COMMENTS ON "LOGIC PROGRAMMING WITH EQUATIONS"

PIER GIORGIO BOSCO, ELIO GIOVANNETTI, CORRADO MOISO, AND CATUSCIA PALAMIDESSI

▷ This note discusses the results of the compilational approach of equational logic programming developed by Van Emden and Yukawa, and compares them with similar results obtained by Bosco et al. and by Fribourg. We show that Van Emden and Yukawa's completeness result contains an inaccuracy, and we suggest how to correct it. ◁

1. INTRODUCTION AND TERMINOLOGY

The purpose of this note is to analyse the differences between some results contained in [1] and analogous results in [2, 3]. The three papers deal with the issue of how the basic execution mechanism of logic programming (i.e., SLD resolution) can be exploited to handle some kinds of equational theories: a set E of equations, which may be interpreted as a rewriting system (by orienting the equations), can be transformed into a set of Horn clauses to which ordinary SLD resolution can be applied to compute normal forms (instead of rewriting) and to solve equations (instead of narrowing [6]) w.r.t. the theory E.

This technique, which we call *flattening*, consists in transforming functional nestings into atom conjunctions in the bodies of the Horn clauses that are created. See [2, 3] for the definitions of the flattening algorithms for generic term rewriting systems, and see [1, 5, 7, 8] for systems with constructors (i.e., rewriting systems involving the distinction between defined-function symbols and data-constructor symbols). The only difference between the two versions is that in the latter case applications of constructors are not flattened.

Following the notation of [2, 3], we will indicate by R_{flat} the set of Horn clauses obtained by compiling the rewriting system R by means of the appropriate version

Address correspondence to P. G. Bosco, C.S.E.L.T., Centro Studi E Laboratori Telecomunicazioni, via Reiss Romoli 274, 10148 Torino, Italy.
Received June 1988; accepted April 1989.

655 Avenue of the Americas, New York, NY 10010 0743-1066/91/$3.50

of the flattening algorithm, by flatg($t1 = t2$) the analogously compiled form of the equational goal $t1 = t2$ [e.g., flatg($g(X) = f(Y)$) is ($?- g(X) = V1$, $f(Y) = V2$, eq($V1, V2$))], and by flatterm(t, Z) the flat form of the term t with Z as "outermost" new variable [e.g., flatterm($f(g(X)), Z$) is ($g(X) = V$, $f(V) = Z$)].

The main results of [1] concern theories with constructors. Using the above terminology, they can be rephrased as follows:

(i) *Soundness* for flattening + resolution (i.e., resolution applied to flattened programs and goals) w.r.t. narrowing: a resolution step on ?− flatterm(t, X) yields a new goal ?− flatterm(t', X), where t' is obtained from t through a narrowing step (Theorem 7.3).

(ii) *Completeness* for flattening + resolution w.r.t. reduction to normal form, in the case of terminating rewriting systems (with some further restrictions): every refutation of ?− flatterm(t, X) computes a substitution that binds X to a (strongly) canonical form of the ground term t, if any (Theorem 7.4).

(iii) *Declarative equivalence* between the theory E and E_{flat} under the standard equality axioms (Theorems 7.1, 7.2).

Results (i) and (ii) seem to be more general than the analogous results in [2, 3], since they apply to terminating, but possibly nonconfluent, systems. Moreover, (ii) does not require, in contrast with [2, 3], the use of the reflexivity axiom, with a consequent gain in efficiency.

As a matter of fact, (ii), as it is formulated in [1], does not hold unless a further restriction on the rewriting system is imposed, namely the condition that functions are *everywhere defined*: a defined function f is said to be everywhere defined if a reduction step can be applied to every term of the form $f(d_1, \ldots, d_n)$, where the d_i's are ground data terms (i.e., terms without definite-function applications in them). This merely amounts to dropping the words "of t" in line 7 of Theorem 7.4.

On the other hand, the generality of some results in [2, 3] exceeds the domain of canonical rewriting systems that was used as reference framework in those papers. In the following section, the connections between various properties stated in [1–4] are made clearer, despite the differences of the formulations.

