A SHORT PROOF OF THE DECIDABILITY OF NORMALIZATION IN RECURSIVE PROGRAM SCHEMES

Zurab Khasidashvili

Intel Corporation, Haifa, Israel
zurab.khasidashvili@intel.com

Dedicated to work of Shalva Pkhakadze on the centenary of his birth

Abstract

We give a short and simple proof of the decidability of normalization in Recursive Program Schemes (RPSs). As a side result, we obtain an algorithm that effectively transforms any RPS into an irreducible one, in which shortest normalizing reductions are easy to construct.

1 Introduction

It is shown in [6] that normalization is decidable in *Recursive Program Schemes (RPSs).* The proof there is quite complex as it employs the concept of *essential chains* of rules: An essential chain of rules is a sequence of rules $r_1, r_2, \ldots$ such that an r_{i+1}-redex has an *essential occurrence* in the right-hand side of r_i, for all $i = 1, 2, \ldots$. Here a subterm (in particular, a redex) is called *essential* if it has a descendant under any reduction of the term (where the concept of *descendant* is a refinement of that of residual; it allows to trace subterms along reductions).[1] It is shown in [6] that a term t in an RPS R is normalizable iff all essential chains of the rules corresponding to essential redexes in t are finite. Showing the decidability of normalization thus required showing the decidability of essentiality in RPSs.

In this paper we give a much shorter and simpler (thus less informative) proof. We assume that the reader is familiar with the basic concepts of Term Rewriting: All needed information can be found in [9, 4, 13]. We use t and s to denote terms, u to denote redexes, and r to denote rules.

[1]For the reader familiar with the concept of *neededness of redexes* [4], we remark that essentiality is a refinement of neededness in that it makes sense for all subterms, and not redexes only.

We write $t \to s$ or $t \xrightarrow{u} s$ if s is obtained from t by reducing a redex u, and $\twoheadrightarrow$ denotes the transitive reflexive closure of $\to$; we write $\overset{\emptyset}{\twoheadrightarrow}$ when the number of steps is 0.

2 The proof

We start by introducing RPSs [2, 9].[2] RPSs have been studied in [11, 12] under the name of *contracting symbols of type I.*

Definition 1. *An RPS R is a Term Rewriting System (TRS) [9, 13] whose alphabet consists of a finite set $\mathcal{F}$ of* unknown *function symbols, a finite set $\mathcal{G}$ of* basic *function symbols, and variables. The rules of R have the form*

$$r : f(x_1, \dots, x_n) \to s,$$

where f is an unknown function symbol in $\mathcal{F}$, x_i are pairwise distinct variables, and s is an arbitrary term built from function symbols (basic or unknown) and variables. There is exactly one rule in R for every unknown function symbol in $\mathcal{F}$.

A rule r as above is called *irreducible* if s is an R-normal form, and is called *reducible* otherwise. We call an RPS *irreducible* if every rule in it with normalizable right-hand side is irreducible. That is, an irreducible RPS may contain rules whose right-hand sides are not in normal form, but these right-hand sides are *not* normalizable. Clearly, if a rule $r \in R$ is irreducible, for any term t in R, the normal form of t w.r.t. $\{r\}$ can be computed in (at most) as many steps as the number of r-redexes in t. This is why irreducible rules are attractive.

Lemma 1. If all rules of an RPS R are reducible, then no reducible term t in R has a normal form.

Proof. Suppose on the contrary that t has a normalizing reduction $t \twoheadrightarrow t' \xrightarrow{u} t^*$. Then the right-hand side of the rule for the redex u must be a normal-form – a contradiction. □

Lemma 2. Let R be an RPS containing an irreducible rule r. Further, for any term s in R, let s^r denote its $\{r\}$-normal form. And finally, let R^r be the RPS obtained from R by $\{r\}$-normalizing the right-hand sides of rules in R (i.e., by replacing all rules $t_1 \to t_2$ in R with $t_1 \to t_2^r$, respectively), and then by removing r. Then a term t in R is normalizable in R iff t^r is normalizable in R^r.

[2]RPSs are called *Recursive Applicative Program Schemes* in [2].

Proof. ($\Leftarrow$) Any step in R^r can be decomposed into an $R\backslash\{r\}$-step followed by a number of r-steps (contracting all created r-redexes). Hence, t^r is R-normalizable (since it is R^r-normalizable), and thus so is t (since $t \twoheadrightarrow t^r$ in R).

($\Rightarrow$) By induction on the length of a shortest R-normalizing reduction $t \xrightarrow{u} t_1 \to \ldots \to t_n$ starting from t. Using the Parallel Moves Lemma (PM) [9, 4], we can construct the following diagram in R, where $t^r \twoheadrightarrow t_1'$ is an R-reduction that contracts all residuals of u in t^r, which are disjoint, if any.

$$\begin{tikzcd}
t \arrow[r, "u"] \arrow[d, two heads] \arrow[dr, phantom, "\text{PM}"] & t_1 \arrow[d, two heads] \\
t^r \arrow[r, two heads] & t_1'
\end{tikzcd}$$

Let $t_1' \twoheadrightarrow t_1^r$ be a reduction that contracts all r-redexes created by contracting the disjoint residuals of u along $t^r \twoheadrightarrow t_1'$ (in fact, there are no other r-redexes in t_1'). By the decomposition property of R^r-steps mentioned above, $t^r \twoheadrightarrow t_1^r$ in R^r if u is not an r-redex, and $t^r = t_1^r$ otherwise. By the induction assumption, t_1^r is normalizable in R^r, hence so is t^r.

$$\begin{tikzcd}
t \arrow[r, "u"] \arrow[d, two heads] \arrow[dr, phantom, "\text{PM}"] & t_1 \arrow[d, two heads] \arrow[rr, two heads] & & t_n \arrow[dd, two heads, "\emptyset"] \\
t^r \arrow[r, two heads] \arrow[dr] & t_1' \arrow[d, two heads] & \text{Ind} & \\
& t_1^r \arrow[rr, two heads] & & t_n^r = t_n
\end{tikzcd}$$

$\square$

Theorem 1. Normalization is decidable in any RPS R.

