Lab session 8: First-Order Logic

Pierre-Alexandre Murena

June 7, 2024

1 English and First Order Logic

1.1 From logical expression to English

Question 1. Translate into English the following sentence:

$$\begin{aligned}\forall\ x, y, l\ \mathit{SpeaksLanguage}(x, l) \land \mathit{SpeaksLanguage}(y, l)\\ \rightarrow \mathit{Understands}(x, y) \land \mathit{Understands}(y, x)\end{aligned}$$

1.2 From English sentences to logical expressions

Question 2. For each of the following sentences in English, state which of the formulae correctly expresses the English sentence:

1. Berlin and Hamburg are both in Germany:
   (a) $\mathit{In}(\mathit{Hamburg} \land \mathit{Berlin}, \mathit{Germany})$
   (b) $\mathit{In}(\mathit{Hamburg}, \mathit{Germany}) \land \mathit{In}(\mathit{Berlin}, \mathit{Germany})$
   (c) $\mathit{In}(\mathit{Hamburg}, \mathit{Germany}) \lor \mathit{In}(\mathit{Berlin}, \mathit{Germany})$
2. There is a country that borders both France and Germany:
   (a) $\exists c\ \mathit{Country}(c) \land \mathit{Border}(c, \mathit{France}) \land \mathit{Border}(c, \mathit{Germany})$
   (b) $\exists c\ \mathit{Country}(c) \rightarrow [\mathit{Border}(c, \mathit{France}) \land \mathit{Border}(c, \mathit{Germany})]$
   (c) $[\exists c\ \mathit{Country}(c)] \rightarrow [\mathit{Border}(c, \mathit{France}) \land \mathit{Border}(c, \mathit{Germany})]$
3. All countries that border Germany are in Europe
   (a) $\forall c\ \mathit{Country}(c) \land \mathit{Border}(c, \mathit{Germany}) \rightarrow \mathit{In}(c, \mathit{Europe})$
   (b) $\forall c\ \mathit{Country}(c) \rightarrow [\mathit{Border}(c, \mathit{Germany}) \rightarrow \mathit{In}(c, \mathit{Europe})]$
   (c) $\forall c\ [\mathit{Country}(c) \rightarrow \mathit{Border}(c, \mathit{Germany})] \rightarrow \mathit{In}(c, \mathit{Europe})$

Question 3. Express in first-order logic the fact that friendship is transitive.

2 Introduction to Prolog

Prolog is a **logic programing** language that is based on first-order logic. Its characteristic is that it is only declarative: the programs are made of assertions and rules only.

2.1 Getting Prolog

To use Prolog, you have several alternatives. Here, we present two convenient ones. Feel free to pick the one you prefer, or to opt for another one in case you are already used to something else.

2.1.1 SWISH

For this session, you can use the online interface of SWISH, which allows you to create Prolog programs and execute queries: `https://swish.swi-prolog.org`.

When you first arrive on the website, you will see a new tab on the left hand side (LHS) window. This new tab offers you the possibility to create a program. Click on this button. This will open an empty program on the LHS, in which you can write your Prolog program.

The right hand side (RHS) window allows you the possibility to make queries. A query corresponds to asking Prolog whether a sentence is true or false. The answer to a query will print in the upper part of the window. We will get into more details in the next sections.

2.1.2 SWI-Prolog

However, if you want to experiment more with Prolog, you can also install SWI-Prolog.

The remainder of this section are instructions specific to SWI-Prolog. You can skip them if you are using SWISH.

On the SWI-Prolog console, you can use the `pwd.` command (don't forget the dot!) to check the current directory. To change directory, you can use the command `cd('...').` where the ... needs to be replaced by the path.

To load a program (for instance here `family.pl`), use the command `[family].` Note that SWI-Prolog can be used **only** to execute already written prograns, not to edit them. If you need to make changes to your program, you must open the pl file in a text editor.

Once your program is loaded, you can execute queries on the console. A query corresponds to asking Prolog whether a sentence is true or false. We will get into more details in the next sections.

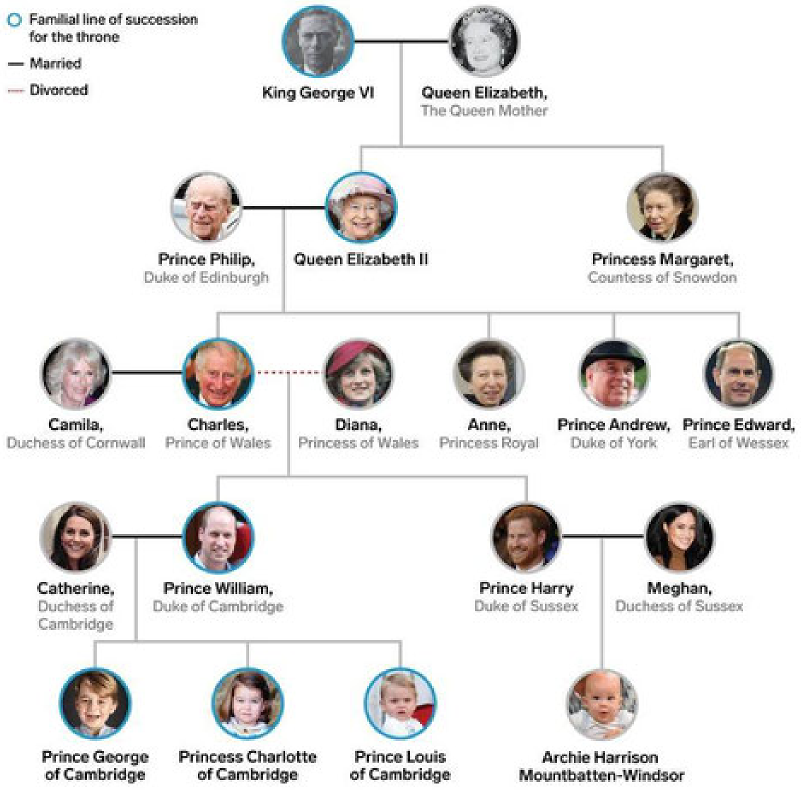
2.2 Assertions

The `family.pl` file contains a list of **assertions**, i.e. facts, about the British royal family (see picture). These facts are statements of two kinds:

- `mother(X,Y)` indicates that X is the mother of Y.
- `father(X,Y)` indicates that X is the father of Y.

