CSE 307 - Relational Programming

TP1 Relational Databases in Datalog

François Fages (Francois.Fages@inria.fr)

Bachelor of Science - Ecole polytechnique

In all practical sessions of this course, you will use a dialect of Prolog called SWI Prolog that you are supposed to have installed on your laptop. We recommend you to use

- your favourite editor to edit Prolog files (.pl not to be confused with Perl mode)
- and run the Prolog interpreter in a terminal window with command `swipl`

At each TP session, you will be asked to upload on the Moodle

- a copy of the Prolog file of the session named `tpi.pl`
- completed with your answers to the questions
  - either missing code (i.e. Prolog *facts* and *rules*)
  - or textual answers in the comment blocks created for that
  - or Prolog *queries* with execution traces, similarly copied in the comment blocks

1. Using SWI Prolog

The Prolog file `tp1.pl` contains a small database of family relations defined *in extension* by Prolog facts, and *in intension* by Prolog rules.

```
man(pierre).
man(david).
man(benjamin).

parent(jean, david).
parent(jean, benjamin).

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

brother(X, Y) :- parent(Z, Y), dif(X, Y), parent(Z, X), man(X).
```

The identifiers stating with a upper case letter or the symbol _ represent a Prolog variable.

Those starting with a lower case letter represent a Prolog constant or a Prolog predicate (or a function but Datalog does not use function symbols).

The SWI-Prolog interpreter is called with the command `swipl` which will bring you in the top level interpreter.

```
prompt% swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- [tp1].
true.

?- man(pierre).
true.

?- man(catherine).
false.

?- man(xyzzy).
false.

?- parent(X,joel).
X = robert ;
X = lucie.

?- parent(X, joel), dif(X, Y), parent(Y, joel).
X = robert,
Y = lucie ;
X = lucie,
Y = robert ;
false.

?- brother(X,lucie).
X = jean ;
X = michel ;
X = jean ;
X = michel.
```

The trace facility allows you to watch the resolution steps (enter Return for the next step).
The variables introduced by the resolution steps are numbered and prefixed with _
The trace can be used here to understand why the brothers are found twice in the answers to the query: once through the father relation, Pierre, once through the mother relation, Catherine.

```
?- trace.
true.

[trace]  ?- brother(X,lucie).
   Call: (10) brother(_11284, lucie) ? creep
   Call: (11) parent(_11722, lucie) ? creep
   Exit: (11) parent(pierre, lucie) ? creep
   Call: (11) dif:dif(_11284, lucie) ? creep
   Exit: (11) dif:dif(_11864{dif = ...}, lucie) ? creep
   Call: (11) parent(pierre, _11864{dif = ...}) ? creep
   Exit: (11) parent(pierre, jean) ? creep
   Call: (11) man(jean) ? creep
   Exit: (11) man(jean) ? creep
   Exit: (10) brother(jean, lucie) ? creep
X = jean ;
   Redo: (11) parent(pierre, _11864{dif = ...}) ? creep
   Redo: (11) parent(pierre, _11864{dif = ...}) ? creep
   Exit: (11) parent(pierre, michel) ? creep
   Call: (11) man(michel) ? creep
   Exit: (11) man(michel) ? creep
   Exit: (10) brother(michel, lucie) ? creep
X = michel ;
   Redo: (11) parent(_14182, lucie) ? creep
   Exit: (11) parent(catherine, lucie) ? creep
   Call: (11) dif:dif(_11284, lucie) ? creep
   Exit: (11) dif:dif(_14324{dif = ...}, lucie) ? creep
   Call: (11) parent(catherine, _14324{dif = ...}) ? creep
   Exit: (11) parent(catherine, jean) ? creep
   Call: (11) man(jean) ? creep
   Exit: (11) man(jean) ? creep
   Exit: (10) brother(jean, lucie) ? creep
X = jean ;
   Redo: (11) parent(catherine, _14324{dif = ...}) ? creep
   Redo: (11) parent(catherine, _14324{dif = ...}) ? creep
   Exit: (11) parent(catherine, michel) ? creep
   Call: (11) man(michel) ? creep
   Exit: (11) man(michel) ? creep
   Exit: (10) brother(michel, lucie) ? creep
X = michel.

[trace]  ?- notrace.
true.

[debug]  ?- nodebug.
true.
```

2. Questions on the relational database

Inspect the file `td1.pl` note that the file contains some more facts and a `woman/1` predicate.

You are asked to write

- Prolog queries for answering questions about familial relationships in that database
- Prolog clauses for defining other familial relationship predicates

Answer the questions directly in the file `tp1.pl` either in textual comment blocks for the queries or as extra Prolog code to insert in that file and try by loading it again with command `[td1]`.

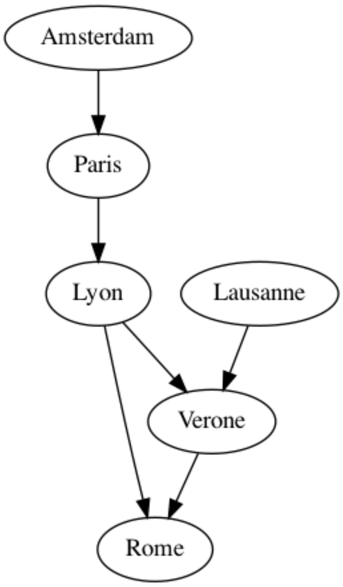
3. Questions on directed graphs

A directed graph G=(S,A) is composed of a set S of vertices and a set A⊆SxS of couples of vertices called arcs.

A directed graph is acyclic if it contains no circuit.

You are asked to

- represent the graph of the simple bus route map depicted below with a predicate `arc/2`
- define the transitive closure predicate path/2 assuming the graph is acyclic
- prove that your program does terminate on acyclic graph



Answer the questions directly in file `tp1.pl`

Finally, don’t forget to upload your file on the Moodle!

https://moodle.polytechnique.fr/course/view.php?id=14906