obtained. Such term is called normal form (Teresa, 2003). As such TRS has an important modeling role in many aspects of software systems. Such as: pattern matching problems (Katoen and Nymeyer, 2000; Jay and Kesner, 2009), implementation and specification of programming languages (Antoy and Johnson, 2004; Ulidowski, 2009), program testing, program validation (Field and Tip, 1994; Lucanu, 2009), automated deduction (Feuillade et al., 2003), security and encryption (Genet and Klay, 2000; Bertolissi, 2008). However, there is a need for efficient, safe and terminating reductions of such systems (Gnaedig and Kirchner, 2009). Hence, the normalization and the reductions of TRS have gained an increasing attention from a theoretical as well from a practical point of view. This can be demonstrated by the continuous work over a span of more than 50 years, represented by the efforts concerning lambda calculus and a recent work, represented by the following examples. Applying tree automata for rewrite strategies (Rety, 1999; Rety, and Vuotto, 2005; Geser et al., 2005, 2007), studying the preservation of recognizability through rewriting (Durand, 2007), investigating decidable approximation of needed reduction strategies (Toyama, 2005) and minimizing the TRS reduction rules and proposing abstract machines for their simulation and efficient implementation (Fokkink, 1998; Visser, 2001).

Our research is a continuation of most recent efforts, especially the ones given in section (2). However, it is distinguished by proposing a generic and formal normalization approach with the following features:

- Identification of an initial and a related set of rewriting rules to be applied on a given term. Such a set is expanded in an incremental way. Thus, minimizing the number of rules to be applied and eliminating the redundant reductions.
- Representing the terms and the rewriting rules in
A Generalized Normalizing …

A Generalized Normalizing …

A Generalized Normalizing …

functional form that reflects their ranking and depths, and facilitates terms matching and rewriting in any order.

- Defining a generic and generalized matching criteria that ease restrictions on the class of the considered TRS and that is appropriate for different terms rewriting systems and reduction strategies
- Representing the term rewriting as a function that can be instantiated by a specific reduction strategy and specific matching criteria to rewrite the term at different positions accordingly.

Two alternative generic frameworks have been proposed to implement the proposed approach. The first one considers each one of the initial rules, and then parses the input term at the positions that are defined by the reducing strategy, with an objective to complete the matching process of the left hand side of the considered rules, As a result their right hand sides are obtained either as a final goal or as a subsequent goal for further expansion or reduction. The second framework considers the subsequent parsing positions and performs matching with respect to the left sides of all initial rules. Thus, the given term is scanned once for all rules.

The reminder of this paper is organized as follows: Section 2 presents an analysis and discussion of the related work. Basic definitions and preliminaries are given in section 3. Functional concepts for: terms; rewriting rules; matching criteria; positional and term matching; and positional and term rewriting. Based on such concepts, a generic normalizing strategy and two frameworks for its implementation are given in section 5 and section 6 respectively. Section7 presents some experimental results and finally a conclusion is given in section 8.

2. RELATED WORK

The most recent research that is related to our work can be classified and analyzed as follows:

A- Expressing the set of descendants R (E) according to different reducing strategies. This has been investigated by Rety (1999), Rety (2005) and Feuillade (2004) using finite tree automata and assuming a constructor–based rewrite system. The analysis of such efforts has shown the following:
- Although the proposed algorithms follow fixed strategy, it is possible to derive a generic one based on an abstract functional representation for the terms, rewriting rules, and based on defining a parameterized matching criterion that can be synchronized and instantiated by different reduction strategies.
- The data structure used to represent set of terms and the matching algorithm plays a crucial role in determining the efficiency of such algorithms. Hence there is a need to derive algorithms that implicitly simulates the tree automata rather than implementing it in explicit way.
- The efficiency of such algorithms can be improved by setting an objective to obtain a possible normal form and R (E) as intermediate results, rather than obtaining R (E) that includes the possible normal forms.

B- Minimizing the number of applied sequences of rewrite rules. This has been investigated by Nymeyer et al. (1997). The analysis of such work reveals the fact that a more efficient computation can be achieved by determining a set of initial rules that can be expanded during the matching and parsing process upon need and in incremental way.

C- Strategies provided by the existing rewriting engines such as Timbuk (Genet, 2001). Such engines follow fixed strategies and not generic ones.

D- Obtaining shortest normalizing derivations. This has been investigated as follows
- Nguyen (2001) and Fokkink (2000) have proposed a normalization procedure based on lazy rewriting. These efforts are compiled ones. Interpreted approaches with the same objectives are the represented by the work suggested by Fokkink (1998). Such work simplifies the rewrite rules. However it produces a large number of new ones. Furthermore, it changes the positions of redexes which make it difficult to synchronize parsing and normalizing derivations.

Based on the above analysis, we have set an objective to formalize the normalization process in a generic way and to minimize it respective rewriting sequences, as described in the following sections.

3. PRELIMINARIES

A term rewriting system TRS is a tuple ((Σ, R), V), where:
- Σ is a finite ranked alphabet (function symbols). The rank is a non negative integer number represents the number of arguments (arity) of the function symbols. Function symbols with 0-arity are called constants. V is a countable set of variables. A set of terms can be built from Σ and V. Such a set is denoted by T_Σ (V) and is
Defined as follows (Nymeyer et al., 1997):
- \( V \subseteq T_\Sigma(V) \Sigma_0 \)
- \( a \in \Sigma_0 \text{ and } t_1, \ldots, t_n \in T_\Sigma(V) \implies a(t_1, \ldots, t_n) \in T_\Sigma(V), n \geq 1 \)
- \( R \) is a set of rewrite rules \( \{ l \rightarrow r | (l, r) \in T_\Sigma(V) \} \), where
- \( l \) is called the left hand side, and \( r \) the right hand side
- The rewrite rule is called linear if \( \text{Var}(r) \subseteq \text{Var}(l) \)

Positions where symbols or variables occur in \( t \) are denoted by sequence of indices. Such positions are defined as follows (Nymeyer et al., 1997): \( \text{Pos}(a(t_1, \ldots, t_n)) = \{ \epsilon, 1. \text{Pos}(t_1), \ldots, \text{Pos}(t_n) \} \). The subterm of \( t \) at position \( p \) is denoted by \( t_p \). \( \text{Var}(t) \) denotes the set of terms parsing, matching and their respective rewriting. Such forms constitute the basis for the proposed generic normalization strategy and enable the following:

1. Decomposition of the terms parsing, matching and terms rewriting into respective positional functions that perform positional parsing, positional matching and positional rewriting. Such positional functions can be applied in any order, provided that a positional matching criterion is satisfied.
2. Composition of the results of these positional functions provided that composite matching criteria are satisfied.
3. Application of the terms parsing and its respective matching and rewriting in any order, provided that the needed synchronization and dependency relationships are maintained.

