Logic and Reasoning
A Case Study in Deterministic Prolog

J. D. Newmarch

School of Information Science and Engineering
Canberra College of Advanced Education
PO Box 1
Belconnen, ACT 2614
Australia

Abstract

Deterministic Prolog programs can now be run with significantly less memory requirements and greater speed than used to be the case. Nevertheless there are problems for which the natural translation into Prolog does not give a deterministic program, and even when a specialised version is created which should be deterministic the Prolog compilers are unable to take advantage of this. This paper considers such a class of problem and shows how programs in this class can be automatically transformed and compiled into very space and time efficient programs.

Keywords and phrases: Prolog, determinism, WAM, compilation

1 Introduction

Prolog programs execute under a top-down backtracking search mechanism (Clocksin and Mellish, 1981). Backtracking allows programs to ‘search’ for a solution with little design by the user. However, even in small programs such search can have an adverse effect on speed and space, and Prolog programmers will often specifically write their programs to minimise the use of search in execution. The ability to reduce search is generally the combination of three factors: the underlying implementation must be able to take advantage of determinism, to minimise execution time and memory usage; the compiler down to the Prolog machine level must be able to generate code to exploit these features, and the programmer must be able to write programs which fit into the constraints of the compiler and the underlying Prolog machine. Such methods are briefly surveyed in the next three sections.

There are a large number of computing problems for which deterministic algorithms are known, and which can also be expressed naturally in Prolog, but for which the Prolog machine gives non-deterministic execution. Clearly it would be advantageous if these known deterministic methods could be used in the logic program. This paper considers such a problem, namely recognition of strings under an LL(1) grammar. It is known that the normal rendition of this into a Prolog program by means of the definite