PREFACE

Historical background

The main part of this thesis is a collection of papers which are the result of the research performed by the author as a member of the computer science department of the Nijmegen University. The research has been partly sponsored by the Dutch Parallel Reduction Machine Project.

In 1984 this project was set up by the government of The Netherlands. The project was led by Prof. Henk Barendregt and involved teams at the universities of Amsterdam, Utrecht, and Nijmegen. The group at Nijmegen was led by Rinus Plasmeijer with the author of this thesis. The aim of the PRM-project was to investigate the feasibility of building a parallel reduction machine for the efficient evaluation of functional languages.

A related project in the United Kingdom, the Flagship project, was following the path of fine-grain execution, developing dataflow ideas and experience with the ALICE project: a multi-processor project of Imperial College London. The Dutch project investigated coarse-grain parallelism on fairly conventional loosely coupled multiprocessor architectures. In common with the team of the University of East Anglia (Prof. Ronan Sleep, John Glauert and Richard Kennaway) which was part of the Flagship project, the Nijmegen group had a strong interest in developing an intermediate language based on a computational model which should reflect the essential aspects of both functional languages as their implementation. It was recognized that the choice of the computational model was the most critical decision to make. It would highly influence the needed compilation effort and the final efficiency of the obtained code.

During a meeting in Oosterbeek (in the Netherlands) a subgroup consisting of Henk Barendregt, Rinus Plasmeijer and the author of this thesis decided that, in principle, Term Rewriting Systems would be best suited as the computational model to use. The patterns which are the basis of Term Rewriting Systems, were expected to contain essential information necessary for efficient implementation. Sharing of terms was felt to be essential in order to obtain efficient implementations on sequential hardware. However, the sharing of terms would be a problem in a parallel environment. We hoped that this problem could be solved in the future. To model this sharing of terms it was decided that therefore graphs had to be used instead of terms. Following meetings at a workshop on the island of Ustica, and at the second Conference on Functional Programming Languages and Computer Architecture at Nancy, September 1985, collaborative work was undertaken by the Nijmegen group and the UEA team.

After some very beneficial initial consultations with Prof. Jan-Willem Klop, a model of graph rewriting was developed within a sound theoretical framework. Spin-offs included a clear theory of

Term Graph Rewriting and the intermediate languages Lean, Dactl1 and Clean, all published in 1987. All these intermediate languages have the same underlying model of graph reduction.

The Dutch Parallel Reduction Machine Project ended in 1987. Its main results are summarized in Barendregt *et al*. (1987c). An international evaluation committee with representatives of universities and industry stressed the high quality of the research meeting international standards and having important practical applications.

The authors work which is reflected in this thesis, focusses on the level of the intermediate language and its underlying model of computation.

Contents of this thesis

Chapter 1 gives an introductory overview of the fields of functional programming and models of computation. It can be skipped by readers who are familiar with these topics.

In chapter 2 the interconnections between the topics of the following chapters are explained and it is motivated why Graph Rewriting Systems are a promising model of computation.

This thesis is essentially a collection of papers: the chapters 3, 4, 5, 6 and 7 are reprints of co-authored papers which are also published elsewhere. The papers are unchanged: only the layout is changed in order to make it more or less uniform and the references are merged in chapter 8. There was a great temptation to revise the papers (e.g. add the improved efficiency figures of the Clean compiler; remove inaccuracies; merge introductions etcetera). This is not done however in order to avoid doing the work twice in different contexts: an integrated overview of the field of functional programming and parallel graph rewriting will already appear as a textbook for students (Plasmeijer & van Eekelen (198-)).

In chapter 3 (van Eekelen & Plasmeijer (1986)) we introduce a new high level specification method for rewriting strategies, which simplifies the reasoning about the correctness of a specification of a reduction strategy. This method is being used for specifying strategies and with a minor change for specifying the semantics of Concurrent Clean. The paper was presented by the author of this thesis at the Workshop on Graph Reduction at Santa Fe, New Mexico.

Chapter 4 is also published in the Proceedings of the Conference on Parallel Architectures and Languages Europe (PARLE) at Eindhoven (Barendregt *et al.* (1987a)). It gives some basic soundness and completeness results of a graph rewriting class which is used to implement term rewriting. This makes it possible to identify restrictions guaranteeing correctness of implementations of Term Rewriting which use sharing.

Chapter 5 is a revised version of Barendregt *et al.* (1987b) which was presented by the author of this thesis at the PARLE conference at Eindhoven. It defines generalized graph rewriting which can also be applied in general declarative and even in non-declarative environments. This chapter will also appear in the special issue of the Journal of Parallel Computing with revised versions of selected papers of the PARLE conference (Barendregt *et al.* (1988)).

Chapter 6 has also been published in the Proceedings of the Third International Conference on Functional Programming Languages and Computer Architecture, Portland, Oregon, USA (Brus *et al.* (1987)). It describes a functional intermediate language based on graph rewriting with which it is shown in practise that functional languages can be efficiently implemented.

Chapter 7 broadens the scope of parallelism in functional graph rewriting to general loosely coupled parallel evaluation. This chapter is also published as internal report of the University of Nijmegen (van Eekelen *et al.* (1988)) and it is also submitted for publication elsewhere. It will form the basis of actual implementations of functional languages on parallel computer architectures like e.g. a Transputer Rack or the DOOM machine of Philips Research Laboratories at Eindhoven.