Computer Science 3MI3 – Prolog tidbits

Mark Armstrong

September 24th, 2020

Contents

Unguarded cases

Not "guarding" the cases in your predicate definitions can result in slightly different behaviour when it comes to backtracking.

```
% This tidbit illustrates the effect of
% not correctly restricting the application of different cases
% in our predicate definitions.


% This predicate is well defined; two cases,
% each of which is restricted to a particular form of list.
fact1([]) :- writeln('fact1, base case'), true, !.
fact1([_|T]) :- writeln('fact2, recursive case'),fact1(T).



% This similar predicate includes an extra case.
fact2([]) :- writeln('fact2, base case'), true, !.
fact2([_|T]) :- writeln('fact2, recursive case'), fact2(T).
% The last case seems unreachable.
% But this will make the search continue and return `false`
% as a second result. (Assuming the querier presses ; or
↪  <space>.)
fact2(_) :- writeln('fact2, supposedly unreachable case'),
↪  false.
```

You can try out this example by loading the source code into your REPL.

```
% Query test to see how Prolog will pause after the query,
% allowing you to to use ; or <space> to then get `false.`
test :- fact2([1]).


% Querying expected will show what we would expect to happen
↪  instead;
% the search returns `true.` and halts, not allowing `;` or
↪  `<space>`.
expected :- fact1([1]).
```