1. tutorial in Prolog

November 21, 2021

Familiarize yourself with the royal family of the British Monarchy. Consider the following people:

- `william`: Prince William of Wales
- `harry`: Prince Henry of Wales
- `charles`: Prince Charles, Prince of Wales
- `diana`: Diana, Princess of Wales
- `camilla`: Camilla, Duchess of Cornwall
- `george`: George VI of the United Kingdom
- `elizabeth`: Elizabeth II, HM The Queen
- `philip`: Prince Philip, Duke of Edinburgh
- `edward`: Prince Edward, Earl of Wessex
- `sophie`: Sophie, Countess of Wessex
- `louise`: Princess Louise of Wessex
- `james`: Prince James of Wessex

and their relationships:

- `male(X)` means that `X` is a man.
- `female(X)` if `X` is a woman.
- `parent(P,C)` if `P` is the parent of `C`. E.g. `P` can be Lady Diana and `C` Prince William. Not the other way round!
- `wife(W,H)` if `W` is (or was) the wife of `H`.

Task 1: Copy&Paste the code on the next page into a file "`royal.pl`". It is loaded into Prolog by the "`[filename]`" command. The console should look like:

```
?- [royal].
% royal compiled 0.00 sec, 30 clauses
true.
```

Task 2: Write a query to ask for all children of `elizabeth`.

Task 3: Define the predicate `husband(Man,Woman)`. Do not list all husbands of all wives as ground facts! :-)

Task 4: Define `person(P)` to be either a male or a female. Try avoiding the `;` symbol (which defines a logical *or*).

Task 5: Define `mother(Mother,Child)` and `father(Father,Child)`. Be careful not to define a son or a daughter.

Check your knowledge:

- What is a difference between a "`Person`" and a "`person`"?
- What is an underscore "`_`"? A *singleton*? Should you avoid it?

```
female(camilla).
female(diana).
female(elizabeth).
female(louise).
female(sophie).

male(charles).
male(edward).
male(george).
male(harry).
male(james).
male(philip).
male(william).

parent(charles,harry).
parent(charles,william).
parent(diana,harry).
parent(diana,william).
parent(edward,james).
parent(edward,louise).
parent(elizabeth,charles).
parent(elizabeth,edward).
parent(george,elizabeth).
parent(philip,charles).
parent(philip,edward).
parent(sophie,james).
parent(sophie,louise).

wife(camilla,charles).
wife(diana,charles).
wife(elizabeth,philip).
wife(sophie,edward).
```

Task 6: Compare the two definitions of the `father` predicate:

a) `father1(F,C) :- male(F), parent(F,C).`

b) `father2(F,C) :- parent(F,C), male(F).`

Which of them is faster on the `father`*`[1/2]`*`(X,charles)` query? Why?

Note. You can estimate Prolog's speed by starting the *trace* mode:

```
?- trace.
true.
[trace] ?-
```

It can be turned off by the `nodebug` command.

Task 7: Write two SLD trees for `father1(X, charles)` and `father2(X, charles)`.

Check your knowledge:

- What is a *left-to-right* rule?
- What is a *top-to-bottom* rule?

Task 8 (optional): Define the `ancestor(Ancestor,Descendant)`, which connects parents with children, grandparents with grandchildren, grand-grandchildren with grand-grand-children, etc.

Task 9 (optional): Using the course literature or Google, study the "negation as failure `\+`" technique or "non-unifiability predicate `\=`".

Define the `sibling(Sibling1,Sibling2,Parent)` predicate: `Sibling1` is the sibling of `Sibling2` and `Parent` is their shared parent.

Be careful, no person is its own sibling. Therefore `sibling(william,william,P)` must fail!