Proof. By induction on the number of rules in R. By Lemma 1, we can assume that R contains an irreducible rule, r. By Lemma 2, a term t has a normal form in R iff t^r has a normal form in R^r, and we conclude (since R^r has fewer rules than R). $\square$

3 Concluding remarks

Decidability of normalization in RPSs can also be derived from an advanced result of Nagaya and Toyama [10, page 264], stating that for a left-linear growing TRS R and a regular tree language L, the set of ground terms s

such that $s \twoheadrightarrow_R t$ for some $t \in L$ is regular. Here R is growing if for any rule $r \in R$ and any variable x that occurs both in left- and right-hand sides of r, x occurs in the left-hand side of r at depth 1 (i.e., just below the root symbol). RPSs are clearly growing.

Unlike the proof in [10], our proof gives an algorithm for transformation of RPSs into simpler and more efficient ones: Given an RPS R and a term t in R, we can construct (using Lemma 2) an irreducible RPS R' such that any term in R is normalizable in R iff it is normalizable in R', and the normal forms coincide. Note that $R' = R'_{irr} \cup R'_{red}$, where all rules in R'_{irr} are irreducible, and the right-hand sides of rules in R'_{red} are not normalizable. (Clearly, it does not make sense to compute R'_{red}-redexes, since such redexes do not have normal forms.) Then, if t contains an R'_{red}-redex that is not in an erased argument of an R'_{irr}-redex, then t has no normal form in R or R'. Otherwise, we normalize t w.r.t. R'_{irr} (e.g., using the innermost essential strategy, which is optimal in orthogonal TRSs in general [6]; a subterm of t is essential w.r.t. R_{irr} iff it is not in an erased argument of an R'_{irr}-redex in t). The obtained R_{irr}-normal form is also the normal form of t in R. (Cf. [1], where family-reductions are designed to achieve optimal evaluation of RPSs.)

We note that the proof presented in this work is based on the fact that the only redexes that can be created by reducing a redex are present explicitly already in the right-hand side of the applied rewrite rule. We therefore expect that the proof can be generalized to *Higher Order Recursive Program Schemes* [8] and *Persistent TRSs* [7] and *ERSs* [5, 8, 3]. Higher Order RPSs correspond to *contracting symbols of types IV and IV′* studied in [11, 12].

References

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

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

3. Glauert J., Kesner D., Khasidashvili Z. Expression Reduction Systems and Extensions: An Overview. *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, A. Middeldorp, V. van Oostrom, F. van Raamsdonk, R. de Vrijer, eds. Springer LNCS 3838, 2005, p. 496-553.

4. Huet G. and Lévy J.-J. Computations in orthogonal rewriting systems. *Computational Logic, Essays in Honor of Alan Robinson*, J. - L. Lassez and G. Plotkin, eds. MIT Press, 1991, p. 394-443.

5. Khasidashvili Z. Expression reduction systems. *Proceedings of I. Vekua Institute of Applied Mathematics of Tbilisi State University*, v. 36, 1990, p. 200-220.

6. Khasidashvili Z. Optimal normalization in orthogonal term rewriting systems. *Proceedings of the* 5^{th} *International Conference on Rewriting Techniques and Applications*, RTA'93, C. Kirchner, ed., Springer LNCS, vol. 690, 1993, p. 243-258.

7. Khasidashvili Z. On the equivalence of persistent term rewriting systems and recursive program schemes *Proceedings of the* 2^{nd} *Israel Symposium on Theory of Computing and Systems*, ISTCS'93, IEEE Computer Society Press, 1993, p. 240-249.

8. Khasidashvili Z. On higher order recursive program schemes. *Proceedings of the* 19^{th} *International Colloquium on Trees in Algebra and Programming*, CAAP'94, S. Tison, ed., Springer LNCS, vol. 787, 1994, p. 172-186.

9. Klop J.W. Term rewriting systems. *Handbook of Logic in Computer Science*, vol. 2, S. Abramsky, D. Gabbay, and T. Maibaum eds., Oxford University Press, 1992, p. 1-116.

10. Nagaya T. and Toyama Y. Decidability for left-linear growing term rewriting systems. *Proceedings of the* 10^{th} *International Conference on Rewriting Techniques and Applications*, RTA'99, P. Narendran and M. Rusinowich, eds., Springer LNCS, vol. 1631, 1999, p. 256-270.

11. Pkhakadze Sh. Some problems of the notation theory. (In Russian) *Tbilisi University Press*, Tbilisi, 1977.

12. Pkhakadze Sh. A N. Bourbaki type general theory and the properties of contracting symbols and corresponding contracted forms. *Georgian Mathematical Journal*, Vol. 6, No. 2, 1999, 179-190.

13. Terese. Term Rewriting Systems. *Cambridge Tracts in Theoretical Computer Science*, Volume 55, 2003.