Theoretical Computer Science 61 (1988) 83-92
North-Holland

NOTE

IMPLEMENTING FIRST-ORDER REWRITING WITH CONSTRUCTOR SYSTEMS

Satish THATTE

Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109, U.S.A.

Communicated by M. Nivat
Received September 1987

Abstract. In previous work [8] we described a technique for translating a general class of equational rewriting systems called *regular systems* [3] to constructor-based systems. This paper extends the previous results by showing that the translation technique described in [8] is much more generally applicable under a slight restriction of rewriting in the translated version of the system. The three additional classes we consider are those of semiregular systems (a generalization of regular systems), canonical systems, and arbitrary call-by-value confluent systems. It is shown that in each case the translation preserves the defining characteristics and normal forms of the class.

1. Introduction

Among the many examples of the use of first-order term-rewriting systems in functional programming [1, 3, 7, 11], only a few [1, 3] use systems that do not require argument patterns in function definitions to be built from a special class of *constructor* symbols. The reason for the popularity of constructor-based definitions is the simplicity and efficiency with which they can be implemented. In [8] we described a technique which can "compile" any program in the broadest class (*regular systems*) considered by Hoffmann-O'Donnell [3] to an equivalent constructor-based program. The basic idea involved can be seen by examining the following two equations, which are a part of an equational LISP interpreter in [2].

(1) eval(cons(x, y), z) = apply(eval(x, y), y, z),

(2) apply(eval(car, z), cons(x, y), $z1$) = car(eval(x, $z1$)).

The interesting point is that "eval" is used both as a function in equation (1), and as a constructor of data patterns in equation (2). The distinction between constructors and defined functions appears to have mainly computational/constructive significance, and increased clarity and economy of expression is often gained by ignoring it. For example, the interpreter mentioned above is quite close in appearance to McCarthy's original equations defining the semantics of LISP. These

gains are achieved at the cost of a considerable increase in the complexity of implementation, especially in pattern matching since it is not necessarily known whether an application needs to be evaluated.

In regular systems, although such dual use of symbols is permitted, the two kinds of applications are actually separated by the rule that left-hand sides of equations must not overlap. It is therefore easy to eliminate dual use by introducing new constructor symbols for symbols with dual use, and new equations to detect and convert the constructor-like occurrences to the new symbols. The technique can be used in the form of a preprocessor to achieve the generality of regular systems with a constructor-based implementation, at the cost of an expansion in the total size of patterns that is quadratic in the worst case, but relatively small in realistic cases such as the LISP interpreter mentioned above. Recent programming systems based on term rewriting [1] are beginning to use classes of rewriting systems more general than regular systems. It is therefore of interest to investigate how far the translation technique can be generalized.

As the two main lemmas in [8] (restated in Section 2) show, the translated system is able to faithfully mimic the behavior of the original system, whether the latter is regular or not. The limits on the application of the method lie in other properties of the translated system. In the case of regular systems, the translation was shown to preserve regularity. Preservation of the defining characteristics of a class of systems is clearly a desirable, and often sufficient condition for applicability. We shall therefore use it as the primary criterion. *Semiregular* systems are the first and simplest new class we consider. For our purposes, the essential aspect of regularity is nonoverlapping left-hand sides. The other rules defining regularity, such as the linearity of left-hand sides, can be viewed as being intended to ensure the Church-Rosser property. The class can therefore be broadened to semi-regular systems by retaining the nonoverlapping rule, and replacing the others by the property they are intended to ensure. In Section 3, we show that the translation preserves the property of semiregularity. *Canonical* (confluent terminating) systems are a widely used and well-behaved class that does *not* guarantee nonoverlapping left-hand sides. It turns out that although canonicity is not preserved by the translation under arbitrary rewriting, it *is* preserved if rewriting in the translated system respects a weak ordering on the new equations. Since the completeness of mimicry mentioned above is valid only under arbitrary rewriting, a notion of "sufficiently complete mimicry" is needed for this case. We use the obvious one requiring that all normal forms of the original system can be produced in the new one. These results for canonical systems are proved in Section 4. Arbitrary confluent systems constitute the most general class one could consider. It is not surprising that in such a general setting, none of the results for the previous classes hold. However, in the practically important case where the original system follows a call-by-value evaluation strategy, the translation does preserve confluence and normal forms under the same weak restriction on rewriting used in the case of canonical systems. This is shown in Section 5.

