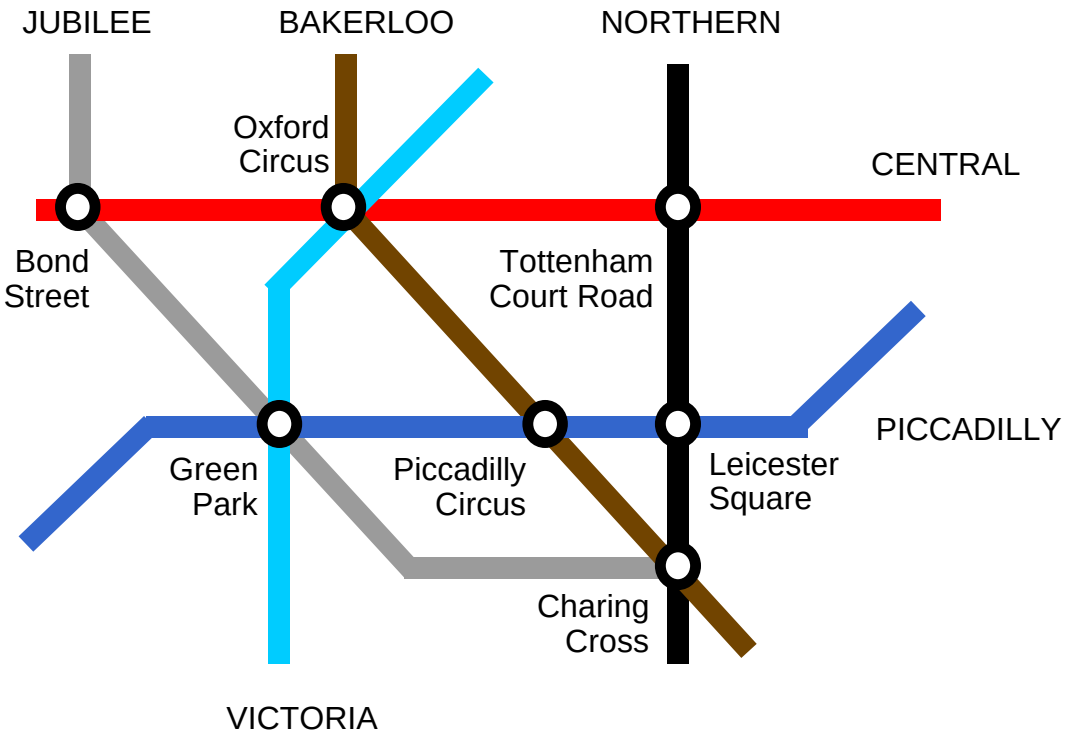


London Underground example


```
connected(bond_street,oxford_circus,central).
connected(oxford_circus,tottenham_court_road,central).
connected(bond_street,green_park,jubilee).
connected(green_park,charing_cross,jubilee).
connected(green_park,piccadilly_circus,piccadilly).
connected(piccadilly_circus,leicester_square,piccadilly).
connected(green_park,oxford_circus,victoria).
connected(oxford_circus,piccadilly_circus,bakerloo).
connected(piccadilly_circus,charing_cross,bakerloo).
connected(tottenham_court_road,leicester_square,northern).
connected(leicester_square,charing_cross,northern).
```

London Underground in Prolog (1)

London Underground in Prolog (2)

Two stations are nearby if they are on the same line with at most one other station in between:

```
nearby(bond_street,oxford_circus).
nearby(oxford_circus,tottenham_court_road).
nearby(bond_street,tottenham_court_road).
nearby(bond_street,green_park).
nearby(green_park,charing_cross).
nearby(bond_street,charing_cross).
nearby(green_park,piccadilly_circus).
…
```

or better

```
nearby(X,Y):-connected(X,Y,L).
nearby(X,Y):-connected(X,Z,L),connected(Z,Y,L).
```


Compare

```
nearby(X,Y):-connected(X,Y,L).
nearby(X,Y):-connected(X,Z,L),connected(Z,Y,L).
```

