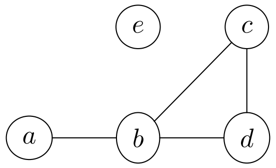
Declaratieve Talen
Prolog 2

1 Graphs



Address the following tasks using the above graph:

1. Transform the undirected graph in Prolog facts. Come up with an appropriate representation but make sure to explicitly represent both the nodes and edges. Since multiple representations are possible, you should pick a representation that allows you to address the following tasks in a straightforward way.

2. Implement a `neighbor/2` predicate that succeeds when two given nodes are direct neighbors. This predicate should be symmetrical such that in any given graph `neighbor(a,b)` and `neighbor(b,a)` yield the same outcome.

```
?- neighbor(a,b).
true.

?- neighbor(b,a).
true.

?- neighbor(a,e).
false.
```

3. Implement a `path/2` predicate that succeeds when a path exists between two given nodes. The predicate should not make any assumptions about the maximum length of the path, e.g. a path can have length 10. Investigate which nodes are reachable from node `a` by only instantiating the predicate's first argument. What strikes you about these solutions? Can you address this issue?

```
?- path(a,c).
true.

?- path(a,e).
...

?- path(a,X).
...
```

4. Resolve the issue with the `path/2` predicate by maintaining a list of visited nodes. When multiple paths exist between two nodes, we are interested in all possible simple paths (i.e. paths without cycles). Use the built-in predicates `\+/1` and `member/2`. Make sure that the arguments of `member/2` have been sufficiently instantiated at the time of the call.

   ```
   ?- path2(a,X).
   X = b ;
   X = c ;
   X = d ;
   X = d ;
   X = c ;
   false.

   ?- path2(a,e).
   false.
   ```

   Tip: Write a helper predicate `pathHelper/3` which explicitly maintains the list of visited nodes as an extra argument. Use this predicate to implement `path/2`.