2. Existing results

We first recall some terminology and notation—with small changes—from [8]. If Σ is a ranked alphabet, T_Σ will denote all ground Σ-terms. 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. A rewriting system R (based on Σ) is simply a set of rules, where each rule is a pair $\langle l, r\rangle$ of Σ-terms. A path P is a nonempty string of integers used to "address" subterms of a term. The empty string addresses the term itself, the string "23" reaches the third argument of the second argument, etc. We use t/P to denote the subterm of t reached by P, and $t[P = w]$ to denote the result of replacing that subterm by the term w. The catenation of paths P and Q will be denoted by $P.Q$. The rewrite relation $\to$ and its reflective transitive closure $\to^*$ will have their usual significance in rewrite systems.

In relation to a given system R, we partition Σ into the two sets C and F of *constructors* and *defined functions* respectively, where F contains all those symbols which occur at the head of some left-hand side l of a rule in R, and C contains the rest. The translation of any system R can now be stated. Let $\Sigma' = \Sigma \cup \{c_f | f \in F\}$. 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. The translation R' of R is the smallest system satisfying the following two assertions:

(1) if $\langle l, r\rangle \in R$, then $\langle l', r\rangle \in R'$;

(2) whenever a *proper* subterm u of a left-hand side in R has a symbol from F at its head, then $\langle u, u''\rangle \in R'$.

We shall partition R' into R'_1 and R'_2, where the former contains rules mandated by (1) and the latter those by (2). Σ' can be partitioned into C' and F' in the way described for Σ, and the new symbols c_f are all in C', i.e., they are constructors. The following lemmas are proved in [8], where H is a homomorphism that projects terms in $T_{\Sigma'}$ back to T_Σ by simply replacing all new symbols c_f with the corresponding f.

Lemma 2.1. *Given t and u in $T_{\Sigma'}$, $t \to u$ in R' only if $H(t) \to^* H(u)$ in R.*

Lemma 2.2. *Given t and u in T_Σ, $t \to u$ in R only if $t \to^* u$ in R'.*

These lemmas show that for *any* rewriting system R, R' exactly mimics R as seen "through" H. It is therefore a little surprising that R' does *not* necessarily preserve any particular property of R such as confluence or sequentiality. The difficulty is created by reductions via R'_2 which are invisible when seen through H but which may divert the computation sequence in strange ways. One therefore needs to show separately that properties that are considered essential are preserved by the translation. For regular systems it is easy to prove [8] that if R is a regular system, then R' is a regular constructor system. Regularity is basically an easily decidable sufficient condition for *confluence*, which (defined as a property of *any* rewrite relation $\to$)

states that whenever $t \to^* u_1$ and $t \to^* u_2$ for any term t, there is a term v such that $u_1 \to^* v$ and $u_2 \to^* v$. This property ensures the uniqueness of normal forms (the Church-Rosser property) and is usually taken to be essential for computational use of rewriting. Confluence is *not* preserved by our translation in general, as in the example below.

System R	System R_1'	System R_2'
$f(g(x), h(y)) \to 0$	$f(c_g(x), c_h(x)) \to 0$	$g(x) \to c_g(x)$
$f(1, 2) \to 0$	$f(1, 2) \to 0$	$h(x) \to c_h(x)$
$g(x) \to 1$	$g(x) \to 1$	
$h(x) \to 2$	$h(x) \to 2$	

We can reduce $f(g(0), h(0))$ to at least two normal forms, namely, $f(1, c_h(0))$ and 0. The trouble is that a reduction using a rule in R_2' (e.g., $h(0) \to c_h(0)$) *commits* the resulting term to the enclosing redex of R_1'. If the other non-constructor components of the redex refuse to "cooperate" by also reducing via R_2', a deadlock results as in this example. Demanding a context-sensitive strategy for deciding the next redex to reduce destroys the effectiveness of the translation by reintroducing all the complexity of non-constructor systems.

