The Oz Programming Model

(Extended Abstract)

Gert Smolka*

Programming Systems Lab
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
email: smolka@dfki.uni-sb.de

Abstract. The Oz Programming Model (OPM) is a concurrent programming model that subsumes functional and object-oriented programming as facets of a general model. This is particularly interesting for concurrent object-oriented programming, for which no comprehensive and formal model existed until now. There is a conservative extension of OPM providing the problem-solving capabilities of constraint logic programming. OPM has been developed together with a concomitant programming language Oz designed for applications that require complex symbolic representations, organization into multiple agents, and soft real-time control. An efficient, robust, and interactive implementation of Oz is freely available.

Computer systems are undergoing a revolution. Twenty years ago, they were centralized, isolated, and expensive. Today, they are parallel, distributed, networked, and inexpensive. However, advances in software construction have failed to keep pace with advances in hardware. To a large extent, this is a consequence of the fact that current programming languages were conceived for sequential and centralized programming.

A basic problem with existing programming languages is that they delegate the creation and coordination of concurrent computational activities to the underlying operating system. This has the severe disadvantage that the data abstractions of the programming language cannot be shared between communicating computational agents. Thus the benefits of existing programming languages do not extend to the central concerns of concurrent and distributed software systems.

Given this state of affairs, the development of concurrent programming models is an important research issue in Computer Science. A concurrent programming model must support the creation and coordination of multiple computational activities. Since concurrency is the result of a generalized control structure, it must be accommodated at the heart of a programming model.

* Supported by the BMBF (contract ITW 9105), the Esprit Basic Research Project ACCLAIM (contract EP 7195), and the Esprit Working Group CCL (contract EP 6028).

The development of simple, practical, and well-founded concurrent programming models turned out to be difficult. The main problem was the lack of a methodology and formal machinery for designing and defining such models. In the 1980's, significant progress has been made on this issue. This includes the development of abstract syntax and structural operational semantics; the development of functional and logic programming, two declarative programming models building on the work of logicians (lambda calculus and predicate logic); the development of CCS [6] and the π-calculus [7], two abstract concurrent programming models developed by Milner and others; and the concurrent constraint model [4, 8], a concurrent programming model that developed from application-driven research in concurrent logic programming [11] and constraint logic programming [2].

The talk reports on the Oz Programming Model, OPM for short, which has been developed together with the high-level concurrent programming language Oz. OPM is an extension of the basic concurrent constraint model, adding first-class procedures and stateful data structures. OPM is a concurrent programming model that subsumes higher-order functional and object-oriented programming as facets of a general model. This is particularly interesting for concurrent object-oriented programming, for which no comprehensive and formal model existed until now. There is a conservative extension of OPM providing the problem-solving capabilities of constraint logic programming. The resulting problem solvers appear as concurrent agents encapsulating search and speculative computation with constraints.

Oz and OPM have been developed at the DFKI since 1991. Oz [16, 13] is designed as a concurrent high-level language that can replace sequential high-level languages such as Lisp, Prolog and Smalltalk. There is no other concurrent language combining a rich object system with advanced features for symbolic processing and problem solving. First applications of Oz include simulations, multi-agent systems, natural language processing, virtual reality, graphical user interfaces, scheduling, time tabling, placement problems, and configuration. The design and implementation of Oz took ideas from AKL [3], the first concurrent constraint language with encapsulated search.

Since January 1995, an efficient, robust, and interactive implementation of Oz, DFKI Oz, is freely available for many Unix-based platforms (see remark at the end of this extended abstract). DFKI Oz features a programming interface based on GNU Emacs, a concurrent browser, an object-oriented interface to Tcl/Tk for building graphical user interfaces, powerful interoperability features (sockets, C, C++), an incremental compiler, and a run-time system with an emulator and a garbage collector.

DFKI Oz proves that an inherently concurrent language can be implemented efficiently on sequential hardware. Research on a portable parallel implementation for shared memory machines has started. More ambitiously, we have also begun work towards a distributed version of Oz supporting the construction of open systems.

There is a full paper [15] corresponding to this extended abstract. The Oz

Primer [14] is an introduction to programming in Oz. There is a simple formal model underlying OPM [12]. The extension of OPM to constraint programming and encapsulated search is the subject of [10, 9]. More on modeling objects in OPM can be found in [1]. Basic implementation techniques for Oz are reported in [5].

References

1. M. Henz, G. Smolka, and J. Würtz. Object-oriented concurrent constraint programming in Oz. In V. Saraswat and P. V. Hentenryck, editors, *Principles and Practice of Constraint Programming*, chapter 2, pages 27–48. The MIT Press, Cambridge, MA, 1995. To appear.
2. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *The Journal of Logic Programming*, 19/20:503–582, May-July 1994.
3. S. Janson and S. Haridi. Programming paradigms of the Andorra kernel language. In V. Saraswat and K. Ueda, editors, *Logic Programming, Proceedings of the 1991 International Symposium*, pages 167–186, San Diego, USA, 1991. The MIT Press.
4. M. J. Maher. Logic semantics for a class of committed-choice programs. In J.-L. Lassez, editor, *Logic Programming, Proceedings of the Fourth International Conference*, pages 858–876, Cambridge, MA, 1987. The MIT Press.
5. M. Mehl, R. Scheidhauer, and C. Schulte. An abstract machine for Oz. In *Proceedings of PLILP'95*, Utrecht, The Netherlands, Sept. 1995. LNCS, Springer-Verlag. To appear.
6. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, Berlin, Germany, 1980.
7. R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.
8. V. A. Saraswat. *Concurrent Constraint Programming.* The MIT Press, Cambridge, MA, 1993.
9. C. Schulte and G. Smolka. Encapsulated search in higher-order concurrent constraint programming. In M. Bruynooghe, editor, *Logic Programming: Proceedings of the 1994 International Symposium*, pages 505–520, Ithaca, New York, USA, 13–17 Nov. 1994. The MIT Press.
10. C. Schulte, G. Smolka, and J. Würtz. Encapsulated search and constraint programming in Oz. In A. Borning, editor, *Second Workshop on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, vol. 874, pages 134–150, Orcas Island, Washington, USA, 2-4 May 1994. Springer-Verlag.
11. E. Shapiro. The family of concurrent logic programming languages. *ACM Computing Surveys*, 21(3):413–511, Sept. 1989.
12. G. Smolka. A foundation for higher-order concurrent constraint programming. In J.-P. Jouannaud, editor, *1st International Conference on Constraints in Computational Logics*, Lecture Notes in Computer Science, vol. 845, pages 50–72, München, Germany, 7–9 Sept. 1994. Springer-Verlag.
13. G. Smolka. The definition of Kernel Oz. In A. Podelski, editor, *Constraints: Basics and Trends*, volume 910 of *Lecture Notes in Computer Science*, pages 251–292. Springer-Verlag, Berlin, Germany, 1995.
14. G. Smolka. An Oz primer. DFKI Oz documentation series, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, 1995.

15. G. Smolka. The Oz programming model. In J. van Leeuwen, editor, *Current Trends in Computer Science*, Lecture Notes in Computer Science, vol. 1000. Springer-Verlag, Berlin, Germany, 1995. To appear.
16. G. Smolka and R. Treinen, editors. *DFKI Oz Documentation Series.* German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, 1995.

Remark. DFKI Oz and related papers are available through anonymous ftp from `ps-ftp.dfki.uni-sb.de` or through WWW from

`http://ps-www.dfki.uni-sb.de/oz/`.