For instance, the first assertion in the program, `father(philip, charles).`, means that Philip is Charles' father.

Note that `mother` and `father` are relations and that `philip`, `charles`, `diana` etc are constants. In Prolog, constants are identified by their names starting with a lower case.


Familial line of succession for the throne
Married
Divorced
King George VI
Queen Elizabeth, The Queen Mother
Prince Philip, Duke of Edinburgh
Queen Elizabeth II
Princess Margaret, Countess of Snowdon
Camila, Duchess of Cornwall
Charles, Prince of Wales
Diana, Princess of Wales
Anne, Princess Royal
Prince Andrew, Duke of York
Prince Edward, Earl of Wessex
Catherine, Duchess of Cambridge
Prince William, Duke of Cambridge
Prince Harry Duke of Sussex
Meghan, Duchess of Sussex
Prince George of Cambridge
Princess Charlotte of Cambridge
Prince Louis of Cambridge
Archie Harrison Mountbatten-Windsor

Question 4. If you use SWI-Prolog, load `family.py`, and if you use SWISH copy-paste the content of the file into the program window. Execute the query `father(philip, charles).`[1] What is the output? Is it consistent with the family tree? Execute a query to check whether Louis is the son of Diana.

Note that Prolog only checks whether a fact is included in its knowledge base. For instance, if you ask whether Elizabeth is the daughter of George VI, it will reply false: this is because it does not find any evidence of this fact in its knowledge base. It does not have the option to reply "I don't know".

More than checking whether a fact is in the knowledge base, one might want to get the answer to more sophisticated question, for instance: Does Diana have children?

Question 5. Write the sentence "Diana has children" in first-order logic.

Question 6. In order to test the existence of children, we need to introduce a variable. Variables in Prolog always start with an upper case. The query `father(william, X).` then

[1] Never forget the dot!

tests the existence of some X such that William is the father of X. Execute this query. You will see that Prolog answers by proposing an instantiation of X. You can get all the instantiations one by one (on SWI-Prolog by pressing ; and on SWISH by clicking Next) or stop the execution (on SWI-Prolog by pressing . or Enter, and on SWISH by clicking Stop). Using this, query who are Diana's children.

In Prolog, the $\wedge$ is represented by a comma. For instance `father(charles, william), mother(elizabeth, harry).` will check whether Charles is William's father **and** Elizabeth is Harry's mother. The $\vee$ is represented by a semicolon. For instance `father(charles, william); mother(elizabeth, harry).` will check whether Charles is William's father **or** Elizabeth is Harry's mother.

Question 7. Write a query[2] checking who are Louis' parents. Check what happens if you change the operator in your query ($\wedge$ instead of $\vee$ or $\vee$ instead of $\wedge$).

2.3 Rules

Until now, we have only explored assertions that are already present in the knowledge base. However, we may want to reason about them, and especially think of other interesting relations. For instance, we may want to define the relation `sibling` or `ancestor`.

Rules in Prolog are formulated as **Horn clauses**, i.e. sentences of the form

$$\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \psi$$

For instance, in the context of the family tree, one might express the fact that a father is always a parent:

$$father(X,Y) \rightarrow parent(X,Y)$$

with all variables in the clause being implicitly universally quantified, i.e.

$$\forall X\ \forall Y\ \ father(X,Y) \rightarrow parent(X,Y)$$

. The syntax of Prolog writes the implication in reverse form, from the consequence to the cause. In other words, this example can be written in Prolog:

```
parent(X,Y) :- father(X,Y).
```

Question 8. Add this rule to the program (note that you cannot add it directly on the console, which only allows you to execute queries). Check that Harry is the parent of Archie. Check that for now Meghan is not recognized as the parent of Archie.

To fix this, we have two options: either changing the previous rule to state that a parent can be either a father or a mother, or by adding another rule stating that a mother is a parent.

Question 9. Implement this new rule and check that now Meghan is identified as Archie's mother.

[2] A single query, not two!

Question 10. Implement the rule `ancestor(X,Y)` (which reads "X is the ancestor of Y") stating that an ancestor is either a parent or an ancestor of a parent.

We can implement the notion of sibling using the rule: "siblings have the same parents".

Question 11. Implement this rule and test it to find the siblings of Harry. What do you observe? Fix the rule to make it work. You may use the command `A \= B` which checks that $A \neq B$. With this new version, you may observe that the results are repeated twice. If so, can you explain?

2.4 Movie domain

In `movie.pl`, you will find data about several movies, their directors and actors.

Question 12. Using queries, answer the following questions:

1. Which movies were released in 2000?
2. Which directors also acted in some movies?
3. Which actors appeared in more than one movie?

Question 13. Add the following rules to the program:

1. `same_year(M1, M2)`: checks that two movies were released the same year.
2. `knows(A, D)`: states that an actor knows the directors of movies it appeared in.