3. Semiregular systems

The difficulty in the example above is created by the overlap between the patterns $f(g(x), h(y))$ and $h(x)$, producing a so-called critical pair. This is made precise in the following.

Definition 3.1. R is said to be *overlapping* iff for some left-hand sides l, l' ($l = l'$ is possible), $u = l/P$ is a proper subterm of l, u is not a variable, and Unify(u, l') succeeds returning substitution α. The term $l[P = u\alpha]$ is called a *superposition* of l and l'.

Confluent nonoverlapping systems have relatively simple rewriting behavior even when they are not regular in the technical sense; we therefore call such systems *semiregular*. In the rest of this section, R is assumed to be semiregular. Since there are no syntactic conditions describing semiregularity, its preservation must be proved by showing directly that the translation R' for a semiregular R is confluent. Actually, a slightly weaker property is proved because confluence may not be preserved for all terms in $T_{\Sigma'}$.

Definition 3.2. A term $u \in T_{\Sigma'}$ is said to be *reachable* (via R') iff $\exists t \in T_\Sigma$ such that $t \to^* u$ in R'.

In actual application, any rewriting process in a transformed program can only start with a reachable term, so it is enough if rewriting sequences starting with reachable terms are confluent, as they are for semiregular systems.

Lemma 3.3. *If t is reachable, and $H(t)=u\to v$ in R, then $\exists w$ such that $t \to^* w$ in R' and $H(w)=v$.*

Proof. Suppose the equation $\langle l, r\rangle$ is used in R to derive $u \to v$. Then there is a P such that $u/P = l\alpha$, and $v = u[P = r\alpha]$. The symbol at the head of t/P cannot be of the form c_f by the assumptions of reachability and nonoverlapping R. If l contains any non-constructor subterms, the corresponding subterms of t/P can be reduced if necessary by the rules of R_2' so that t/P becomes an instance of l' and the rule $\langle l', r\rangle$ can be used to achieve the desired result. □

Lemma 3.4. *If t, w are reachable and $u = H(t) = H(w)$, then there is a reachable v such that $t \to^* v$ and $w \to^* v$ in R'.*

Proof (*sketch*). It is easy to see that for any reachable t, $H(t) \to^* t$ in R_2'. Therefore $u \to^* t$ and $u \to^* w$ in R_2'. Since the system R_2' is always terminating and confluent, v is simply the normal form of u in R_2'. □

Theorem 3.5. *Rewriting sequences for reachable terms in R' are confluent if R is semi-regular.*

Proof. Let t be reachable, and $t \to^* u$ and $t \to^* v$ in R'. By Lemma 2.1, there are derivations $H(t) \to^* H(u)$ and $H(t) \to^* H(v)$ in R. Since R is confluent, $\exists w$ such that $H(u) \to^* w$ and $H(v) \to^* w$ in R. By Lemma 3.3, there are w_1 and w_2 such that $u \to^* w_1$ and $v \to^* w_2$ in R', and $w = H(w_1) = H(w_2)$. By Lemma 3.4, there is a w' such that $w_1 \to^* w'$ and $w_2 \to^* w'$ in R'. □

Since a constructor system is trivially nonoverlapping, this theorem proves that, in essence, semiregularity is preserved by our translation method.

4. Canonical systems

In addition to confluence, a canonical system is also assumed to be *terminating* (or Noetherian) which simply means that there are no infinite rewriting sequences in any of these systems. This class is particularly familiar in the context of theorem proving, and the famous Knuth-Bendix completion method [6] can be often used to mechanically transform a nonconfluent terminating system to an equivalent canonical one. In this section we describe a slight restriction of the notion of rewriting in a translation R' under which canonicity is shown to be preserved.

