AKL

A Multiparadigm Programming Language

AKL

A Multiparadigm Programming Language Based on a Concurrent Constraint Framework

by
Sverker Janson

A Dissertation submitted
in partial completion of the requirements
for the Degree of Doctor of Philosophy
Uppsala University
Computing Science Department

Computing Science Department
Uppsala University
Box 311, S-751 05 Uppsala
Sweden

Swedish Institute of Computer Science
Box 1263, S-164 28 Kista
Sweden

Uppsala Theses in Computing Science 19
ISSN 0283-359X
ISBN 91-506-1046-5

SICS Dissertation Series 14
ISRN SICS / D--14--SE
ISSN 1101-1335

Doctoral thesis at Uppsala University 1994
from the Computing Science Department

Abstract

Janson, S., 1994. AKL—A Multiparadigm Programming Language.
Uppsala Theses in Computing Science 19. 212 pp. Uppsala. ISSN 0283-359X.
ISBN 91-506-1046-5.

New programming languages conceived by adding yet another permutation of new features on top of established languages offer only complexity and confusion to software developers. New basic principles are necessary that support the desired functionality using a minimum of concepts.

This thesis reports on an investigation into principles for combining the constraint solving and don't know nondeterministic capabilities of Prolog and the constraint logic programming languages with the process-describing capabilities of concurrent logic languages such as GHC. The result, AKL, is a coherent language supporting multiple programming paradigms, such as concurrent, object-oriented, functional, logic, and constraint programming. In addition, AKL offers a large potential for automatic parallel execution.

The operational semantics of AKL is captured by a computation model, involving rewriting of “semi-logical” expressions that form the computation states. Constraints are used to express data and interaction. The computation model is then augmented with control, giving an execution model, which demonstrates how to perform computations in a systematic manner. Finally, an abstract machine brings us close to a real implementation. It is similar to the one underlying the AGENTS programming system for AKL developed at SICS by the author and his colleagues.

The research reported herein has been supported by the Swedish National Board for Industrial and Technical Development (NUTEK), and by the sponsors of SICS: Asea Brown Boveri AB, Ericsson Group, IBM Svenska AB, Nobel-Tech Systems AB, the Swedish Defence Material Administration (FMV), and Swedish Telecom.

Sverker Janson, Computing Science Department, Uppsala University,
Box 311, S-751 05 Uppsala, Sweden

ISSN 0283-359X
ISBN 91-506-1046-5

Printed in Sweden by Graphic Systems, Stockholm, 1994.

To Kia, Adam, and Axel

CONTENTS

PREFACE

This is not intended as a standardising document for AKL, which is a young language, in need of freedom. However, the basic investigations of language principles and corresponding implementation techniques have both reached a point of quiescence where they can benefit from an exposition of this kind.

Readers acquainted with early publications on AKL will notice a few modifications: various improvements of syntax and terminology which are regarded as a step forward, leaving old dross behind.

CONTRIBUTIONS

Research is an incremental activity; stone is laid upon stone. Although this makes it difficult to single out the unique aspects of any one particular work, an attempt is made to specify those ideas and achievements that are believed to be unique to the research reported in this dissertation.

The original contributions reported in this dissertation are:

- basic principles for don't know nondeterministic concurrent constraint logic programming languages that combine, coherently and uniformly, the *searching* and *constraint solving* ability of the constraint logic programming languages (e.g., Prolog) with the *process describing* ability of the concurrent logic programming languages (e.g., GHC)
- improved control and synchronisation for such languages, in particular the notion of *stability*
- new combinators for such languages, e.g., *logical conditional*, that are amenable to stronger logical interpretations
- a simple formal computation model for such languages
- an implementation methodology based on a stepwise refinement of the formal computation model via an execution model to an abstract machine
- basic implementation techniques for languages featuring don't know nondeterminism and hierarchical constraint stores, which can serve as a foundation for further refinements and optimisations

- the notion of *ports* for efficient process communication and object-oriented programming in concurrent constraint languages

SOURCE MATERIAL

This dissertation, although a monograph, is to a large extent based on a number of previous publications and reports:

- Seif Haridi and Sverker Janson. Kernel Andorra Prolog and its Computation Model. In Warren and Szeredi, eds., *Logic Programming: Proceedings of the Seventh International Conference*, MIT Press, 1990.
- Sverker Janson and Seif Haridi. Programming Paradigms of the Andorra Kernel Language. In Saraswat and Ueda, eds., *Logic Programming: Proceedings of the 1991 International Symposium*, MIT Press, 1991.
- Seif Haridi, Sverker Janson, and Catuscia Palamidessi. Structural Operational Semantics for AKL. *Journal of Future Generation Computer Systems* **8**(1992).
- Torkel Franzén, Seif Haridi, and Sverker Janson. An Overview of AKL. In *ELP'91 Extensions of Logic Programming*, LNAI 596, Springer-Verlag, 1992.
- Sverker Janson and Johan Montelius. The Design of a Prototype Implementation of the Andorra Kernel Language, Deliverable Report, ESPRIT project 2471 (PEPMA), December 1992.
- Sverker Janson, Johan Montelius, and Seif Haridi. Ports for Objects. In Agha, Wegner, and Yonezawa, eds., *Research Directions in Concurrent Object-Oriented Programming*, MIT Press, 1993.
- Sverker Janson and Seif Haridi. An Introduction to AKL—A Multiparadigm Programming Language. In *Constraint Programming*, NATO-ASI Series, Springer-Verlag, forthcoming.

I thank my co-authors for generously allowing me to use also parts of the material where no obvious borderlines between the contributions of different authors exist.

ACKNOWLEDGEMENTS

First and foremost, I express my profound gratitude to my friend and advisor Seif Haridi for his unparalleled enthusiasm and continuous support. He has contributed to all parts of this research in too many ways to describe them all.

I would like to thank my co-worker Johan Montelius for his energy, cheerfulness, and “good vibes”. We have had fruitful co-operation on implementation as well as on programming paradigms.

The formal sections of this dissertation draw heavily upon the terminology introduced by Torkel Franzén in his quest to analyse the formal aspects of AKL. Some of his results are also summarised in this dissertation. I thank him for his