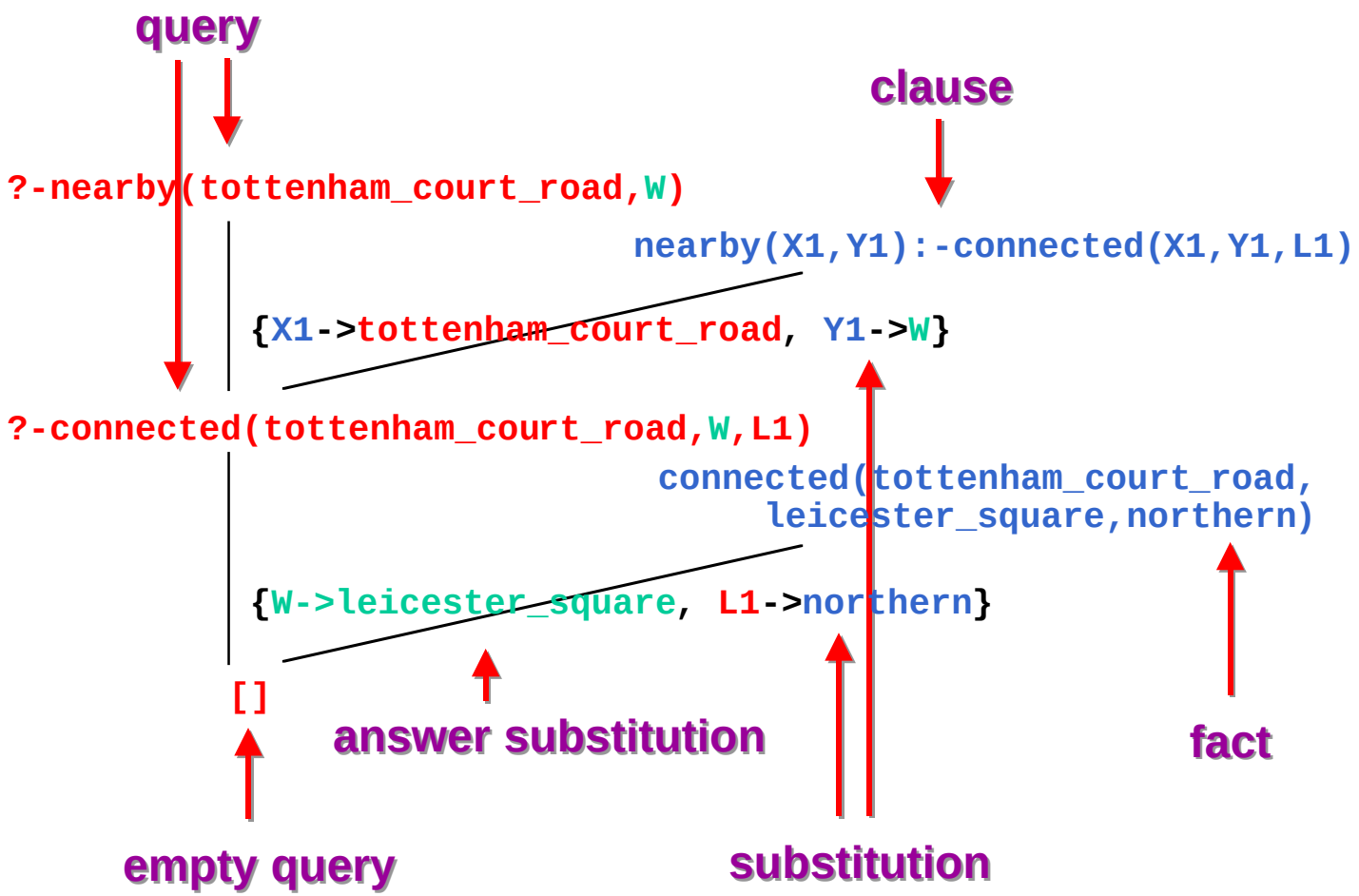
with

```
not_too_far(X,Y):-connected(X,Y,L).
not_too_far(X,Y):-connected(X,Z,L1),connected(Z,Y,L2).
```