Arbitrary rewriting sequences in the translation of an overlapping confluent system are not always confluent, as shown by the example in Section 2. However, in practice it is not necessary to consider arbitrary sequences. In particular, the translation R' of a system R is naturally partitioned into R_1' and R_2', and the rules in the two parts

need not be considered equal. Actual computation is accomplished by the rules in R_1' with the rules in R_2' playing the auxiliary role of preparing redices for R_1'. We therefore consider a weakly ordered version of R' in which the rules in R_1' are preferred over those in R_2'. This gives rise to restricted notions of *ordered* rewriting steps and sequences which we denote by $\rightsquigarrow$ and $\rightsquigarrow^*$ respectively.

Definition 4.1. $t \rightsquigarrow t[P = w]$ in R' iff *one* of the following two conditions hold
(1) $t \to t[P = w]$ in R_1';
(2) t/P is *not* a redex in R_1' and $t \to t[P = w]$ in R_2'.
In other words, if a term is a redex in both R_1' and R_2', it can only be reduced by the former rule.

Notions such as confluence and reachability in R' must now be redefined using ordered rather than arbitrary rewriting. The restriction to ordered rewriting is not a particularly onerous one even for parallel implementations since rules in R_2' are apt to be implemented in a special way (e.g., with tagging) anyway. However, the simulation of R under ordered rewriting is significantly less complete for overlapping systems. Recall that Lemmas 2.1 and 2.2 are stated for arbitrary not ordered rewriting in R'. Lemma 2.1 continues to hold for ordered rewriting as a special case, but Lemma 2.2 does not hold for instances of superpositions (defined in Section 3). In addition to the preservation of canonicity, we must therefore show that the normal forms of R can be produced (modulo H) by ordered rewriting in R'. Both proofs hinge on the following lemma, which is a weaker version of Lemma 3.3.

Lemma 4.2. *If t_1 is reachable by ordered rewriting and $H(t_1) \to^* t_2$ in R, then there is a $u \in T_\Sigma$ such that $t_1 \rightsquigarrow^* u$ in R' and $t_2 \to^* H(u)$ in R.*

The proof of this lemma requires the following auxiliary result.

Proposition 4.3. *If t is reachable by ordered rewriting and $H(t) \to u$ in R, then there is a term v such that $t \rightsquigarrow^* v$ in R' and $H(t) \to v$ in R.*

Proof. Suppose the subterm $z = H(t)/P$ is replaced in the reduction $H(t) \to u$. If z is not an instance of any superposition, then $t \rightsquigarrow^* u$. Otherwise, suppose z is an instance of a superposition of l and l', where the subterm l/Q unifies with l'. Clearly, there is a w such that $H(t)/P.Q \to w$ in R. The proposition can now be reconsidered for this reduction. The process of lengthening the path in $H(t)$ cannot go on indefinitely, therefore a redex in $H(t)$ that is not an instance of a superposition will be reached eventually, yielding the required reduction $H(t) \to v$ in R. □

The most convenient way to prove Lemma 4.2 is to use a principle called *Noetherian induction* [4], which is stated as follows. Suppose T is a set of terms and $\to$ is a Noetherian (terminating) rewrite relation over T. Let $\to^+$ denote the transitive (not

reflexive) closure of $\rightarrow$ and, for any $t \in T$, let $\Delta^+(t)$ denote the set $\{u \mid t \rightarrow^+ u\}$. A predicate P over terms will be said to be $\rightarrow$-complete iff $\forall t \in T.\, [\forall u \in \Delta^+(t).\, P(u)] \Rightarrow P(t)$. The principle simply states that any $\rightarrow$-complete predicate holds for all of T. The idea is essentially to use normal forms as the basis cases and extend the predicate over the rest of the set by using the well-foundedness of $\rightarrow^*$. Specialized for our purposes, the principle states that a predicate P holds for the set of all terms reachable by ordered rewriting (which is Noetherian) if it can be shown to be $\rightsquigarrow$-complete for that set.