**Definition 1. Terms: A Functional Form.**

A functional form of a given term \( t \) is defined as the function:

\[ T(t) = \text{TC}(T_1(T_1.1(\ldots), T_1.2(\ldots), \ldots, T_n(\ldots)(\ldots))(t)), \text{ where:} \]

\[ \epsilon, \ldots, n \text{ corresponds to the set of } \text{Pos}(t) \text{ ordered according to their precedence, reflecting the arity and the depth of the most general form of the term } t. \]

- \( T_1, \ldots, T_n \) are functions, defined to extract the subterm that corresponds to the most general form of the term at positions \( c, \ldots, n \) respectively. Hence, each function \( T_i \) is defined as a mapping: \( T_i(t) \rightarrow t_p \) if \( j = 0 \) and \( T_1 \rightarrow t | (T_1.1, \ldots, T_2) \), if \( i \geq 1 \).

- \( T(t) \) is then defined as a composition of the functions \( T^c(t), T_1(t), \ldots, T_{n-1}(t), T_n(t) \) that can be applied, during term parsing and rewriting, according to any order.

**Definition 2. Rewriting Rules: A Functional Form**

Let \( r_j \) denotes a rewrite rule \( l \rightarrow r \), a functional form of the left hand side and the right hand side of \( r_j \) are defined as:

\[ \text{LR}(r_j) = \text{LR}_c \text{LR}_1(\text{LR}_1.1(\ldots)\ldots)(r_j) \text{ and } \text{RR}(r_j) = \text{RR}_c(\text{RR}_1(\text{RR}_1.1(\ldots)\ldots)r_j) \text{ respectively. Where: each } \text{LR}(r_j) \text{ and each } \text{RR}(r_j) \text{ are defined as the following mappings:} \]

\[ \text{LR}(r_j) \rightarrow \text{LR}(r_j)_j \text{ if } |j| = 0 \text{ and } \text{LR}(r_j) \rightarrow \text{LR}(r_j)_j \text{ (LR}(r_j)_j, \ldots, \text{LR}(r_j)_j, \text{if } j \geq 1. \]

\[ \text{RR}(r_j) \rightarrow \text{RR}(r_j)_j \text{ if } |j| = 0 \text{ and } \text{RR}(r_j) \rightarrow \text{RR}(r_j)_j \text{ (RR}(r_j)_j, \ldots, \text{RR}(r_j)_j, \text{if } j \geq 1. \]

Such mapping decompose the left hand side and the right hand side of \( r_j \) into their constituent subterms. Thus, given TRS rules, their respective functional forms reflect their ranking (as well as their depths) and enable their matching to be decomposed in any order.

**Definition 3. Positional Matching Criteria: A Functional Form**

For a given rule \( r_j \) and a given term \( t \), a positional matching criterion \( \text{MCi}(t, r_j) \) at an arbitrary term rewriting position \( i \) is defined as:

\[ \text{MCi}(t, r_j) = \text{LR}_i(r_j)_j^c \times \text{Ti}(t)_j^c \rightarrow (\text{MPI}, \text{PRWi}), \text{ where:} \]

- \( \text{MPI} \) is a positional matching predicate = \{0,1\}
- \( \text{PRWi} = (\text{rswi}\times \text{Ti}(t)_j^c) \) or ( \( \text{rswi}\times \text{LRi}(r_j)_j^c \) ) or Null) is the positional rewriting defined as the needed rewriting sequence \( \text{rswi} \) and / or the needed substitution in order to have \( \text{Ti}(t)_j^c = \text{LRi}(r_j)_j^c \) and hence
Definition 4. Term Matching Criteria: A Functional Form

For a given rule \( r_j \) and a given term \( t \), the term matching criteria is defined as terms in the following functional form.

\[
\text{MC}(t, r_j) : \text{MC}(\text{MC}_1(\ldots), \text{MC}_2(\ldots), \ldots, (\text{MC}_n(\ldots)) (t, r_j) = \prod_{i=1}^{n} \text{MPI}(t, r_j) \rightarrow \prod_{i=1}^{n} \text{MPI}(t, r_j) \times rws(t, r_j) \rightarrow \text{T}(t, r_j) \rightarrow \text{T}(t, r_j),
\]

where \( rws(t, r_j) \) represents the sequence of positional matching criteria of the term \( t \) and the rule \( r_j \) into a sequence of positional matching criteria. Such a sequence can be activated (during the term rewriting) in any order. However, the matching predicate \( \prod_{i=1}^{n} \text{MPI}(t, r_j) \) serves as indicator for a successful matching between the term and the left side of the rewriting rule \( r_j \). Thus, \( \text{MC}(t, r_j) \) represents the sequence of the rewriting rules to be applied on the considered term, if \( \prod_{i=1}^{n} \text{MPI}(t, r_j) = 1 \).

Definition 5. Positional Matching: A Functional Form

A rewriting step rewrites a given term \( t \) at position \( i \). Its respective functional form is defined as: \( \text{MT}(t, r_j) : (\text{MC}(t, r_j) \rightarrow (\text{MPI}(t, r_j), rsw_i(t, r_j))) \rightarrow (\text{MPI}, \text{Ti}(t, r_j), rsw_i(t, r_j)), \) if \( \text{MPI}(t, r_j) = 1 \). Otherwise, \( \text{MT}(t, r_j) \) returns \( \text{Ti}(t, r_j) \) is defined as a function that calls the matching criteria \( \text{MC}(t, r_j) \) at position \( i \). It rewrites the term with the needed rewrite sequence \( rsw_i \), if \( \text{MPI}(t, r_j) = 1 \). Otherwise, a failure is indicated by \( \text{MT}(t, r_j) \rightarrow (0, \text{Ti}(t), \text{Null}) \).