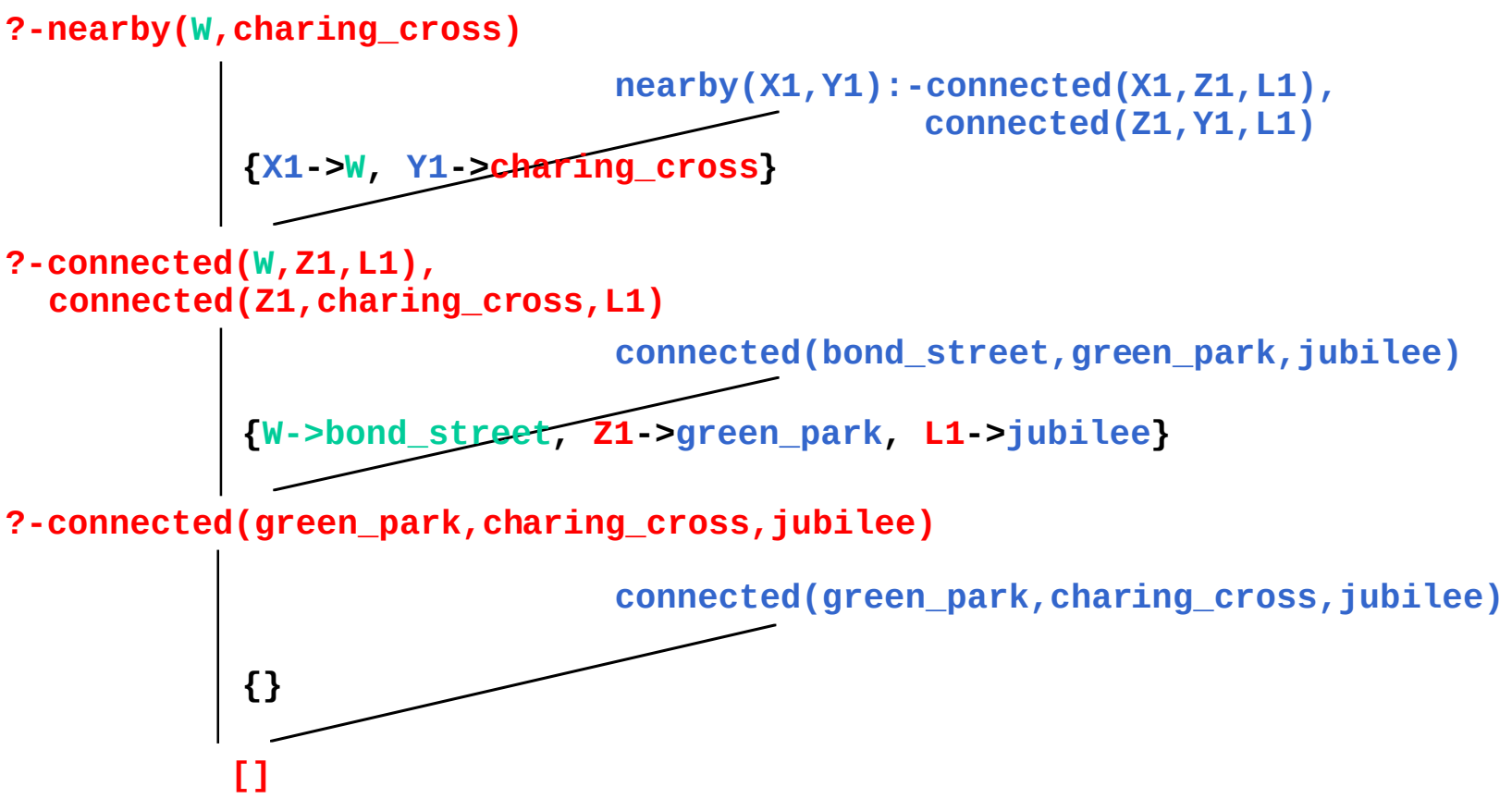
This can be rewritten with don't cares:

```
not_too_far(X,Y):-connected(X,Y,_).
not_too_far(X,Y):-connected(X,Z,_),connected(Z,Y,_).
```