2. THE CORRESPONDENCE BETWEEN RESOLUTION AND NARROWING OR REWRITING AND THE ROLE OF THE REFLEXIVITY AXIOM

Let R be a rewriting system. The properties of flattening + resolution proved in [2, 3] that are of interest in this discussion are roughly the following:

(1) *Soundness w.r.t. narrowing.* For every refutation of flatg($t = s$) w.r.t. $R_{\text{flat}} \cup \{X = X, \text{eq}(X, X)\}$, with answer substitution σ, there is a succeeding narrowing sequence in R computing the same substitution for the variables in $t = s$ (Property 3, Theorem 3, in [3]).

(2) *Completeness w.r.t. rewriting.* If $t \to_R^* s$ via a *basic* derivation [6], then there exits a substitution σ such that $R_{\text{flat}} \cup \{X = X\} \models \sigma$ flatterm(t, Z), and $\sigma Z = s$. Therefore, σ will be computed by the SLD-resolution of ?− flatterm(t, Z) (Lemma in [2]).

(3) *Completeness w.r.t. narrowing.* For every succeeding *basic* narrowing sequence [6] that computes a solution σ of the equation $t1 = t2$ in R, there

exists a (SLD) refutation w.r.t. $R_{\text{flat}} \cup \{X = X, \text{eq}(X, X)\}$ of the goal flatg($t1 = t2$) that computes the same solution (Theorem 2 in [3]).

These propositions and their proofs are independent of any assumption (in particular canonicity) on the rewriting system: they state that flattening + resolution, considered as an equation solving algorithm, is sound and complete w.r.t. basic narrowing (i.e., it finds all and only the solutions that basic narrowing does). They were given in [2, 3] within the framework of canonical systems, for, since only in this case is basic narrowing complete w.r.t. ordinary first-order logic semantics, only for canonical systems does (3) imply the ordinary completeness of flattening + resolution.

Owing to this independence of the properties of the rewriting system, proposition (1), if restricted to *theories with constructors*, is basically equivalent to result (i).

For nonconfluent rewriting systems the very basis of narrowing as equation-solving algorithm fails, because of the failure of the property that equality between terms is operationally equivalent to reduction to a common form. A completeness result analogous to (3) cannot therefore be obtained with the approach taken in [1], where terminating, but possibly nonconfluent, rewriting systems are considered. Only a much weaker property of completeness with respect to rewriting, similar to (2), can be established, namely (ii), which ensures that flattening + resolution is able to compute, for every term t, at least one of its normal forms.

There is still one difference between (ii) and (2), (3), and more generally between the two approaches. Namely, in (2), (3) the flattened form of the program has to be supplemented with the reflexivity axiom, which is, for convenience of some proofs and of the definition of flattening, represented by a pair of unit clauses, corresponding to two different roles in the derivation:

eq(X, X). Its application corresponds to the final syntactical unification step of succeeding narrowing sequences.

$X = X$. Its application corresponds to the possibility of not selecting a subterm during the narrowing or rewriting of a term.

In dealing with equality, the flattening compilation step replaces the use of the transitivity and substitutivity axioms, but does not replace reflexivity and symmetry. While the symmetry axiom can be eliminated owing to the confluence property, recourse to the reflexivity axiom (or to an equivalent rule) cannot be avoided in general if completeness is to be preserved, and therefore its inclusion in the set of Horn clauses used for resolution is mandatory.

On the other hand, as remarked in the introduction, in [1] the use of the reflexivity axiom is not required, but at the same time the restriction to signatures with constructors, not needed in [2, 3], is introduced. As a matter of fact, if we try to specialize the propositions (ii) or (2), (3) to the case of theories with constructors, we observe that the axiom $X = X$ is still necessary, as can be seen from the following example, drawn from [4]: Given the term rewriting system $R = \{f(X) = a,\ g(b) = a\}$, where a, b are constructors, in solving (verifying) the equation $f(g(a)) = a$, valid in R, which is flattened into $?- g(a) = V1,\ f(V1) = V2$, eq($V2, a$), resolution with $X = X$ is necessary to resolve (i.e., eliminate) $g(a) = V1$.