Definition 6. Term Matching: A Functional Form

For a given term \( t \), its respective matching is defined by functional form:

\[
\text{MT}(t, r_j) : \text{MT} \in (\text{MT}_1(\ldots), \text{MT}_2(\ldots), \ldots, \text{MT}_n(\ldots)) \rightarrow ((\text{MPI}(t, r_j), (\text{MC}(t, r_j), \text{T}(t, r_j), \text{rsw}(t, r_j)))) \rightarrow \text{MT}(t, r_j),
\]

where \( \text{MT}(t, r_j) \) is defined as a composition of the functions \( \text{MT}_1(\ldots), \text{MT}_2(\ldots) \) that are applied at the positions \( \varepsilon \), \( 1, \ldots, n \) to rewrite the term with the needed rewrite sequences at these positions, in order to obtain the term rewritten according to \( r_j \). Thus, \( \text{MT}(t, r_j) \rightarrow ((\text{MT}(t, r_j) = 1), (\text{T}(t, r_j) = \text{T} \in (\text{T}_1(\ldots), \text{T}_2(\ldots), \ldots, \text{T}_n(\ldots))) \rightarrow \text{LR}(\text{LR}(\text{LR}(\ldots(\ldots, \text{T}(t, r_j)))) \rightarrow \text{Ti}(t, r_j), \) if \( \text{MT}(t, r_j) = \text{Ti}(t, r_j) \rightarrow (0, \text{Null}) \), otherwise.

Definition 7. Term Rewriting: A Functional Form

Let \( r_j : l \rightarrow r \in \mathbb{R} \) and \( r_i : r \rightarrow s \), otherwise \( r_i : \varepsilon \), the term rewriting is defined as one of the following functional forms:

\[
\text{TR}(t, r_j) : (\text{MT}(t, r_j) \rightarrow (T(t, r_j) \rightarrow (T(t, r_j)), \text{TR}(T(t, r_j)), r_j)) \rightarrow (\text{MT}(t, r_j) \rightarrow (0, T(t, r_j)), \text{Null}),
\]

where: \( \text{TR}(T(t, r_j), r_j) \) is a function that rewrites the term \( T(t, r_j) \) as produced by the function \( \text{MT}(t, r_j) \), and having the form \( T \in (T_1(\ldots), T_2(\ldots), \ldots, (T(t, r_j) = \text{LR}(\text{LR}(\text{LR}(\ldots(\ldots(\text{T}(t, r_j)))) \rightarrow \text{Ti}(t, r_j)) \rightarrow (0, \text{Null}) \), otherwise.

Example 1: Let \( t = a \cdot (b + (c, d)) \), \( \Sigma = \{a, b, c, d\} \), \( \Sigma_\varepsilon = \{+\} \) and \( \Sigma_\text{arity} = \{\} \).
\[ R = \{ r_1: a \rightarrow r, r_2: b \rightarrow r, r_3: d \rightarrow a \} \]

We illustrate the above definitions as follows:

- The functional form of the term \( t \) is \( T(t) = Tc(T1, T2) \) (T2.1, T2.2), where: \( Tc(t) = +T(T1, T2), T1(t) = a, T2(t) = b + c \), \( T2.1(t) = b \) and \( T2.2(t) = c \).

- The functional form of \( R \) is \( \{ LR(1), RR(1) \}, LR(r1) = +LR1, RR(r2), RR(r3) \), where: \( LR1(r1) = +LR1, LR2(r1), LR1(r1) = a, LR2(r1) = r \) and \( RR(r1) = r \).

- The application of the functional forms as given by definitions 3-7 on the term \( t \) with respect to the rewrite rule \( r1 \) proceeds as follows:

\[ \text{rule } r1 \text{ proceeds as follows:} \]

- The positional matching criteria are instantiated as follows:

\[ \text{(MP1, T1(t, r1), rws1 (t, r1))) = (1, a, r), since MP(r1) = 1} \]

\[ \text{MTc (t, r1) : (MC}1_{1} (t, r1) \rightarrow (\sum MP_{c}, T1(t), rws (t, r1))) = (1, +, e), since MPc (r1) = 1} \]

\[ \text{MR}_{2}(t, r1) : (MC}2_{2}(t, r1) \rightarrow (MP_{2}, T2(t), rws_{2} (t, r1))) = (1, +, b, c), since MP_{2}(t, r1) = 1} \]

- The term matching and the term rewriting are as follows:

\[ \text{TR}(t, r1) : (MT(t, r1) \rightarrow T(t, r1)) \rightarrow T(t, r1) = (1, +a r1, c, r2) \rightarrow T(t, r1) \rightarrow T(t, r1) \rightarrow r \rightarrow r). \]

5. GENERIC NORMALIZING STRATEGY

The functional form \( TR(t, r) \) for a given term \( t \), as given by definition 7, permits terms parsing, matching and rewriting in any order. To produce set of normal forms respective to the term \( t \), we introduce a generic normalization strategy, defined as a function that impose the effect of different reduction strategies on \( TR(t, r) \) in a generic way. We formalize such strategy as follows, while its implementation is given in section 6.

Let \( ((\Sigma, R), V) \) be a TRS, where \( R \) is a set of rewrite rules \( \{ r_1, \ldots, r_n \} \) represented as:

\[ \text{LR(r1) = LR1, LR2(1), \ldots, LRn(1), RR(r1) = RR1, RR2(1), \ldots, RRn(1)} \]

Let \( t \) is a term over TRS represented as \( T(T1(\ldots) \ldots Tn(\ldots)) \).

Let \( R \) be a generic reduction strategy, either innermost or outermost.

Let \( IR = \{ IR_{j} \mid i \in IR(T(t)) \} \) is the set of initial rewriting rules at the root positions of the subterms of the most general forms of \( t \) (such set constitutes a safe approximation that reduces the number of redundant and failing reductions). We define the possible set of normal forms of the term \( T(t) \) with respect to the set \( IR \) as the set \( N(T(t), IR) \).

