Information Processing Letters 20 (1985) 83–85
North-Holland

15 February 1985

ON THE CORRESPONDENCE BETWEEN TWO CLASSES OF REDUCTION SYSTEMS

Satish R. THATTE

Department of Computer and Communication Sciences, 2500 East Engineering, University of Michigan, Ann Arbor, MI 48109, U.S.A.

Communicated by G.R. Andrews
Received 26 January 1984
Revised 12 June 1984

Keywords: Term-rewriting, constructors, semantics

1. Introduction and preliminaries

Equationally specified reduction systems [4,7,10] are a model of computation that is rather attractive for certain applications, e.g., for defining primitive functions for new types in applicative programming. At least two distinct classes of such systems have appeared in the literature. Huet and Levy [4], O'Donnell [7] and Rosen [9] deal with what we call class C_I systems (C_I-systems, for short), while the programming language HOPE [1], and the work in [6,10] exemplifies class C_{II} systems (C_{II}-systems, for short). C_I is designed to be as inclusive as possible while ensuring that its members satisfy the Church–Rosser property [9]. The notion of constructors, on which C_{II} is based, is akin to a similar notion in algebraic specifications [2]. On the face of it, class C_I seems to be strictly larger than class C_{II}, and in a sense it is. However, it turns out that there is a natural correspondence between the two classes which permits any C_I-system to be *embedded* without change of behavior into a C_{II}-system, as shown in this article. We believe that difficult problems such as sequential evaluation strategies and construction of semantic models can be solved more easily for C_{II}, but the solutions are applicable to C_I via the transformation.

A reduction system is based on a non-empty *ranked alphabet* $\Sigma = \Sigma_0 \cup \cdots \cup \Sigma_n$, which contains all function symbols in the system. T_Σ denotes the set of all (ground) terms formed with symbols in Σ. In addition, terms may include nullary *variables*. Given a term $f(t_1, \ldots, t_k)$, the occurrences of function symbols in $t_1, \ldots, t_k$ are said to be *inner* occurrences in relation to this term. A term is said to be *linear* iff no variable occurs more than once in it. A *reduction system* R is simply a set $\{E_1, \ldots, E_m\}$ of *equations*, where each E_i is an ordered pair $\langle \ell_i, r_i \rangle$ of terms. A path p in a term t is a possibly empty string of integers. We say that p *reaches* subterm t/p in t. The empty string Λ reaches the term itself, the string "k" reaches the k*th* argument, "km" reaches the m*th* argument of the k*th* argument etc. Finally, t[p = w] denotes the term obtained by replacing t/p at p by w. The first-order unification algorithm [9] is denoted by UNIFY. The reduction relation $\rightarrow$ and its reflexive transitive closure $\rightarrow^*$ have their usual significance in the context of term-rewriting systems. Our main result is concerned with the notion of the 'meaning' of functions in Σ, as determined by R. The following definition expresses the most comprehensive operational meaning of a function in a reduction system. Let $\mathscr{P}(S)$ denote the *powerset* of S. For a reduction system R operating in T_Σ, the *meaning function* μ_R maps each symbol $f \in \Sigma_k$ to a function $\mu_R(f) : (T_\Sigma)^k \rightarrow \mathscr{P}(T_\Sigma)$, such that

$$\mu_R(f)(t_1, \ldots, t_k) = \{y \mid f(t_1, \ldots, t_k) \rightarrow^* y \text{ in } R\}.$$

2. Main definitions and results

The two classes of reduction systems of interest to us are defined by distinct sets of restrictions. Actually, three of the four restrictions are common to both, and only the fourth restriction distinguishes them. The three common restrictions are the following.

K1. Each ℓ_i, $1 \leqslant i \leqslant m$, must be linear.

K2. Each variable that occurs in r_i must also occur in ℓ_i, $1 \leqslant i \leqslant m$.

K3. Given any i, j such that $1 \leqslant i, j \leqslant m$, if $\text{UNIFY}(\ell_i, \ell_j)$ succeeds yielding α, then $r_i\alpha = r_j\alpha$.

The fourth and last restriction for a C_I-system is the following.