This is therefore a counterexample to the present form of Theorem 7.4: R is a terminating system, and every equation in R is of the form $f(t1,\ldots,tn)=s$, where no defined symbols occur in any of $t1,\ldots,tn$. Consider the ground term $f(g(a))$ (f is a defined symbol), whose normal form is a, strongly canonical (i.e., a data term): the SLD tree for $S'\cup\{?-G\}$, with $S'=\{f'(a,X), g'(a,b)\}$ and $G=(g'(V,a), f'(X,V))$, is finitely failed, while according to the theorem there should not be failures and every refutation should bind X to a.

From the results in [4] it follows that, in the case of theories with constructors, (ii) still holds even with the elimination of $X=X$, provided that functions are everywhere defined. We may conclude that Theorem 7.4 of [1] becomes correct—and equivalent to the result of [4]—only with the addition of the above restriction. In the terminology of [1] this amounts to requiring that all the canonical terms, and not just the canonical forms of the term to be reduced, be strongly canonical.

If, on the other hand, the condition that functions are everywhere defined is not imposed, then the need for the explicit presence of the reflexivity axiom requires that every system P flattened by the transformation adopted in [1] be supplemented with a clause $f'(f(X1,\ldots,Xn), X1,\ldots,Xn)$ for each symbol f of a function not everywhere defined. In this case the SLD tree for $P_{\text{flat}}\cup\{?-G\}$ for the left-to-right PROLOG strategy may present both finite failures and infinite computations.

3. CONCLUSIONS

If one does not want to impose restrictions on the theories with constructors, the reflexivity axiom must be kept [2,3,8]. The elimination of $X=X$ results in a remarkable reduction of the search space, but modifies the semantics of the underlying equational theory: a *strict* semantics for function and constructor applications must be adopted.

In K-LEAF (a logic + functional language based on nonterminating conditional rewriting systems with constructors) a reduction of the search space has been achieved, without loss of completeness, by replacing the axiom $X=X$ with an *outermost atom selection rule* plus an *elimination rule* [7]. Moreover, in many useful cases (namely, the so-called sequential systems) this strategy avoids finite failures in computing normal forms.

This work has been partially supported by EEC ESPRIT Project no. 415 (Parallel Architectures and Languages for Advanced Information Processing). We would like to thank Giorgio Levi for helpful discussions, and the referees for their useful suggestions.

REFERENCES

1. van Emden, M. and Yukawa, K., Logic Programming with Equations, *J. Logic Programming* 4:265–288 (Dec. 1987).
2. Bosco, P. G., Giovannetti, E., and Moiso, C., Refined Strategies for Semantic Unification, in: H. Ehrig et al. (eds.), *Proceedings of TAPSOFT '87*, Lecture Notes in Comput. Sci. 250, Springer-Verlag, 1987, pp. 276–290.

3. Bosco, P. G., Giovannetti, E., and Moiso, C., Narrowing vs. SLD-Resolution, *Theoret. Comput. Sci.*, July 1988, pp. 3–23.
4. Fribourg, L., SLOG: A Logic Programming Language Interpreter Based on Classical Superposition and Rewriting, in: *Proceedings of the 1985 Symposium on Logic Programming*, IEEE Computer Soc. Press, July 1985, pp. 172–184.
5. Fribourg, L., Prolog with Simplification, in: *Proceedings of France-Japan Symposium on Artificial Intelligence and Computer Science*, North-Holland, 1986.
6. Hullot, J.-M., Canonical Forms and Unification, in: *Proceedings of the 5th Conference on Automated Deduction*, Lecture Notes in Comput. Sci. 87, Springer-Verlag, 1980, pp. 318–334.
7. Levi, G., Bosco, P. G., Giovannetti, E., Moiso, C., and Palamidessi, C., A Complete Semantic Characterization of K-LEAF, a Logic Language with Partial Functions, in: *Proceedings of the 1987 Symposium on Logic Programming*, IEEE Computer Soc. Press, 1987, pp. 318–327.
8. Tamaki, H., Semantics of a Logic Programming Language with a Reducibility Predicate, in: *Proceedings of the 1984 Symposium on Logic Programming* IEEE Computer Soc. Press, 1984, pp. 259–264.