- The positional matching criteria are instantiated as follows:

\[ \text{MC(t, r1) : MC1, MC2( t, r1) \rightarrow (T1(t, r1) \times e (t, r1), T2(t, r1) \rightarrow r2(t, r1))} \]

- The positional matching is instantiated as follows:

\[ \text{MTc (t, r1) : (MC}1_{1} (t, r1) \rightarrow (MP_{1}, T1(t), rws1 (t, r1))) = (1, +, e), since MPc (r1) = 1} \]

\[ \text{MT}_{2}(t, r1) : (MC}2_{2}(t, r1) \rightarrow (MP_{2}, T2(t), rws_{2} (t, r1))) = (1, +, b, c), since MP_{2}(t, r1) = 1} \]

- The term matching and the term rewriting are as follows:

\[ \text{MT(t, r1) : MT(c, T1(\ldots), T2(\ldots)) = (1, +a r1, c, r2) \rightarrow T(t, r1) \rightarrow r \rightarrow r).} \]

Lemma 1: Generic Normalizing Function (GNF)

Let \( ((\Sigma, R), V) \) be a TRS, where \( R \) is a set of rewrite rules \( \{ r_1, \ldots, r_n \} \) represented as:

\[ \text{LR(r1) = LR1, LR2(1), \ldots, LRn(1), RR(r1) = RR1, RR2(1), \ldots, RRn(1)} \]

Let \( t \) is a term over TRS represented as \( T(T1(\ldots) \ldots Tn(\ldots)) \).

Let \( R \) be a generic reduction strategy, either innermost or outermost.

Let \( IR = \{ IR_{j} \mid i \in IR(T(t)) \} \) is the set of initial rewriting rules at the root positions of the subterms of the most general forms of \( t \) (such set constitutes a safe approximation that reduces the number of redundant and failing reductions). We define the possible set of normal forms of the term \( T(t) \) with respect to the set \( IR \) as the set \( N(T(t), IR) \).

- The positional matching criteria are instantiated as follows:

\[ \text{MC(t, r1) : MC1, MC2( t, r1) \rightarrow (T1(t, r1) \times e (t, r1), T2(t, r1) \rightarrow r2(t, r1))} \]

- The positional matching is instantiated as follows:

\[ \text{MTc (t, r1) : (MC}1_{1} (t, r1) \rightarrow (MP_{1}, T1(t), rws1 (t, r1))) = (1, +, e), since MPc (r1) = 1} \]

\[ \text{MT}_{2}(t, r1) : (MC}2_{2}(t, r1) \rightarrow (MP_{2}, T2(t), rws_{2} (t, r1))) = (1, +, b, c), since MP_{2}(t, r1) = 1} \]

- The term matching and the term rewriting are as follows:

\[ \text{MT(t, r1) : MT(c, T1(\ldots), T2(\ldots)) = (1, +a r1, c, r2) \rightarrow T(t, r1) \rightarrow r \rightarrow r).} \]
- RS is a generic reduction strategy, instantiated by different specific ones.

- GNF(t, r, RS) applies the order induced by RS on the decomposition sequence of functions TR(t, r) and MT(t, r) respectively, with an objective to complete matching of the left side of the rule r1 (LR(r1)) and rewriting the matched subterm by the right side (RR(r1)), either as a final goal or as a subsequent goal for further expansion or reduction. Proof (by construction):

  GNF (t, r, RS) applies the order induced by the reduction strategy RS on the decomposition sequence produced by TR (t, r), and MT (t, r), respectively, as follows:

  \[
  \text{RS (TR(t, r))} \rightarrow \text{RS (MT(t, r))} \rightarrow \text{RS (MT (T_1 (\ldots), MT_{\ldots}, MT_{\ldots}))} \rightarrow \\
  \text{RS (MT_{n1} (\ldots), MT_{n2} (\ldots), MT_{n3} (\ldots), \ldots, MT_{nk} (\ldots))} \rightarrow \\
  \text{((MC_{n1} (\ldots), MC_{n2} (\ldots), MC_{n3} (\ldots), \ldots, MC_{nk} (\ldots), MC_{nk+1} (\ldots))} \rightarrow \\
  \text{((MP_{n1} = 1, T_{n1} (t) \times rws_{n1} (t, r_1)) \rightarrow (T_{n1} (\ldots), T_{n2} (\ldots), T_{n3} (\ldots), \ldots, T_{nk} (\ldots))(t, r_1))} \\
  \]

  Where \( n1, n2, n3, \ldots, nk \in \mathbb{N} \) and represent the order induced by the reduction strategy.

  The definition of GNS shows the following: Firstly, the function MT is applied at the initial position n1 induced by RS, and if MP_{n1} = 1, the term(T_{n1} (\ldots)(t, r_1)) is rewritten by rws_{n1}. Secondly, MT_{n1} is applied at the subsequent position n2 defined by RS, and if MP_{n1} = 1 and MP_{n2} = 1 then the term(T_{n1} (\ldots), T_{n2} (\ldots))(t, r_1) is rewritten by rws_{n2}. This process is continued, until the whole term is rewritten, provided that the dependency relationships between the reduced subterms are maintained. This is insured by the required condition that a subterm at position n is reduced if its matching predicate and the matching predicates of its predecessors are true (\( \prod_{\varepsilon} MP_{n-1} = 1 \) and MP_{n} = 1). Thus, the order according to which the term t is rewritten is expressed in a generic way. The specific order is obtained by the instantiation of GNS (t, r, RS) by different specific normalization strategies. Such instantiation is reduced to the instantiation of RS by a specific reduction strategy. This is achieved as follows.

