Abstract Complexity of Prolog 
Based on WAM

Alexander Ja. Dikovsky
Russian Academy Inst. for Applied Mathematics,
4, Miuskaja sq., Moscow, Russia 125047
Email: dikovsky@applmat.msk.su
Fax: (7-095-)972-0737

Abstract. Simple definitions of time and space complexity measures of Prolog programs based on an abstraction of Warren Abstract Machine instructions set is suggested. Its effect is discussed on practical measuring time and space consumption, complexity classifications and optimization of programs.

1 INTRODUCTION

Proper choice of adequate time and space complexity criteria for computations in Prolog needs thorough analysis. If we try to conceive criteria based on the standard resolution semantics we would only come to somewhat like logical inferences used in practice for measuring time and to some total weight of terms used in a program for measuring space.

The notion of "logical inference" (LI) is somewhat fuzzy. Sometimes LI corresponds to one unfold (or resolution) step, and sometimes - to one unification of two terms. These are in fact of the same order. However, they differ for asymptotic time bounds. Moreover, for measuring time of highly nondeterministic programs it is convenient to distinguish between unfold steps and backtrack steps (BT), thus measuring the total time in LIs and BTs. In any case timeBT(computation) < timeLI(computation). However, for deterministic programs timeBT = 0, whereas for backtrack-choice programs it is close to timeLI.

Much more complex is the problem of choice of adequate space complexity measures for Prolog. Term-weight measures (cf.[4]) reflect that aspect of space consumption which pertains to generating terms. But it does not cope with space needed for control. For example, consider the following logically natural program which counts the number of non-list items in complex lists.

Example 1 % nlin(+Complex_list,-Items_number).
nlin([ ] ,0).
nlin([E|T],N) :-
    nlin(E,N1),
    nlin(T,N2),
    N is N1+N2.
nlin(_,1).
All terms unified in its computations are garbage. However, it wastes a lot of space, and can overwhelm Prolog workspace for complex enough lists. Standard semantics of Prolog founded on SLD-resolution rule [8] gives no terms to account for workspace size. This size depends on implementation. After long evolution de facto standard of compiling Prolog has been formed embodied in Warren Abstract Machine (WAM) instructions set [11, 1]. Almost all professional interpreters and compilers of Prolog use standard stacks: local stack(s) for activation frames and choice points, trail for backtrackable variables and global stack (or heap) for lists and structures [11, 1, 9]. So, we might think of abstract Prolog interpreter as of a sort of multi-stack automata, and define Prolog workspace as the size of its stacks. This, however would be too straightforward because the bulk of WAM instructions implement various recursion optimization rules: tail recursion optimization [3, 10], last call or activation frame optimization [11, 1], arguments indexing [11, 1], garbage collection [2], and so on. In presence of these rules abstract interpreter becomes too overburdened by technical details and this is the main difficulty in space complexity abstraction. Nevertheless space complexity measures definitely must reflect the effect of WAM optimizations so that to be practically feasible. WAM optimizations indirectly regulate the style of practical programming. For example, more practical version of the nlin/2 predicate above should be somewhat like this:

Example 2  \%

\begin{verbatim}
nlin(L,N) :-
nlin_t(L, [ ], 0, N).
nlin_t([ ], [ ], N, N) :- !.
nlin_t([ ], [E|T], N, R) :- !,
nlin_t(E, T, N, R).
nlin_t([E|T], S, N, R) :- !,
nlin_t(E, T[S], N, R).
nlin_t(_, L, N, R) :- N1 is N+1,
nlin_t([ ], L, N1, R).
\end{verbatim}

And the measures should indicate the reason for which this program needs constant workspace.

2 ASM-COMPLEXITY

Of course it would be contradictio in adjecto if so that to write or estimate programs in logic programming language we should know details of its implementation. Instead we should use very simple abstract model of standard interpreter sufficient for adequate measuring workspace. We have offered such a model called abstract stack machine (ASM) in [5, 6] and sketch it here. It is founded on the