Strategies and Analysis Techniques in Functional Program Optimization [*]

Santiago Escobar,
Dep. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n, E-46022 Valencia, Spain
E-mail: sescobar@dsic.upv.es

This paper abstracts the contents of the Ph.D. dissertation which has been recently defended by the author. Functional programs are commonly modelled by term rewriting systems. The execution of such programs often gives rise to useless, dangerous, and inefficient evaluation sequences which must be avoided in order to improve their computational behaviour. The thesis presents novel methods and techniques to optimize term rewriting systems either by defining new strategies to execute programs or by analyzing and transforming programs in order to improve their computational behaviour.

Keywords: Functional and Logic Programming, Term Rewriting Systems, Rewriting (Narrowing) Strategies, Program Analysis and Transformation

1. Extended Abstract of the Ph.D. Dissertation

Computer systems play an important role in the modern information society. However, the low quality of software and its low level of abstraction, inhibit the necessary confidence of final users and system developers. Correctness of computer programs by a mathematical theory of computation is the fundamental concern of the theory of programming and of its application in large-scale software engineering [13,17,18]. Formal methods provide software engineering with the suitable scientific and technological framework to become a real engineering, as predictable as civil or electrical engineering are [11,12,13]. Indeed, the use of declarative rule-based programming languages during all program development stages ensures that correct and certified formal methodologies are followed during the whole software production process.

Programs in declarative rule-based programming languages are often described as *term rewriting systems* [19]. Program execution consists of reducing (or *rewriting*) *input terms* to *output terms* by applying a sequence of rules. *Narrowing* [16] is an extension of rewriting for term rewriting systems which permits the instantiation of variables in input terms in order to enable rewriting steps on the instantiated expression. Rewriting (as well as narrowing) is generally undecidable, i.e. it is not possible to determine if a term rewrites (narrows) to another one. The reduction space associated to a given input term is huge due to the different possibilities for selecting subterms to reduce and the rules applicable to those subterms. This situation is even worse in the case of narrowing due to the instantiation process. In fact, the reduction (or narrowing) space usually contains useless (no interesting output expression is achievable), dangerous (no interesting output expression is guaranteed), and inefficient reduction sequences (an alternative more efficient sequence exists which delivers an equivalent outcome).

This thesis faces the problem of how to define efficient methods to improve the computational behavior of term rewriting systems, i.e. to shrink the reduction space associated to an input term by selecting which subterms and/or rules should be used for rewriting (or narrowing). In this thesis, we consider the following different approaches to optimize programs:

1. By dynamically selecting the allowed reductions in execution time due to the definition of appropriate reduction or narrowing strategies. We refine this approach into two cases:
   (a) The allowed reductions are automatically calculated from the program, i.e. the appropriate reduction strategy is inferred

[*]Work partially supported by MCyT under grants TIC2001-2705-C03-01, HA2001-0059, and HU2001-0019.

from the program. Here, we study the definition of optimal rewriting and narrowing strategies. Roughly speaking, we provide an incremental definition of the *outermost-needed narrowing* strategy [10] (the best known narrowing strategy for term rewriting systems) and we improve it, obtaining the *natural narrowing* strategy.

(b) The allowed reductions are provided by assertions of the programmer included into the program, i.e. the appropriate reduction strategy is induced by the programmer. Here, we study different aspects of the inclusion of syntactic strategy annotations [20] into term rewriting systems. First, we formulate the computational model associated to *on-demand strategy annotations*. Also, we provide two program transformations: the first one solves the incompleteness problem which is eventually introduced by the absence of some annotations; the other one encodes on-demand strategy annotations into standard annotations in order to introduce a flavour of laziness into languages which only consider standard strategy annotations.

2. By statically analyzing the program in order to discard (and remove) useless parts. We use program analysis and transformation techniques [1,14]. Here, we investigate a different semantic problem which is orthogonal to the definition of reduction or narrowing strategies. Roughly speaking, we analyze and remove the irrelevant data appearing in the program in the form of *redundant arguments of functions*, which produce inefficient reduction steps which cannot be avoided by a reduction strategy.

On the other hand, the proposed techniques are semantically correct, i.e. they are strong enough to avoid undesired sequences but not too much restrictive in order to preserve the sequences of interest. Furthermore, in order to prove the practicality of all the proposed techniques, several prototypes and systems have been implemented as part of the research work developed in this thesis: UPV-Curry and Natur (for item $1(a)$ above), OnDemandOBJ (for item $1(b)$), and RedArgs (for item 2).

The main achievements of this Ph.D. thesis have been presented in [2,3,4,5,6,7,8,9,15].

2. Formal Information on the Ph.D. Dissertation

Author: Santiago Escobar
Title: Strategies and Analysis Techniques for Functional Program Optimization
Language of Presentation: English
Advisors: Professors María Alpuente and Salvador Lucas
Date of Defense: October 31, 2003
Institution granting degree: Universidad Politécnica de Valencia, Spain
Evaluation Committee: Hélène Kirchner, Francisco López-Fraguas, José Meseguer, Isidro Ramos (chair), and Albert Rubio

Further information on the thesis can be found at the URL:

`http://www.dsic.upv.es/~sescobar`

Acknowledgements

The author wants to thank his advisors María Alpuente and Salvador Lucas for many stimulating discussions on the different topics of this thesis and for providing the necessary support to develop it.