- Innermost (bottom-up) Normalizing Strategy (BNS)

  According to BNS, the order of computation of TR (t, r) and respectively MT (t, r) is as follows: The function MT_{n1}(t, r) is applied first, followed by the function MT_{n2}(t, r) provided that MPn(t, r) is true. Thus, the innermost subterms are reduced first and subsequently the terms at lower positions. Hence, the instantiation of the generic normalization strategy by BNF is reduced to define RS (TR(t, r) = BNS (TR(t, r)), where:

  \[
  \text{BNS (TR(t, r))} \rightarrow \text{BNS (MT(t, r))} \rightarrow \text{BNS (MT (\ldots), MT_{\ldots}, MT_{\ldots}))} \rightarrow \\
  \text{((MC_{n1} (MC_{n1} (\ldots), \ldots, MC_{n2} (\ldots), \ldots, MC_{nk} (\ldots), MC_{nk+1} (\ldots))} \rightarrow \\
  \text{((MP_{n} = 1, T_{n} (t) \times rws_{n} (t, r)) \rightarrow (T_{n} (\ldots), (t, r), ((MP_{n} = 1 \text{ and MP}_{n+1} = 1)) \rightarrow (T_{n+1} (\ldots), T_{n} (\ldots)(t, r_1))} \rightarrow \\
  \text{((MP_{n} = 1, T_{n} (t) \times rws_{n} (t, r_1)) \rightarrow T \varepsilon (T_{1} (\ldots), T_{2} (\ldots), \ldots, T_{nk} (\ldots))} \rightarrow \\
  \]

  Example 2 Let a TRS be given as in Example1, we illustrate the instantiation of the generic normalization strategy (GNF) by BNS and TNS as follows:

  \[
  \text{Example 2} \rightarrow \text{Example1} \\
  \text{The instantiation of GNF by BNS proceeds as follows:} \\
  \text{- The instantiation of GNF by BNS proceeds as follows:} \\
  \text{- Outermost (top down) Normalizing Strategy (TNS)} \\
  \text{According to TNS, the order of computation of TR (t, r) and respectively MT (t, r) as follows:} \\
  \text{MTn (t, r) then MTn-1(t, r), provided that MPn+1 (t, r) is true. Thus, the outermost subterms are expanded first and subsequently the terms at upper positions. Hence, the instantiation of the generic normalization strategy by TNS proceeds as follows:} \\
  \text{- The instantiation of GNF by TNS proceeds as follows:} \\
  \text{- The instantiation of GNF by BNS proceeds as follows:} \\
  \text{- Outermost (top down) Normalizing Strategy (TNS)
MC \in (MC_1, MC_2) \rightarrow (MPc = 1, T_c (t)) \times \text{rws}_c (t, r_1)) \rightarrow \text{r}_1\rightarrow (T_1) (t, r_1)) (\text{mp}_1 = 1 \text{ and } \text{mp}_c = 1 = 1), T_2 (t) \times \text{rws}_2 (t, r_2)) \rightarrow T \in (T_1, T_2)(t, r_1). \text{ Hence, such instantiation produces the following results:}

((1, +, \text{e} \rightarrow +), (1, a, \text{e} \rightarrow +a), ((1, + (bc) \times r_2) \rightarrow +a r)) \rightarrow r

6. GENERIC NORMALIZING FRAMEWORKS

To implement the generic normalizing strategy we define two generic normalizing frameworks. (Each one can be instantiated as bottom-up or as top- down): The first framework (GNFR1) is described in section 6.1. It is based on considering the initial rules one by one, according to a predefined order, and on normalizing the term with respect to each one of the considered rules. The second framework (GNF2) is described in section 6.2. The second framework considers the subsequent parsing positions and performs their respective matching to the left sides of all initial rules. Thus, the given term is scanned once for all rules. A distinguished feature of both frameworks is their incremental approach. Further more, the cardinalities of the generated rewriting sequences (rws) is optimized. This due to two factors: the use of initial rules which restricts the number of rules to be applied and defining the needed rws based on established goals.

6.1. Generic Normalizing Framework (GNFR1)

Let \((\Sigma, R, V)\) is a TRS, where \(R\) is a set of rules \(\{r_1, \ldots, r_n\}\) represented as: \(LR(r_i) = LRC(r_i) \rightarrow RR(r_i) = \text{rr}(r_i)\). The computation of \(MT(t, r_i)\) is performed based on applying its constituent positional matching functions (\(MTp(t, r_i)\)). Each \(\text{MTp}(t, r_i)\) computes its respective \(\text{MCp}(t, r_i), \text{Tp}(t, r_i)\) and \(\text{rws}_p(t, r_i)\), as shown in figure 3. The proposed GNF1 permits adoption of any reduction strategy RS and respectively the order according to which the computation of \(MT\) is performed. Hence, GNFR can be instantiated by different specific normalization strategies. We have distinguished two specific ones: bottom-up (BNS) and top-down (TNS), where MT can be instantiated to select next \(LRp(r_i)\) according to BNS or TNS respectively.

GNF1\(i\) \(r_i, T(t), RS, MP(t, r_i), \text{rws})\)
\{ \(T(t, r_i) = T(t), MP(t, r_i) = MP(t, r_i)\), \(\text{rws}(t, r_i) = \text{rws}\) \}
\{MP(t, r_i) = 1 \text{ if } \text{RR}(t, r_i) = \text{IRR} \{ \text{return} (MP(t, r_i), T(t, r_i) = \text{RR}(t, r_i) \text{ rws}(t, r_i)) \}
\}\}
\{else if GNF1 \(r_i = 1, \text{RR}(t, r_i) \text{ and } \text{rws}(t, r_i)\) \}
\{else if return (MP(t, r_i) = 0, T(t, r_i) = T(t), \text{rws}(LRi) = \text{Null})\}

Figure 1. The Generic Normalizing Function (GNF1\(i\))
A Generalized Normalizing …

Riad S. Jabri

While ( ∃ p ∈ POS (LR(ri)) such that LRp (ri) is un processed)
Select next LRp (ri) according to RS
(MPp (t, ri), rws_p (t, ri), Tp (t, ri)) = MTP (t, ri)
If MPp (t, ri) = 1 { rws (t, ri)= rsw (t, ri) ∪ rsw_p (t, ri), MP(t, ri) = MP(t, ri) ∩ MPp(t, ri) }
Elseif return (MP(t, ri)=0, T(t, ri)=, rws(LRi)= Null)

Figure 2. The Term Matching Function (MTp (t, ri))

Theorem 1
Let ((Σ, R), V) be is constructor-based TRS. Let t is a
term over TRS, represented as T (t). Let R is a set of
rules{ r 1,…,r n}, represented as {LR(r i) → RR(ri),…,
LR(rn) → RR(rn)}.
Applying GNF1ri (ri, T(t), RS, MP(t, ri), rws) on T(t)
according to the reduction strategy RS, defined as BNS
(TNS), produces the set of the t descendants R*(t) as
intermediate results and its normal form as end results.