Proof of Lemma 4.2. Lemma 4.2 can be expressed in the form of the predicate $P(t)$ defined as

$$\forall u \in T_\Sigma.\quad H(t) \rightarrow^* u \;\Rightarrow\; \exists v \in T_\Sigma \cdot \text{ s.t. } [t \rightsquigarrow^* v \text{ and } u \rightarrow^* H(v)].$$

To show that this P is $\rightsquigarrow$-complete, suppose $H(t) \rightarrow^* u$. If $H(t) = u$, there is nothing to prove; otherwise, by Proposition 4.3, there is a w such that $t \rightsquigarrow^* w$ and $H(t) \rightarrow H(w)$ in R. Since $\rightarrow$ is confluent, there is a z such that $u \rightarrow^* z$ and $H(w) \rightarrow^* z$. Since $w \in \Delta^+(t)$ (for $\rightsquigarrow$), we may assume that there is a v such that $w \rightsquigarrow^* v$ and $z \rightarrow^* H(v)$, thus completing the proof. □

With Lemma 4.2, the normal forms of R and R' (via ordered rewriting) are easily shown to be essentially identical.

Theorem 4.4. *For any term $t \in T_\Sigma$, $t \rightarrow^* u$ and u is irreducible in R iff $t \rightsquigarrow^* v$, v is irreducible in R' and $H(v) = u$.*

Proof. The "if" part is immediate from Lemma 2.1. To see the "only-if" part, given $t \rightarrow^* u$ in R with u irreducible, we have $t \rightsquigarrow^* v'$ in R' and $H(v') = u$ by Lemma 4.2. This v' can be reduced to its normal form v by ordered rewriting and $H(v) = u$ by Lemma 2.1. □

Termination is obviously preserved by the translation, so preservation of canonicity reduces to the preservation of confluence.

Theorem 4.5. *If R is canonical, then ordered rewriting sequences for any term reachable by ordered rewriting are confluent.*

Proof. Suppose t is reachable by ordered rewriting starting with term $x \in T_\Sigma$ and $t \rightsquigarrow^* u$ and $t \rightsquigarrow^* v$. Suppose x' is the normal form of x in R and u' and v' are the normal forms of u and v by ordered rewriting in R'. By Theorem 4.4, $x' = H(u') = H(v')$. Since u' and v' are reachable and irreducible, $u' = v'$ by Lemma 3.4. □

5. Systems with confluent innermost rewriting

The next class we consider contains systems that are not confluent in the usual sense. This amounts to making a virtue of necessity since our translation does not preserve confluence in any useful sense for arbitrary confluent systems if the arbitrary rewriting relation in the original system must be simulated. The only useful application of the translation for nonterminating and overlapping systems we are aware of occurs when the original system uses a call-by-value or innermost-redex-first operational semantics. Only this restricted rewriting relation—which we shall call *innermost rewriting*—need therefore be confluent in both the original and translated systems. This condition is not comparable with the arbitrary confluence condition in that neither implies the other. For example, innermost rewriting is confluent in the system $\{f(g)\to 1, f(g)\to 2, g\to g\}$, while arbitrary rewriting is not. Conversely, arbitrary rewriting is confluent in $\{f(g)\to f(h), f(g)\to f(k), f(h)\to 1, f(k)\to 1, h\to h, k\to k\}$, but innermost rewriting is not. Less contrived examples of these phenomena no doubt occur in practice. The properties of this case are rather similar to those of semiregular systems. Lemmas 2.1, 2.2, 3.3 and 3.4 hold when restated for (ordered) innermost rewriting.

Let $\twoheadrightarrow$ denote innermost reduction in R and $\rightsquigarrow$ denote ordered innermost reduction in R', where $\twoheadrightarrow$ is assumed to be confluent. The definition of reachability is now based on $\rightsquigarrow$. The proofs of the following lemmas are not difficult, and are only sketched here.

Lemma 5.1. *Given t and u in T_Σ, $t\rightsquigarrow u$ in R' only if $H(t)\twoheadrightarrow^* H(u)$ in R.*

Proof. Similar to the proof of Lemma 2.1. □

Lemma 5.2. *Given t and u in T_Σ, $t\twoheadrightarrow u$ in R only if $t\rightsquigarrow^* u$ in R'.*

Proof. The restriction to ordered rewriting in $\rightsquigarrow$ is no restriction in simulating an innermost reduction in R since the subterms of such a redex cannot be redices of R'_1. The rest follows as in the proof of Lemma 2.2. □

The proofs of the next two lemmas depend on the following proposition.

