COMP90048 Declarative Programming
Semester 1, 2018
Peter J. Stuckey

Declarative Programming

Answers to workshop exercises set 6.

QUESTION 1

Download the files borders.pl, cities.pl, countries.pl, and rivers.pl. These files contain facts about the world circa 1980. Create a file world.pl and insert the four lines:

```
:- ensure_loaded(borders).
:- ensure_loaded(cities).
:- ensure_loaded(countries).
:- ensure_loaded(rivers).
```

These lines will automatically load the four files when you load your world.pl file.

Start up SWI Prolog and load your world.pl file.

This will define a predicate borders/2 (that notation means a predicate named borders that takes two arguments) describing which countries and oceans border which others.

Give a query to find what borders Australia (remember: Prolog symbols are all lower case).

ANSWER

```
?- borders(australia, X).
X = indian_ocean ;
X = pacific.
```

QUESTION 2

Give a query to find what shares a border with both France and Spain.

ANSWER

```
?- borders(france, X), borders(spain, X).
X = andorra ;
X = atlantic ;
X = mediterranean ;
false.
```

QUESTION 3

The files you have loaded also define a predicate country/8:

```
    country(Country,Region,Latitude,Longitude,Area,
            Population,Capital,Currency)
```

where Country is a country located in Region at the indicated Latitude and Longitude, occupying the specified Area, occupied by the specified Population, with the specified Capital city and using the specified Currency.

Give a query to find what countries share a border with both France and Spain. Remember, _ specifies a "don't care" variable.

ANSWER

```
?- borders(france, X), borders(spain, X), country(X,_,_,_,_,_,_,_).
X = andorra ;
false.
```

QUESTION 4

Edit your world.pl file and define a predicate country/1 so that country(C) holds when C is any country. Reload your file and use your new country/1 predicate to find what countries share a border with both France and Spain. Note that you can type the goal "make." to Prolog to reload any changed files, much like ":reload (or ":r") in GHCi.

ANSWER

    In world.pl:

```
country(C) :- country(C,_,_,_,_,_,_,_).
```

    In SWI Prolog:

```
?- make.
    /home/peter/subjects/686/workshops/examples/world compiled 0.00 sec,
6 clauses
true.

?- borders(france, X), borders(spain, X), country(X).
X = andorra ;
false.
```

QUESTION 5

Edit your world.pl file again to define a predicate larger/2 so that larger(Country1, Country2) holds when the area of Country1 is larger than that of Country2. You can use the (infix) predicates < and > to compare numbers, but note that you must ensure that the arguments of a comparison are bound when the comparison is executed, so the goals that bind the values to be compared must appear before the comparison.

Which is bigger, Australia or China?

ANSWER

In world.pl:

```
larger(C1, C2) :-
    area(C1, Area1),
    area(C2, Area2),
    Area1 > Area2.

area(Country, Area) :-
    country(Country,_,_,_,Area,_,_,_).
```

This could be done without the auxilliary area/2 predicate, but this way is a little nicer.

In SWI Prolog, reload, then:

```
?- larger(australia, china).
false.
```

China.

QUESTION 6

The predicate river/2 relates rivers, their countries, and the sea they drain into. river(River, Countries) holds when River is a river that flows through or into all of the countries on the list Countries.

The member/2 predicate is an SWI Prolog built-in that relates lists and their elements. member(Elt, Lst) holds when Elt is an element of Lst.

Write a predicate river_country(River, Country) that holds when River is a river, Country is a country, and River flows into and/or out of Country.

Also write a predicate country_region(Country, Region) that holds when Country is a country in region Region.

Give a query to find a river that flows between countries in different regions.

ANSWER

    In world.pl:

```
river_country(River, Country) :-
    river(River, Countries),
    member(Country, Countries),
    country(Country).

country_region(Country, Region) :-
    country(Country,Region,_,_,_,_,_,_).
```

    In SWI Prolog (after reloading):

```
?- river_country(River, Country1),
|    river_country(River, Country2),
|    country_region(Country1, Region1),
|    country_region(Country2, Region2),
|    Region1 \= Region2.
```