Exercise 1.1



Proof tree



Exercise 1.2

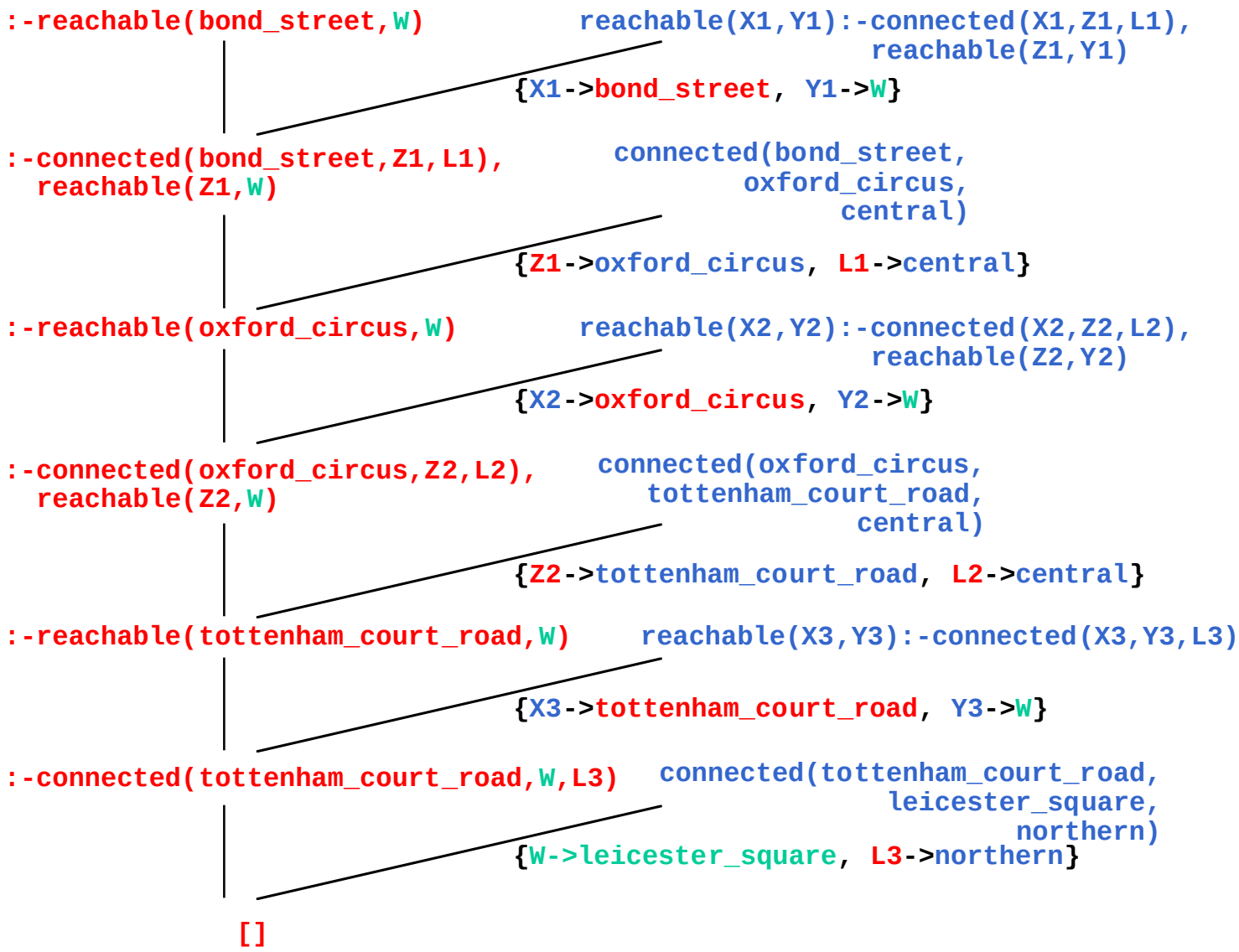
A station is reachable from another if they are on the same line, or with one, two, … changes:

```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L1),connected(Z,Y,L2).
reachable(X,Y):-connected(X,Z1,L1),connected(Z1,Z2,L2),
                connected(Z2,Y,L3).
```