Proof (by construction)
Applying GNF1ri (ri, T(t), BNS(TNS), MP(t, ri), rws)
on T(t) proceeds as follows : TR(t, ri) →…. MT n(t, ri)
MTn-1( t, ri) .If RS is BNF (TNF), the subterms at a
higher(lower) position are matched and then the subterms
at a lower position (higher positions), provided that if any
of the subterms at higher positions (lower position) are
reducible, they will be normalized. This is insured by the
condition MPn(n-1) to be true in order to match the
subterm at position n-1(n) and recursively calling the
function GNF1 upon detection of reducible terms at
higher positions (lower position), as well as if the
instances of the right sides that are reducible. This insures
recognizing these terms, their respective right sides and
any subterms derived from them.

If (LRp(ri) = Tp(t, ri)) {return (MPp (t, ri) =1, Tp(t, ri)= Tp(t, ri), rws_p (t, ri)= ε )}
Elseif If (σ (LRp(ri)) =Tp(t, ri)) { If ∃ a rule r_k | first(LRk (rk))= Tp(t, ri) |n, n (∈ …,[p])
{ ( Tm(t, ri)= | ε | Tp(t, ri) | ε | ∪ … ∪ Tp(t, ri) |ε),
(MPn(t, ri), Tn(t, ri), rws_n(t, ri)) = GNF1 rk (rk, Tn(t), RS, MP(t, R), rws),
return (MCp(t, ri)= MPn(t, ri), Tp(t, ri)=Tm(t) ∪ Tn(t, ri),
rws_p(t, ri)=rws_n(t, ri) )
Elseif ( return (MCp(t, ri)=1, Tp(t, ri)=Tm(t), rws_p (t, ri)= ( ε , σ )},
Elseif If ( LRp(ri) = σTp(t)) { If ∃ a rule r_k | first(LRk(ri))= LR(rk)
{return (MPp (t, ri), Tp (t, ri), rws_p (t, ri)) =
GNF1 rk (rk, LR (ri), RS, MP (t, R), rws)}
Elseif return( MPp(t, ri)=1, rws_p (t, ri)= ( ε , σ ), Tp(t, ri)=LRp(ri))
Elseif If (LRp (ri) ≠ Tp (t)) If Tp → , LRp
{ return(MPP(t, ri)=1, rws_p (t, ri)= r Tp(t, ri)=LRp(ri))
Elseif If LRp → , Tp
{ return(MPP(t, ri)=1, Tp(t, ri)= r Tp(t, ri)=LRp(ri))
Elseif If LRp → , , Tp
{ return(MPP(t, ri)=1, rws_p (t, ri)= r Tp(t, ri)=LRp(ri))
Elseif If (Tp → r1 , c and LRp → r2 , c)
{ return (MPp(t, ri)=1, rws_p (t, ri)= (r1 , rws(LRp(ri)) = r2)
Elseif If ∃ a rule r_k | first (LR (rk)) = Tp(t) and RR(rk) = LRp(ri), σ)
{ return (MPp (t, ri), Tp (t, ri), rws_p (t, ri)) =
GNF1 rk ((rk, σ), Tp (t), RS, MP (t, R), rws)}
Elseif return (0, Null)

Figure 3. The Positional Matching Function (MTp (t, ri))

Example 3: This example is borrowed from (Rety,
2005), where it has been used to demonstrate a proposed
algorithm that generates the set of the set of descendants
R (E).
Let E = g (f(s(a))) and R= {g(x) → h(x), h(p(x) →
g(x), f(s(a)) → p(f(x))). Considering E=t and applying
Definition 8: Most General Form of Terms

Given a term \( t \), its most general form is defined as \( \text{MGT}(t) = T^c(\ldots)T_i(\ldots)\ldots T_n(\ldots) \), where \( i \in \text{POS}(t) \) and \( Ti \) plays the role of variables as well as extracting functions, as in definition 1, Thus \( Ti \) is also represented as \( \text{MGT}(Ti) = Ti(T_1, T_2, \ldots, T_n) \). The purpose of this definition is to represent the term in incremental and ranked way as it is being parsed, for examples:

- Let \( t = +cc \), rank \((+)=2 \) and rank\((c)=0 \). The initial representation of the term \( t \) is \( \text{MGT}(t) = +(T_1, T_2) \) and the subsequent representations, during parsing, are: \(+ (c, T_2) \) and \(+ (c, c) \).

- Let \( t = c + c \). \( \text{MGT}(t) \) will be \( c + (T_1, T_2), c+(c, T_2), c+(c, c) \).

Definition 9: Positional Rewriting Sequence

Given a term \( t \) represented as \( \text{MGT}(t) = T(T_1, T_2, \ldots, T_i, \ldots, T_n) \), \( T \), we define a respective positional rewriting sequence as \( \text{PR}(t) = R \in \{ R_1(R_1.1(\ldots)) \ldots R_2(\ldots), R_i \} \), where \( R \) = rewriting rule or rws to be applied at position \( i \). The purpose of this definition is to represent the rewriting sequences \( R(R_1(R_1.1(\ldots)\ldots),R_2,\ldots,R_i) \) in incremental and ranked way as they are defined, for example: \( r_1 \in \{ R_2 \ldots r_1 \in \in \{ r_4 \} \) (R_2 R_1 R_2).

6.2.1. Generic Normalizing Framework (GNFR2) Pass 1

Pass 1 has as parameters a term \( t \) and a set of rewriting rules \( R \). It proceeds according to the following steps.

