Declaratieve Talen
Prolog 4

1   Holiday lights

The European government has decided to festively light major European high-
ways during the holiday season. There will be differently colored lights, however
each highway will have exactly one color. During the holiday season you plan
some family visits, and you like some variety while driving. Therefore you de-
cide to drive each highway exactly once, but two consecutive highways you drive
can not be lit by the same color.

A more abstract representation of this problem is as follows. Assume M
bidirectional connections that each connect two places, and are lit by a specific
color. Assume N places, numbered from 1 through N. You start at place 1.
Now you have to plan a trip such that each connection is used exactly once, no
two consecutive connections have the same color, and your trip should end back
at place 1.

Consider the following example. There are three places. There is a yellow
highway between places 1 and 2, a blue highway between 2 and 3, and a yellow
highway between 1 and 3. You can drive a yellow highway from 1 to 2, a blue
highway from 2 to 3, and another yellow highway from 3 back to 1.

Such a graph can be represented by Prolog facts. For our example we can
write:

```
highway(1,2,yellow).
highway(2,3,blue).
highway(1,3,yellow).
```

Before you start planning your trip, you need to check the following two
conditions:

1. The nodes (i.e. places) have an even number of connections,

2. If a node K, such that $1 < K \leq N$, has X connections lit by the same
   color, then the node K should have at least X connections with other
   colors.

Write the predicate `check` which will check these two conditions for a
network of highways given by `highway/3` facts. This predicate succeeds if both
conditions are met, and fails otherwise.

Hint Decompose the check into two separate predicates, `check_even_at(N)` and `check_colors_at(N)`, verifying the first and second conditions at node `N`, respectively.

Hint Try to use negation to quickly check the conditions for every node. Remember that write the negation of a conjunction, you must leave a space between `\+` and the conjunct (e.g., `\+ (p(X),q(Y))`).

Some examples:

```
highway(1,2,yellow).
highway(2,3,yellow).
highway(1,3,blue).

?- check.
fail.

highway(1,2,yellow).
highway(2,3,blue).
highway(1,3,yellow).

?-check.
true.
```

Write the predicate `tour(T)` which will first check the conditions listed above for a network of highways given by `highway/3` facts, and then calculates a trip. Our example has two possible trips: `T = [2-yellow, 3-blue, 1-yellow]` and `T = [3-yellow, 2-blue, 1-yellow]`. In such a case you only return the smallest trip: `T = [2-yellow, 3-blue, 1-yellow]`. We will use the order defined by `@</2`. Notice that this query `?- [2-yellow] @< [3-yellow]` succeeds.

The solution `T = [2-yellow, 3-blue, 1-yellow]` tells us we will drive a yellow highway from 1 to 2, then a blue highway from 2 to 3, and finally a yellow highway from 3 to 1.

The predicate `tour` will fail if `check` fails. Some examples:

```
highway(1,2,yellow).
highway(2,3,blue).
highway(1,3,yellow).

?- tour(T).
T = [2-yellow, 3-blue, 1-yellow]


highway(1,2,c).
highway(2,3,a).
highway(1,3,b).
highway(3,5,a).
```

```
highway(3,4,c).
highway(5,4,d).

?- tour(T).
T = [2-c, 3-a, 4-c, 5-d, 3-a, 1-b] ;
```