4. Even More Recursion

Miscellaneous Information

- **Trace Command**
  In Prolog the command `trace.` switches the interpreter to “trace mode”. Every time a query is asked, the interpreter will show every step it takes to solve it. At every step the user can type <Enter> to continue or `a` to abort (see other options with `h`). To switch off “trace mode”, enter the command `nodebug`. This way it is possible to check and understand what happens when the program does not work as expected.

More Recursion

As we have seen, a recursive program usually looks something like this:

```prolog
ancestor(A, B) :- parent(A, B).

ancestor(A, B) :- parent(A, X), ancestor(X, B).
```

There are two (or possibly more) clauses, at least one of which is a base case and the other a (set of) recursive predicate(s) calling itself (themselves). In this, the order of the clauses is very important. Not only the order of base case (e.g. first clause of the example) and the recursive rule, but also the order of statements within the recursive clause, compare e.g.:

```prolog
ancestor(A, B) :- parent(A,X), ancestor(X,B).
```

vs.

```prolog
ancestor(A, B) :- ancestor(X,B), parent(A,X).
```

Exercise 1

- Change the order of clauses, one by one until you have tried all possible permutations of the recursive ancestor clauses. Use the `trace` command to analyse how this changes the behaviour and possibly results of the search.
- Write a short but comprehensive description of what you think is going on, in particular how order might affect the search. Would it be acceptable to write `ancestor(A, B) :- ancestor(B,X), parent(X,A).` in terms of search results this would yield? And how does this compare to the two other versions above?

Exercise 2

- Building on last week’s exercise, define a predicate `knowEachOther(A,B)` that is defined by A and B knowing each other if they have a common ancestor OR if they are friends with the same person (`friend` can be stated as a fact).
- To make it easier for yourself you should test each sub-predicate on either side of the OR separately to better see whether each one of them is working correctly on their own. Remember, for OR one or both clauses could be correct in order to return an overall success. You might also want to think about redefining the friend fact/predicate, so it’s symmetric.
- **Note:** don’t forget to include a base case to make sure the recursive predicate will terminate.
- **Note:** again, describe the predicates in English using comments – this is essential for receiving credit for this exercise!

Submission

Send both exercises in one `.pl` file, (each exercise clearly marked within!) bearing your full name and lab number, i.e. `carmen-klaussner-lab5.pl`. Submit this by Monday, 22nd 2016: 6 pm to: klaussnc@tcd.ie.