1 – Initialization

1.1 Determine the most general form of the term \( \text{MGT}(t) \), the set of initial rules \( \text{IR} = \{ r_1, r_2, \ldots, r_m \} \) to be applied at the root position of \( t \) and represent the respective set of rules \( R \) in ranked form. Thus, obtaining:

\[ \text{MGT}(t) = T \in \text{E} = (T_1, T_2, \ldots, T_i, \ldots, T_n) \]

\[ \text{PR}(t) = \{ \text{PR}_1 = r_1((R_1(R_1.1(\ldots)\ldots), R_2, \ldots, R_i))(t, r_1), \text{PR}_2 = r_2((R_1(R_1.1(\ldots)\ldots), R_2, \ldots, R_i)(t, r_2), \ldots, \text{PR}_n = r_n((R_1(R_1.1(\ldots)\ldots), R_2, \ldots, R_i)(t, r_n)) \}

\]

1.2 Set the initial parsing position \( p = e \).

1.3 While \( (\exists \ p \in \text{POS} (T(t)) \) such that \( Tp(t) \) is unprocessed) \{1.3.1 Select the next subterm \( T(T_i) \) at position defined by \( p = \) next \( p \) \}

(For bottom up: the function next \( (p) \) is defined to return the positions in descending order)

1.2. T(t) = T \in \text{E} = (T_1, T_2, \ldots, T_i, \ldots, T_n) \)

1.3. For each \( R_i \in \text{PR} \{ \}

If (the respective rule \( R_p \in \text{PR}_i \) has been defined) then do nothing, else select the rule \( r_j \) respective to the root of \( Tp(t) \). If the selected \( r_j \) has not been defined, then define \( r_j \) as follows:

Let \( k \) be the position of the root, \( r_j \) is any rule such that \( T(t)_k = \text{First}(r_j) \) and \( RR(t)(r_j) = LR_k(r_j, \sigma) \) ;

\[ \text{MP}(t, r_j), \text{rws}_p(t, r_j, \sigma) = \text{MTP}(T(t), r_j) ; \text{R}p(\text{PR}_i) = \]
rws_p (t, r_j); MP_p (PR_i) = MP_p (t, r_j);  
(MP_r (PR_i) = MP_r (PR_i) and MP_p (PR_i)); pass1-position (PR_i) = pass1-position (PR_i) \cup \mathcal{P} 
}
1.4 End pass1.

The matching function MP used by Pass1 is the same as the function used by GNFR1, but with a modification that reflects the need to return the matching predicate (MP_p (t, r_j) and the rewriting sequence rws_p (t, r_j), without rewriting the term. An extract from the function MP_p (t, r_j) is given in figure 4 to illustrate such modifications.

![Figure 4. An Extract from a Modified Version of the Function MP_p(t, r_j)](image)

### 6.2.2 Generic Normalizing Framework GNFR2: Pass 2

Pass2 rewrites the term t according to the rewriting sequence PR(t IR) = \{PR_1, PR_2, \ldots, PR_n\}, computed by PR_1 = PR_1(t, IR), PR_2(t, IR), \ldots, PR_n(t, IR)

For each PR_i \in PR \{Pass2-position = Reverse (pass1-position (PR_i)),

For k=1 to \| pass2-position (PR_i) \| {J= Next (pass2-position (PR_i))

If (MP (r_j (PR_i) =1) {T_j (PR_i, t) = TR (T_j (t), r_j),

Append (T (PR_i, t), PR_i (t, PR_i)); Elseif Failure ;} 

If T (PR_i, t) is a redex \{Pass1 (T (PR_i, t))\} Elseif return (T (PR_i, t) \in PR_i (t, PR_i)) \}

### Example 4: Applying GNFR2 on t= g(f(s(a)))

The goal has been solved using semantic unification based on narrowing strategy, called RR\^1. Also, it has been solved by other strategies. Table (1) is given in (Butow et al, 1999) to solve the goal g(f(s(a)))=? c(a), using the following TRS: R1: f(a) \rightarrow a, R2 : f(b(X) \rightarrow b(f(X)), R3: f(c(X) \rightarrow a, R4: g(a, X) \rightarrow b(a), R5: g(b(X), a) \rightarrow a, R6: g(b(x),b(Y)) \rightarrow b(a), R7: g(b(x),c(Y)) \rightarrow b(a), R8: g(c(x), Y) \rightarrow b(a));

The goal has been solved by GNFR1 and GNFR2 to solve the goal. Such goal has been solved by GNFR1 and GNFR2 as follows.

**A) GNFR1 solution**

- The term t= g(X, f(X)) and the rewriting rules {r_1, \ldots, r_6} have been represented as T(t)= T (T \in \mathcal{T}_E (T_1, T_2(T_1)), and as (LR(r_1)=\ldots, RR(r_1)=\ldots, LR(r_6)=\ldots, RR(r_6)=\ldots) accordingly.

- The first two steps have produced : (rws(t,r_6)= rsw(t,r_6)) \cup \mathcal{R}_w (t, r_6) = c, \sigma (X \rightarrow b(x)), (MP(t,r_6) = MP (t, r_6)) and MP_1 (t, r_6) =1 and ( T (t, r_6) = T (t, r_6) \cup T \rightarrow g(b(x)))

- At the third step has produced: MP(t, r_6) =1, T (t, r_6) = g(b(x), b(Y)) and (rws(t,r_6)= c, \sigma (X \rightarrow b(x)), (c, \sigma (X \rightarrow f(X2))) r_6)

- The last step has produced T (t, r_6) =RR(R_6)

**B) GNFR2 solution**
- The initialization step has produced : $\text{MT}(t)= T \varepsilon (T_1, T_2)$, (PR1= R6 (R1,R2), and (T(t)=g(T1, T2))

- Pass1 has produced : (T(t)=g( X f(X)), and PR1= R6 ((c, σ (X→ b(X1)), (R2, (e, σ (Y→ f(X2))) ((c, σ (X→ b(X2)

- Pass2 has produced the following:

  $T \ 2.1 \ (PR1,t)= TR (T2.1(t),((c, σ (X→ b(X2)= f(b(X2)) and T (PR1,t)= f(b(X2))

  $T \ 2 \ (PR1,t)= TR (T2 (t) (R2, (e, σ (Y→ f(X2))= (b(Y)) and T (PR1,t)= (b(Y))