K4. If u is a subterm of ℓ_i, $u \neq \ell_i$, and u is not a variable, then $\text{UNIFY}(u, \ell_j)$ fails for $1 \leqslant j \leqslant m$. *Note*: $i = j$ is possible.

In these and all future uses of UNIFY we assume that the variables used in the two terms are disjoint. This may be accomplished by renaming without loss of generality.

In order to define C_{II} we need a preliminary definition. In our system R, let $\ell_i = f_i(t_{i1}, \ldots, t_{in_i})$, $1 \leqslant i \leqslant m$. Let $F = \{f_i \mid 1 \leqslant i \leqslant m\}$. The last restriction for a C_{II}-system is the following.

K5. No symbol in F occurs in any t_{ij}, $1 \leqslant j \leqslant n_i$, $1 \leqslant i \leqslant m$.

The symbols in $\Sigma - F$, i.e., those not defined by equations, are called *constructor* symbols. The strict division between constructor and nonconstructor symbols in C_{II}-systems resembles the strict division between predicate and function symbols in logic programming [5].

It is easy to show that **K5** implies **K4**, i.e., that C_{II} is a subset of C_I. We wish to show that, for every system R in C_I, there is a corresponding system $R^\#$ in C_{II} such that the behavior of $R^\#$ parallels that of R within the domain of discourse for R.

To show this, suppose R belongs to C_I. With each $f \in F$ associate a new (constructor) symbol c_f. Let $\Sigma^\# = \Sigma \cup \{c_f \mid f \in F\}$. Let t′ denote the term t with every *inner* occurrence of $f \in F$ replaced by c_f and t″ the term with *all* occurrences so replaced. $R^\#$ is the smallest system which satisfies the following two assertions:

(1) If $\langle \ell, r\rangle \in R$, then $\langle \ell', r\rangle \in R^\#$.

(2) Whenever $u = f(t_1, \ldots, t_k)$, $f \in F$, is a *proper* subterm of a left-hand side in R, $\langle u', u''\rangle \in R^\#$.

Example (Equations are written as $\ell = r$ for readability). Let R be:

(1) $f(g(con(nil), x)) = r1$,
(2) $f(g(con(f(nil)), x)) = r2$,
(3) $g(nil, x) = r3$.

Then $R^\#$ is:

(1) $f(c_g(con(nil), x)) = r1$,
(2) $f(c_g(con(c_f(nil)), x)) = r2$,
(3) $g(nil, x) = r3$,
(4) $g(con(nil), x) = c_g(con(nil), x)$,
(5) $g(con(c_f(nil)), x) = c_g(con(c_f(nil)), x)$,
(6) $f(nil) = c_f(nil)$.

$R^\#$ clearly satisfies **K5** since every left-hand side in it is of the form t′. Moreover, it belongs to C_{II} since **K1** and **K2** are unaffected, and **K3** is satisfied since none of the new left-hand sides required by assertion (2) can be unified with those required by assertion (1) since R satisfies **K4**.

It remains to demonstrate the equivalence of behavior between R and $R^\#$. $R^\#$ is expected to deal with terms in $T_{\Sigma^\#}$ which contains T_Σ as a subset. The map $h: T_{\Sigma^\#} \to T_\Sigma$ is defined as $h(e) = d$ where d is obtained from e by replacing every occurrence of c_f in e by f, for every $f \in F$. Clearly, $h(t') = h(t'') = t$.

Lemma 1. *Given* t_1 *and* t_2 *in* $T_{\Sigma^\#}$, $t_1 \to t_2$ *in* $R^\#$ ***only if*** $h(t_1) \to^* h(t_2)$ *in* R.

Proof (sketch). If $t_1 \to t_2$ by an equation of the form $\langle u', u''\rangle$, then $h(t_1) = h(t_2)$. If the equation is of the form $\langle \ell', r\rangle$, then $h(t_1) \to h(t_2)$ by $\langle \ell, r\rangle$ in R. □

Lemma 2. *Given* t_1 *and* $t_2 \in T_\Sigma$, $t_1 \to t_2$ *in* R ***only if*** $t_1 \to^* t_2$ *in* $R^{\#}$.

