On Eliminating Loops in Prolog

by

David Poole
Randy Goebel

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-85-27
August 1985

On eliminating loops in Prolog

David Poole
Randy Goebel

Logic Programming and Artificial Intelligence Group
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

Recent papers have explained how infinite loops in a Prolog search tree can be avoided by the use of subgoal deletion. We show here that this works only in limited cases, and argue that these cases can be better avoided by slight modifications of the program, rather than by increasing the complexity of all programs with a rule that has very limited applicability.

Introduction

It is relatively easy to generate infinite loops in Prolog, e.g., the assertions

```
p(X) <- p(Y)
p(a);
```

and the goal

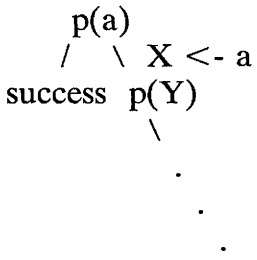
```
?p(a);
```

generate the infinite branch

```
p(a)
 | X <- a
p(Y)
 | Y' <- Y
p(Y')
 .
 .
 .
```

Although the Prolog search space, defined by SLD resolution (e.g., see [Lloyd82]), is complete, Prolog's depth first left-to-right search strategy is incomplete. In other words, the search can stumble into an infinite branch of a proof tree even when an existing proof lies on another branch. For example in the above program, a proof of the goal p(a) can be had by re-ordering the rules, thus placing the terminating

proof tree branch before the infinite branch, viz.



Covington [Covington85] reports that when a new subgoal of the proof tree "matches" another subgoal higher in the proof tree, that branch can be "blocked," thus avoiding the undesirable infinite loop.

Loveland and Reddy [Loveland81] prove that this strategy works in the propositional case, and further claim "The results presented here in the propositional format with equality, hold equally well in the first-order format ... since it is a propositional rule." [p. 647]

Brough and Walker [Brough84] describe this as a goal terminating interpreter (for restricted "simple knowledge bases") and allow failure on the basis of subgoals that are "syntactically identical." Their notion of syntactically identical is not explicity defined.

Counter examples

There are several cases where some kind of match might sanction subgoal deletion, but the deletion is incorrect.

- Subgoal is more general than higher goal:

```
?p(a);
p(b);
p(a) <- p(X);
```

- Subgoal is less general than higher goal:

```
?p(X) q(X);
p(a);
q(b);
p(X) <- p(a);
```

- Subgoal and higher goal refer to independent variables:

```
?p(X) q(X);
p(a);
q(b);
p(X) <- p(Y);
```

In all of the above cases, the rules can be reordered to generate infinite branches. Any of the potential subgoal deletion rules applied to the above programs would result in a correct derivation not being found.

- Subgoal and higher goal refer to variables bound together but without a constant binding. That is, any instance of the the subgoal will force the same instance of the higher goal. Here there are no counter examples, because of the following theorem:

  Theorem: *Suppose G1 is a subgoal occuring below G2 along a branch of a proof tree. If G1 and G2 are already identical by binding, then G1 may be failed.*

  Proof: Suppose there is a proof containing G1. We need to show that there is a proof without G1. The subproof of G1 may be used as a proof for G2, with the resulting proof having one less instance of goal G1. By induction, all instances of G1 can be removed.

Conclusions

The cases where subgoal deletion works are very limited. In particular, subgoal deletion as sanctioned by unification works in very few cases. It does work for biconditionals, but in this case one would probably prefer to choose a canonical element of the equivalence class, and convert all other equivalent cases to that element. It also works for symmetric relations, but in that case one can introduce a symmetric relation that is defined by two rules whose antecedents are the corresponding asymmetric relations, e.g.,

```
symP( X Y ) <- P( X Y );
symP( X Y ) <- P( Y X );
```

References

[Brough84] D.R. Brough and A. Walker (1984), Some practical properties of logic programmin interpreters, *Proceedings of the International Conference on Fifth Generation Computer Systems*, November 6-9, Institute for New Generation Computing, Tokyo, Japan, 149-156.

[Covington85] M.A. Covington (1985), Eliminating unwanted loops in Prolog, *ACM SIGPLAN Notices*, vol. 20, no. 1, 20-26.

[Lloyd82] J.W. Lloyd (1982), Foundations of logic programming, Technical Report 82/7, Department of Computer Science, University of Melbourne, Melbourne, Australia, August.

[Loveland81] D.W. Loveland and C.R. Reddy (1981), Deleting repeated goals in the problem reduction format, *ACM Journal* **28**(4), 646-661.