On termination of OBJ programs*

Salvador Lucas

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, E-46022 Valencia, Spain
e.mail: slucas@dsic.upv.es

Eager rewriting-based languages such as Lisp, OBJ*, CafeOBJ, ELAN, or Maude use innermost rewriting to evaluate initial expressions. A frequent problem here is nontermination. *Syntactic annotations* (i.e., associated to the arguments of symbols) have been used in OBJ2 [FGJM85], OBJ3 [GWMFJ00], CafeOBJ [FN97], or Maude [CELM96] as *replacement restrictions* to (hopefully) avoid nontermination. Within the program text, they are specified as sequences of integers in parentheses called *local strategies*. For instance, the following OBJ3 program:

```
obj EXAMPLE is
  sorts Sort .
  op 0 : -> Sort .
  op s : Nat -> Sort .
  op cons : Sort Sort -> Sort [strat: (1 0)] .
  op from : Sort -> Sort .
  op sel : Sort Sort -> Sort .
  var X Y L : Sort .
  eq sel(s(X),cons(Y,L)) = sel(X,L) .
  eq sel(0,cons(X,L)) = X .
  eq from(X) = cons(X,from(s(X)) .
endo
```

specifies a *explicit* local strategy for the list constructor `cons`. Symbols with no explicit strategy obtain a *default* strategy which is associated by the system (see [GWMFJ00]).

Local strategies serve to *completely guide* the *evaluation strategy* of OBJ programs[1]: when considering a function call $f(t_1, \ldots, t_k)$, only the arguments whose indices are present as *positive* integers in the list associated to the local strategy of f are evaluated (following the ordering which has been specified in the list). If an index 0 is found, then the reduction of the external function call is attempted. *Negative* indices indicate that the corresponding argument is to be evaluated 'on-demand', where a 'demand' is an attempt to match a pattern to the term that occurs in such an argument position [GWMFJ00,OF00].

According to this, the second argument of the list constructor `cons` will *never* be evaluated within a call to `cons`. The presence of such 'true' replacement restrictions is often invoked to justify that, even though the underlying execution mechanism is innermost rewriting, OBJ programs are able to achieve a *lazy* behavior thus avoiding nontermination [GWMFJ00]. For instance, with our OBJ3 program, the evaluation of `sel(s(s(0)),from(0))` should produce `s(s(0))`, even though the 'infinite list' `from(0)` is a part of the expression. The question is: can we (maybe automatically) *ensure* this? i.e., will the evaluation *terminate*? Unfortunately, no formal analysis about how a particular choice of replacement restrictions modify termination of OBJ (like) programs is available.

Term rewriting systems (TRSs) provide a suitable computational model for programs written in more sophisticated programming languages. Syntactic replacement restrictions can be associated to symbols f of a signature Σ by means of a *replacement map* $\mu : \Sigma \to \mathcal{P}(\mathbb{N})$ [Luc98] that discriminates the argument positions $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ on which we *can* perform replacements. For instance, the previous OBJ3 program can be associated to TRS

* This work has been partially supported by CICYT TIC 98-0445-C03-01.

[1] As in [GWMFJ00], by OBJ we mean OBJ2, OBJ3, or CafeOBJ.

```
sel(s(x),y:l) → sel(x,l)
sel(0,x:l)    → x
from(x)       → x:from(s(x))
```

and replacement map μ given by $\mu(:) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for all other symbol f.

Negative integers in local strategies (allowed in OBJ3 or CafeOBJ programs) can be managed using an *additional* replacement map μ_D containing their absolute values. Context-sensitive rewriting (*CSR*) [Luc98] and lazy (graph) rewriting (*LR*) [FKW00,KW95] provide operational models for using replacement restrictions specified by a single replacement map μ. We define *on-demand rewriting* (using two replacement maps μ and μ_D) as a generalization of *LR* that also includes *CSR*. Termination of *CSR* has been studied in some extent in [GM99,Luc96,Zan97]. We connect termination of on-demand rewriting (*ODR*) and termination of *CSR*.

Semantics of OBJ programs is usually given as a recursive evaluation function *eval* (mapping terms into their computed values, if any) rather than specifying the concrete rewrite steps leading to computed values (see [OF00]). Expressing OBJ computations as rewritings enables the use of the theory of Term Rewriting to analyze termination. We show that *CSR* and *ODR* can be used to model computations of OBJ programs thus giving a tool for analyzing termination. For instance, by using the results of [Zan97], we can show that context-sensitive reductions with the previous TRS under the replacement map μ are terminating. Thus, our OBJ3 program is terminating.

References

[CELM96] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In *Proc. 1st International Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science, Elsevier Sciences, 1996.

[FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 52-66, ACM Press, 1985.

[FKW00] W. Fokkink, J. Kamperman and P. Walters. Lazy Rewriting and Eager Machinery. *ACM Transactions on Programming Languages and Systems*, 22(1):45-86, 2000.

[FN97] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.

[GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

[GWMFJ00] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducig OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.

[KW95] J.F.Th. Kamperman and H.R. Walters. Lazy Rewriting and Eager Machinery. In J. Hsiang, editor, *Proc. of the 6th International Conference on Rewriting Techniques and Applications, RTA'95*, LNCS 914:147-162, Springer-Verlag, Berlin, 1995.

[Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, LNCS 1099:122-133, Springer-Verlag, Berlin, 1996.

[Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[OF00] K. Ogata and K. Futatsugi. Operational Semantics of Rewriting with the On-demand Evaluation Strategy. In *Proc of 2000 International Symposium on Applied Computing, SAC'00*, pages 756-763, ACM Press, 2000.

[Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.