References

[1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers, Principles Techniques and Tools.* Addison-Wesley, Reading, MA, 1986.

[2] M. Alpuente, R. Echahed, S. Escobar, and S. Lucas. Redundancy of arguments reduced to induction. In M. Comini and M. Falaschi, editors, *Proc. of the 11th Int'l Workshop on Functional and (Constraint) Logic Programming WFLP'02*, volume 76 of *Electronic Notes in Theoretical Computer Science*, pages 255–268, Grado (Italy), 2002. Elsevier Sciences Publisher.

[3] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. Improving on-demand strategy annotations. In Matthias Baaz and Andrei Voronkov, editors, *Proc. 9th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'02)*, volume 2514 of *Lecture Notes in Computer Science*, pages 1–18, Tbilisi (Georgia), 2002. Springer-Verlag, Berlin.

[4] M. Alpuente, S. Escobar, and S. Lucas. Incremental needed narrowing. In P. Tarau and K. Sagonas, editors, *Proc of the Int'l Workshop on Implementation of Declarative Languages, IDL'99*, pages 1–18, Paris (France), 1999.

[5] M. Alpuente, S. Escobar, and S. Lucas. UPV-Curry: an Incremental Curry Interpreter. In J. Pavelka, G. Tel, and M. Bartosek, editors, *Proc. of 26th Seminar on Current Trends in Theory and Practice of Informatics, SOFSEM'99*, volume 1725 of *Lecture Notes in Computer Science*, pages 327–335, Milovy (Czech Republic), 1999. Springer-Verlag, Berlin.

[6] M. Alpuente, S. Escobar, and S. Lucas. Correct and complete (positive) strategy annotations for OBJ. In F. Gadducci and U. Montanari, editors, *Proc. of the 4th Int'l Workshop on Rewriting Logic and its Applications, WRLA 2002*, volume 71 of *Electronic Notes in Theoretical Computer Science*, Pisa (Italy), 2002. Elsevier Sciences Publisher.

[7] M. Alpuente, S. Escobar, and S. Lucas. Removing redundant arguments of functions. In H. Kirchner and C. Ringeissen, editors, *9th Int'l Conference on Algebraic Methodology And Software Technology, AMAST 2002*, volume 2422 of *Lecture Notes in Computer Science*, pages 117–131, Reunion Island (France), 2002. Springer-Verlag, Berlin.

[8] M. Alpuente, S. Escobar, and S. Lucas. On-demand evaluation by program transformation. In J.L. Giavitto and P.E. Moreau, editors, *Proc. of the 4th Int'l Workshop on Rule-Based Programming, RULE 2003*, volume 86.2 of *Electronic Notes in Theoretical Computer Science*, Valencia (Spain), 2003. Elsevier Sciences Publisher.

[9] M. Alpuente, S. Escobar, and S. Lucas. OnDemandOBJ: a laboratory for strategy annotations. In J.L. Giavitto and P.E. Moreau, editors, *Proc. of the 4th Int'l Workshop on Rule-Based Programming, RULE 2003*, volume 86.2 of *Electronic Notes in Theoretical Computer Science*, Valencia (Spain), 2003. Elsevier Sciences Publisher.

[10] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Journal of the ACM*, volume 47(4), pages 776–822, 2000.

[11] F.P. Brooks. No silver bullet–essence and accident in software engineering. *Journal of Information Processing*, 1986.

[12] F.P. Brooks. The mythical man-month: Essays on software engineering. *Twentieth Anniversary Edition, Reading, MA: Addison-Wesley, 322 pp*, 1995.

[13] F.P. Brooks. Three great challenges for half-century-old computer science. *Journal of the ACM*, 50(1):25–26, 2003.

[14] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. of Fourth ACM Symp. on Principles of Programming Languages*, pages 238–252, 1977.

[15] S. Escobar. Refining weakly outermost-needed rewriting and narrowing. In D. Miller, editor, *Proc. of 5th Int'l ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'03*, pages 113–123, Uppsala (Sweden), 2003. ACM Press, New York.

[16] M. Hanus. The Integration of Functions into Logic Programming: From Theory to Practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

[17] T. Hoare. The verifying compiler: A grand challenge for computing research. *Journal of the ACM*, 50(1):63–69, 2003.

[18] J. McCarthy. Problems and Projections in CS for the Next 49 Years. *Journal of the ACM*, 50(1):73–79, 2003.

[19] TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, 2003.

[20] E. Visser. A survey of rewriting strategies in program transformation systems. In B. Gramlich and S. Lucas, editors, *Proc. of the 1st Int'l Workshop on Reduction Strategies in Rewriting and Programming, WRS 2001*, volume 57 of *Electronic Notes in Theoretical Computer Science*, Utrecht (The Netherlands), 2001. Elsevier Sciences Publisher.