PROLOG GENERAL INFORMATION

Prolog Command Line

The Prolog interpreter is held in the file cmds/prolog on the Prolog diskette. The Prolog command line is:

```
OS9: prolog [-m<mgthresh>]
```

The optional "-m<mgthresh> parameter may be used to specify the threshold for garbage collection. When Prolog passes this threshold, it will begin garbage collection, retaining only terms and variables that are referred to in parts of the program not yet executed. If the first garbage collection does not bring the total amount of memory used by Prolog below the threshold, Prolog will continue to include a garbage collection in each interpreter cycle until the amount of memory is brought below the threshold. If the threshold is set too low, garbage collection will be started more frequently than necessary, which may substantially slow down program execution. On the other hand, the threshold should not be set so high that Prolog actually cannot acquire enough memory to reach it. In this case, execution can be aborted through lack of memory even though much of the memory used could be released on garbage collection. The threshold may be specified as a number (of pages) or, if followed by the letter 'k', as a n umber of kilobytes. For example:

```
OS9:prolog -m5k
OS9:prolog -m20
```

Both these lines set the garbage collection threshold to 5 kilobytes. Command line I/O redirection may be used to run a Prolog program stored in a disk file and to direct the output to a disk file or character device. For example:

```
OS9:prolog -m5k <report >/p
```

which runs the Prolog program in the file "report" in the current data directory, and sends the output to the parallel printer (/p).

Prolog Dialogues

If the standard input is redirected (either at the command line or during the execution of a Prolog program), the input prompt is suppressed. Prolog's normal prompt is "P>" which it issues when expecting a rule or query. If the rule or query is not syntactically complete when ENTER is pressed, Prolog waits for another line of input (and will wait for as many lines as necessary to complete the syntax, or to trigger an error condition), but does not issue another prompt after the first one. If a rule is entered, it is added to the rule base as the last clause for the predicate it names, and Prolog returns with the "P>" prompt. If a goal is entered, Prolog attempts to find all solutions by backtracking. The results for all variables appearing in the query are printed within angle brackets (<>). If the query contains no variables, a set of empty angle brackets is still printed for each solution found. After all solutions have been printed, the next "P>" prompt appears. If no angle brackets appear between the query and the next prompt, then no solutions have been found.

Structure Sharing

Metasoft Prolog uses a method known as "structure sharing" to save memory. This means that whenever a variable is bound to a term, the term is not copied into the environment, but a pointer is set linking the variable to the term.

Optimisation

Metasoft Prolog includes two important optimisations. The first is optimisation of deterministic calls. A deterministic call is one which creates no alternatives. When such a call terminates, it is possible to recover all the stack space normally associated with choice points. Some predicates are implicitly deterministic, ie the clause that was selected, although not the last available, is the last that can b matched. If the programmer knows this to be the case, he can us the cut (/) to tell the interpreter that the call is deterministic so that on termination the stack space can b recovered. The second form of optimisation is called tail-recursion optimisation. This means that the Prolog interpreter can recognise a recursive call which can be replaced, in effect, by a goto. Consider the 'C' function tailr():

```
tailr() {
      do_something();
      tailr();
}
```

Compare this to the Prolog program:

```
tailr():-
      do-something(),
      tailr();
```

The 'C' program will run out of stack space, but could be rewritten as the equivalent 'C' program:

```
tailr() {
A:
      do_something();
      goto A;
}
```

which will run indefinitely. The process of replacing the recursion at the end of the function with a goto is known as tail-recursion optimisation. In the case of the Prolog program, provided that do-something() was deterministic, the tail-recursion optimisation is performed automatically by the interpreter. Consider the dialogue:

```
P>go(?n):-W(?n),W("@13"),ADD(?n,1,?m),go(?m);
P>go(1);
1
2
...
```

Execution will continue indefinitely (provided that the garbage collection threshold has been set sensibly) without any kind of memory limitation. It is important to note that tail-recursion optimisation only applies where the call is deterministic. As in the optimisation of deterministic predicates, this can be enforced by the programmer by use of the cut (/) predicate.

The Dictionary

Prolog maintains a dictionary of all the constants in use at a given time. When a new constant comes along, it is added to the dictionary. When a constant comes along that already appears in the dictionary, instead of creating a new copy of it, a pointer is set to refer to the existing dictionary entry. Thus the use of long constants such as descriptive predicate names is encouraged as the memory overhead is very small. The same dictionary is used for variable names, so descriptive variable names can also be used freely.

The Relational Database

The relational database is held as a binary tree indexed by predicate name. To speed up processing, when the last clause for a predicate is deleted, the index entry is not deleted. If new clauses are later added for a predicate that has been deleted, the old index entry is reused. The moral is, don't create too many new predicate names during the execution of a program.

Undefined Predicates

If a goal calls for the matching of a predicate which does not appear in the relational database, the goal automatically fails. This is in contrast to some implementations of Prolog which raise an error on the same condition.