Programming Paradigms

Lab 12. Cut and negation as failure in Prolog

Non-unifiable

Exercise 11.1

Implement predicate `nonunifiable/2` that succeeds whenever its arguments cannot be unified. Provide three implementations:

1. First, using `=` and `\+`
2. Second, using `=` without `\+`
3. Third, using cut-fail combination, without `=` or `\+`

```
?- nonunifiable(test, test).
false
```

Partitioning a list

Exercise 11.2(a)

Implement predicate `partition/4` that takes a pivot and a list and produces two lists: one with elements less than the pivot and another one with elements greater than or equal to the pivot.

```
?- partition(3, [1, 5, 2, 6, 3, 0], L, G).
L = [1, 2, 0]
G = [5, 6, 3]
```

Partitioning a list efficiently

Exercise 11.2(b)

Improve predicate `partition/4`, making it more efficient without changing its meaning with green cuts.

```
?- partition(3, [1, 5, 2, 6, 3, 0], L, G).
L = [1, 2, 0]
G = [5, 6, 3]
```

Siblings

Exercise 11.3

Implement predicate `sibling/2` that succeeds whenever two people are siblings (they share a common parent). Make sure that a person cannot be its own sibling.

```
?- sibling(jack, jack).
false
```

DFS proper

Exercise 11.4

Implement predicate `path/3` that searches for a path between two nodes in an undirected graph.

```
?- path(1, 3, Path).
Path = [1, 4, 3]
```

```
connected(1,2).
connected(3,4).
connected(5,6).
connected(7,8).
connected(9,10).
connected(12,13).
connected(13,14).
connected(15,16).
connected(17,18).
connected(19,20).
connected(4,1).
connected(6,3).
connected(4,7).
connected(6,11).
connected(14,9).
connected(11,15).
connected(16,12).
connected(14,17).
connected(16,19).
```

Filtering unifiable terms

Exercise 11.5

Implement predicate `filterUnifiable/3` that takes two lists and a term. The second list is the output argument and should contain terms from the first list that unify with the given term. The variables in the first list should not be instantiated as a result of this predicate:

```
?- filterUnifiable([X, b, t(Y)], t(a), List).
List = [X, t(Y)]
```

Hint: consider using checks of the form `\+ term1 = term2`.

Eight queens problem

Exercise 11.6

Solve the eight queens problem.

Use green cuts to make the implementation efficient.