Proposition 5.3. *If t is reachable and t/P has a symbol of the form c_f at its head, then t/P is an instance of a right-hand side in R'_2.*

Proof. Easy by induction on the length of the derivation reaching t. □

Lemma 5.4. *If t is reachable, and $H(t)=u\twoheadrightarrow v$ in R, then $\exists w$ such that $t\rightsquigarrow^* w$ in R' and $H(w)=v$.*

Proof. Suppose t/P is the redex used to derive $u\to v$. The symbol at the head of t/P cannot be of the form c_f by Proposition 5.3, given the assumptions of reachability and ordered rewriting. The rest follows as in the proofs of Lemmas 3.3 and 5.2. □

Lemma 5.5. *If t, w are reachable and $u = H(t) = H(w)$, then there is a term v such that $t \rightsquigarrow^* v$ and $w \rightsquigarrow^* v$ in R'.*

Proof. Suppose x and y are the normal forms of t and w in R_2' with the proviso that only the redices permitted to be reduced by ordered rewriting are considered. We claim that $x = y$ is the required v. The only way to contradict this is to assume that there is a P such that the symbol at the head of x/P is of tne form c_f, while that at the head of y/P is f itself. Let P be a maximal path which satisfies these conditions. Then by Proposition 5.3, y/P must be a redex of R_2'. Since the corresponding redex in the precursors of x/P was reduced and since P is maximal and x and y are reachable, y/P cannot be a redex of R_1', thus contradicting the assumption that y is in normal form with respect to ordered reduction in R_2'. □

Theorem 5.6. *The relation $\rightsquigarrow$ for reachable terms in T_Σ is confluent.*

Proof. Similar to the proof of the Theorem 3.5, replacing Lemmas 2.1, 2.2, 3.3 and 3.4 by Lemmas 5.1, 5.2, 5.4 and 5.5. □

6. Conclusions

Considering its relatively narrow origin, our translation technique turns out to be surprisingly robust for very different classes of rewriting systems under ordered rewriting (ordered rewriting is the same as arbitrary rewriting for semiregular systems). The main shortcoming of the technique is that certain important properties of the original system may not be fully preserved. A good example is *sequentiality* [5, 9]; a regular system may be strongly sequential even though the corresponding constructor system is not, if the former does not belong to a special class called simple systems.

The utility of the present technique is yet to be tested in practice, but it seems to offer the same advantages of simplicity and generality for first-order rewriting that combinators [10] do for λ-calculus dialects.

Acknowledgment

I would like to thank Michael O'Donnell for encouragement in pursuing this problem and Joseph Goguen for asking questions that led me to consider these extensions.

References

[1] K. Futatsugi et al., Principles of OBJ2, in: *Proc. 12th POPL*, New Orleans (1985) 52-66.

[2] C.M. Hoffmann and M. O'Donnell, Programming with equations, *ACM TOPLAS* 4(1) (1982) 83-112.

[3] C.M. Hoffmann and M. O'Donnell, Implementation of an interpreter for abstract equations, in: *Proc. 11th POPL*, Salt Lake City (1984) 111-121.

[4] G. Huet, Confluent reductions: abstract properties and applications to term-rewriting systems, *J. ACM* 27(4) (1980) 797-821.

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

[6] D. Knuth and P. Bendix, Simple word problems in universal algebra, in: J. Leech, ed., *Computational Problems in Abstract Algebra* (Pergamon Press, Oxford, 1970) 263-297.

[7] R. Milner, A proposal for standard ML, in: *Proc. ACM Symp. on LISP and Functional Programming*, Austin (1984) 184-197.

[8] S.R. Thatte, On the correspondence between two classes of reduction systems, *Inform. Process. Lett.* 20 (1985) 83-85.

[9] S.R. Thatte, A refinement of strong sequentiality for term-rewriting with constructors, *Inform. and Comput.* 72(1) (1987) 46-65.

[10] D. Turner, A new implementation technique for applicative languages, *Software Practice and Experience* 9 (1979) 31-49.

[11] D. Turner, An overview of Miranda, *SIGPLAN Notices* 21(12) (1986) 158-166.