  $T \ 1 \ (PR1,t)= TR (T1 (t) ((c, σ (X→ b(X1)= (b(X1)

and $T (PR1,t)= (b(X1),(b(Y))$

- The term $t= \varepsilon$ has been called to normalize $t$ with respect to r1, r2 and r3 respectively. As an example, the normalization according to r1 proceeded as follows:

$$
\text{GNF1}_{r1} \ r1, \ T(t), \ RS, \ MP(t, \ r1), \ rws
$$

$$
\text{step1&2} \ MP \ (t, r1) = MP \ (t, r1) \ (MP \ (r1) = 1), \ T (t,r1) = \varepsilon, \ rws \ (t,r1) = E \ r6 \ rws \ (r1) = E \ r5 \ r4, \ rws \ (r1) = E \ r5 \ r4.
$$

The bottom-up approach proposed in (Durand and Senizergues, 2007) is based on the so called marked derivations, which have been used to prove that the bottom - rewriting is inverse recognizability preserving. The derivation sequence generated by applying GNFR2 on the term f(h(h(a))) is exactly the marked derivation given in (Durand and Senizergues, 2007) for the same term. Such results demonstrate that our approach preserves recognizability and generates the set of descendants of a recognizable term.

**Example 7:** This example was given in (Katoen and Nymeyer, 2000) to illustrate a proposed approach for a code generator based on terms rewriting. A weighted TRS was given as $(\Sigma, V, R, C)$, where: $\Sigma = (S, r), V = \{ x, y \}, S = \{ +, i, c, d, r \}, r^+(i) = 2, r(i) = 1$ and r(c) = r(d) = r(r) = 0. The set rewrite rules of was defined as $R = \{ r1 : +(d, c) → i(d), r2 : +(d, r) → r, r3 : +(x, y) → +(y, x), r4 : i(r) → r5 : d → r6 : c → d, r7 : r → d \}$. The cost function C was defined as $C(r1)=4, C(r2)=5, C(r3)=0, C(r4)=2$.

The application of GNFR1 on $t= +(c, i(d))$ has produced r as follows:

- The term $t= +(c, i(d))$ and the rewriting rules $\{r1, \ldots, r7\}$ have been represented as $T(t)= T E( T_1, T_2(T_2,i))$ and as $(LR(r1)=\ldots, RR(r1)=\ldots), \ldots, (LR(r7)=\ldots, RR(r7)=\ldots)$ accordingly. The set of initial rules IR= $\{r1, r2, r3\}$, so $\text{GNF1}_{r3}$ has been called to normalize $t$ with respect to r1, r2 and r3 respectively. As an example, the normalization according to r1 proceeded as follows:

$$
\text{GNF1}_{r1} \ r1, \ T(t), \ RS, \ MP(t, \ r1), \ rws
$$

$$
\text{step1&2} \ MP \ (t, r1) = MP \ (t, r1) \ (MP \ (r1) = 1), \ T (t,r1) = \varepsilon, \ rws \ (t,r1) = E \ r6 \ rws \ (r1) = E \ r5 \ r4, \ rws \ (r1) = E \ r5 \ r4.
$$

**Table 1. A comparison between GNFR1, GNFR2, R^\text{R^R} and other strategies**

<table>
<thead>
<tr>
<th>Method</th>
<th>#solutions found</th>
<th># Branches unfinished</th>
<th># Dead ends</th>
</tr>
</thead>
<tbody>
<tr>
<td>R^R</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Forward decomposition</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Innermost narrowing</td>
<td>16</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Left-right basic narr.</td>
<td>25</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Needed narrowing</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>GNFR1 &amp; GNFR2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table (2) shows the comparisons results between GNFR1, GNFR2 and the Pattern Matching Algorithm (PMA) suggested in (Katoen and Nymeyer, 2000), as well as the results of a naive algorithm (NA) given in (Katoen and Nymeyer, 2000). Such a comparison is given in terms of the cardinality of the rws at each position.
A Generalized Normalizing …

Table 2. Cardinality of rewriting sequences

<table>
<thead>
<tr>
<th>Method</th>
<th>Cardinality of rws per position</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(c)</td>
</tr>
<tr>
<td>NA</td>
<td>3</td>
</tr>
<tr>
<td>PMA</td>
<td>3</td>
</tr>
<tr>
<td>GNFR1 &amp; GNFR2</td>
<td>1-2</td>
</tr>
</tbody>
</table>

8. CONCLUSION

In this paper, we have proposed a generic and formal normalization approach and two alternative frameworks for its implementation. Both frameworks are based on determining a set of initial rules to be applied at the root positions of the subterms of the most general forms of a given term t. Such set constitutes a safe approximation that reduces the number of redundant and failing reductions. In addition, both frameworks normalize the term t according to an order induced by a reducing strategy, given as a parameter. However, the first framework, called GNFR1, considers the initial rules one by one and for each rule it parses the input term, at the positions that are defined by the reducing strategy, with an objective to complete the matching process of the left side of the considered rule and obtaining its right side respectively either as a final goal or as a subsequent goal for further expansion or reduction. The second framework, called GNFR2 considers the initial rules and at each parsing position completes the respective matching for all left sides of these rules. Thus, the term t is scanned once for all rules. The main contributions of our work are as follows:

- We have provided the theory on which the generic normalizing strategy is based and proposed generic frameworks for its implementation.
- We have introduced functional forms for the terms, rewriting rules, terms matching and terms rewriting. Such forms can be used to decompose, aggregate and synchronize the term parsing, matching and rewriting. Thus, permitting a generic reduction strategy to be applied on other than the constructor–based TRS.
- We have introduced the concept of initial rules in order to minimize the cardinality of the rewriting sequences.

We have implemented GNFR1 and GNFR2 using innermost and outermost reduction strategies and we have applied them using different examples, representing different approaches to demonstrate their genercity, correctness and performance. They have shown a superiority in terms of minimizing the number of rewriting solutions and the respective number of rewriting sequences. Also they have opened directions for future work such as their application in real code generators, program correctness and encryption.

REFERENCES