Proof (sketch). Suppose the equation $E = \langle \ell, r \rangle$ is used in R to derive $t_1 \to t_2$. There is a p such that $t_2 = t_1[p = r\alpha]$, and $t_1/p = v = \ell\alpha$. If ℓ contains any proper subterms that satisfy assertion (2) above, then the corresponding subterms of v can be reduced using the equations introduced by that assertion in an innermost first fashion until the reduced version of v becomes an instance of ℓ'. The equation $\langle \ell', r \rangle$ can then be used to obtain t_2. □

Recall that $R^{\#}$ operates in $T_{\Sigma^{\#}}$ in the context of the definition of meaning functions. In the Theorem below, h has been extended pointwise to act on sets of terms.

Theorem. *For all* $f \in \Sigma_k$, $0 \leqslant k \leqslant n$, $\mu_R(f) \subseteq h \times \mu_{R^{\#}}(f)$ *in the sense that, for each* $(t_1, \ldots, t_k) \in (T_\Sigma)^k$,

$$\mu_R(f)(t_1, \ldots, t_k) = h(\mu_{R^{\#}}(f)(t_1, \ldots, t_k)).$$

Proof. We have

$$\mu_R(f)(t_1, \ldots, t_k) \subseteq h(\mu_{R^{\#}}(f)(t_1, \ldots, t_k))$$

by Lemma 2,

$$\mu_R(f)(t_1, \ldots, t_k) \supseteq h(\mu_{R^{\#}}(f)(t_1, \ldots, t_k))$$

by Lemma 1. □

3. Conclusions

We have demonstrated the possibility of simulating any C_I-system with a C_{II}-system. The construction is useful in many practical situations where only a small number of left-hand sides violate **K5**, and hence the size of $R^{\#}$ increases only modestly over that of R. In the worst case, if all the original left-hand sides are made up almost entirely from symbols in F, the size of $R^{\#}$ could be quadratically larger than that of R. However, $R^{\#}$ is not always the smallest C_{II}-simulation of R. For instance, in the Example illustrating the construction of $R^{\#}$, equations (4) and (5) could be replaced with the single equation

$$g(con(x), y) = c_g(con(x), y),$$

considerably reducing the size.

The equations of the form $\langle u', u'' \rangle$ are obviously responsible for the expansion of the size of $R^{\#}$ over R. The terms u′ are patterns for what Hoffmann and O'Donnell call 'root-stable' terms [3]. Root-stability of a term t means t cannot become a redex. This situation is easy to detect in *sequential* evaluation, hence in the sequential case the equations $\langle u', u'' \rangle$ can be dispensed with. The question of optimal simulation of C_I-systems in the nonsequential case is still open.

Acknowledgment

I would like to thank Michael O'Donnell and Fritz Ruehr whose comments on previous draft of the paper led to a considerable improvement of the presentation.

References

[1] R.M. Burstall, D.B. MacQueen and D.T. Sanella, HOPE: An experimental applicative language, in: LISP-80 Conference, Stanford, CA, 1980.

[2] J.A. Goguen, J.W. Thatcher, E.G. Wagner and J.B. Wright, Abstract data types as initial algebras, and correctness of data representations, in: Proc. Conf. on Computer Graphics, Pattern Recognition, and Data Structures, 1975.

[3] C.M. Hoffmann and M.J. O'Donnell, Implementation of an interpreter for abstract equations, in: Proc. 11th POPL, Salt Lake City, 1984.

[4] G. Huet and J.-J. Levy, Computations in nonambiguous linear term rewriting systems, Tech. Rept. 359, INRIA, France, 1979.

[5] R. Kowalski, Logic for Problem Solving, Artificial Intelligence Series (North-Holland, Amsterdam, 1979).

[6] C.F. Nourani, Abstract implementations and their correctness proofs, J. ACM 30 (1983) 343.

[7] M.J. O'Donnell, Computing in Systems Described by Equations, Lecture Notes in Computer Science 58 (Springer, Berlin, 1977).

[8] J.A. Robison, A machine-oriented logic based on the resolution principle, J. ACM 12 (1965) 23–41.

[9] B.K. Rosen, Tree-manipulating systems and Church–Rosser theorems, J. ACM 20 (1) (1973).

[10] S.R. Thatte, Algebraic types in lazily evaluated applicative languages, Ph.D. Dissertation, University of Pittsburgh, 1982.