On Graph Rewriting and Narrowing

Rachid Echahed and **Jean-Christophe Janodet**
Laboratoire LEIBNIZ-Institut IMAG, CNRS
46, avenue Felix Viallet F-38031 Grenoble - France
Email : First-name.Last-name@imag.fr

Abstract

Functional logic programming languages integrate, in a uniform way, the most important features of functional programming and logic programming paradigms. See [7] for a survey and [9, 8] for recent language propositions.

The operational semantics of most functional logic programming languages is based on first-order term rewriting and narrowing. These techniques are well mastered now and optimal strategies have been proposed in the literature, e.g., [2, 3].

However, in practice data structures are not always represented as first-order terms but rather as (cyclic) graphs. Some declarative programming languages such as Haskell, Clean or Life allow to work with graphs explicitly.

There are many reasons that motivate the use of graphs. They actually allow sharing of subexpressions which leads to efficient computations. They also permit to go beyond the processing of first-order terms by handling efficiently real-world data structures (e.g. Data bases).

We propose new optimal strategies for graph rewriting and narrowing which are good candidate for the implementation of functional logic languages. For this purpose, we first introduce the framework of admissible graph rewriting systems which could be seen as a natural extension to graphs of the well known class of orthogonal constructor-based term rewriting systems. A graph is admissible if none of its defined operations belongs to a cycle.

Orthogonal graph rewriting systems are not confluent in general. This is due to the presence of so-called collapsing rules. A rule is collapsing if its right-hand side is a variable. We show that admissible graph rewriting is a confluent relation even in the presence of collapsing rules. In addition, we show that admissible graph rewriting relation is confluent modulo bisimulation. Two graphs are bisimilar iff they represent the same rational tree. This last result is necessary, for instance, to show the soundness of the implementation of first-order terms as dags.

Afterwards, we define a sequential graph rewriting strategy by using Antoy's definitional trees [1] and show that the resulting strategy computes only needed redexes and develops optimal derivations w.r.t. the number of steps. Finally, we tackle the graph narrowing relation over admissible graphs and propose a sequential narrowing strategy which computes independent solutions and develops shorter derivations than most general graph narrowing.

The reader may find in [5, 4] the detailed definitions and proofs. An extended abstract could be consulted in [6].

References

[1] S. Antoy. Definitional trees. In *Proc. of the 4th Intl. Conf. on Algebraic and Logic programming*, pages 143–157. Springer Verlag LNCS 632, 1992.

[2] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Proc. 21st ACM Symposium on Principles of Programming Languages*, pages 268–279, Portland, 1994.

[3] S. Antoy, R. Echahed, and M. Hanus. Parallel evaluation strategies for functional logic languages. In *Proc. 14th International Conference on Logic Programming*, pages 138–152, Leuven, 1997.

[4] R. Echahed and J. C. Janodet. Introducing graphs in functional logic programming languages. Technical report, IMAG, 1997. Available via URL : ftp://ftp.imag.fr/pub/LEIBNIZ/ATINF/c-graph-narrowing.ps.gz.

[5] R. Echahed and J. C. Janodet. On constructor-based graph rewriting systems. Technical report, IMAG, 1997. Available via URL : ftp://ftp.imag.fr/pub/LEIBNIZ/ATINF/c-graph-rewriting.ps.gz.

[6] R. Echahed and J. C. Janodet. Admissible graph rewriting and narrowing. In *Proc. of the 1988 Joint International Conference and Symposium on Logic Programming*, Manchester, 1998.

[7] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

[8] M. Hanus (ed.). Curry: An integrated functional logic language. Available at `http://www-i2.informatik.rwth-aachen.de/~hanus/curry`, 1997.

[9] J. W. Lloyd. Combining functional and logic languages. In *Proc. of Int. Logic Programming Symposium*, pages 43–57, 1994.