…

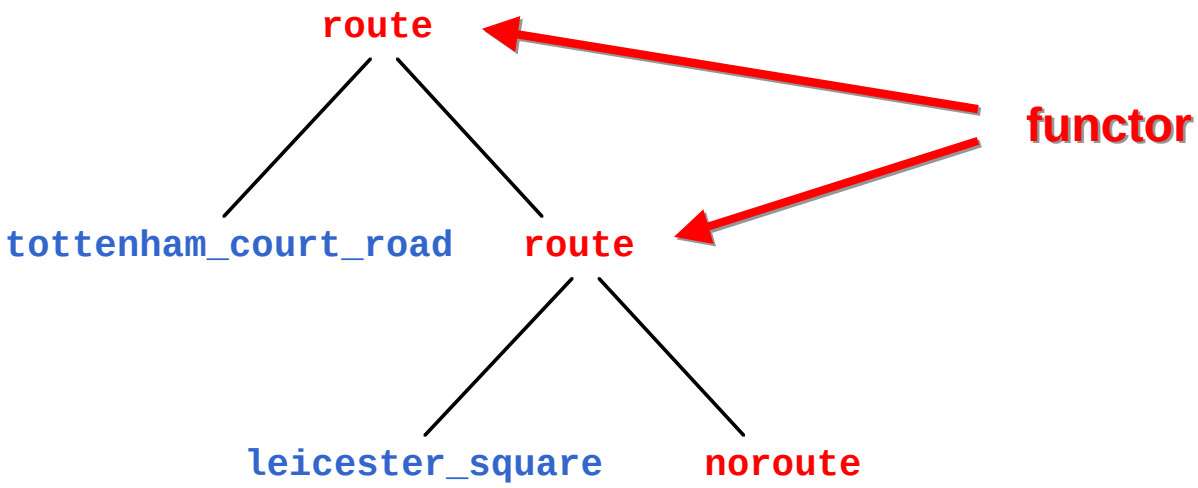
or better

```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L),reachable(Z,Y).
```

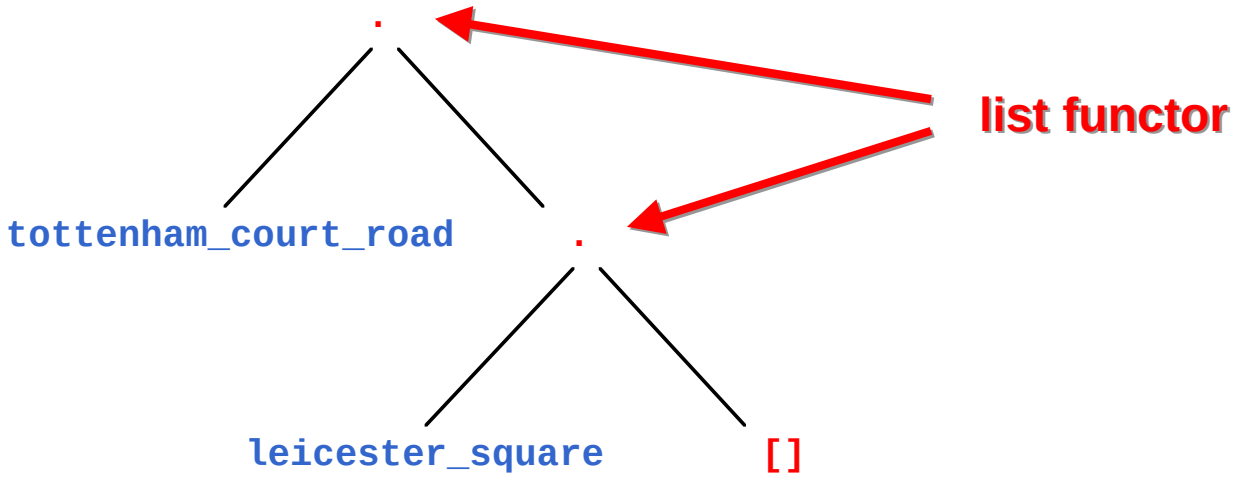
Recursion (2)


```
reachable(X,Y,noroute):-connected(X,Y,L).
reachable(X,Y,route(Z,R)):-connected(X,Z,L),
                           reachable(Z,Y,R).

?-reachable(oxford_circus,charing_cross,R).
R = route(tottenham_court_road,route(leicester_square,noroute));
R = route(piccadilly_circus,noroute);
R = route(picadilly_circus,route(leicester_square,noroute))
```

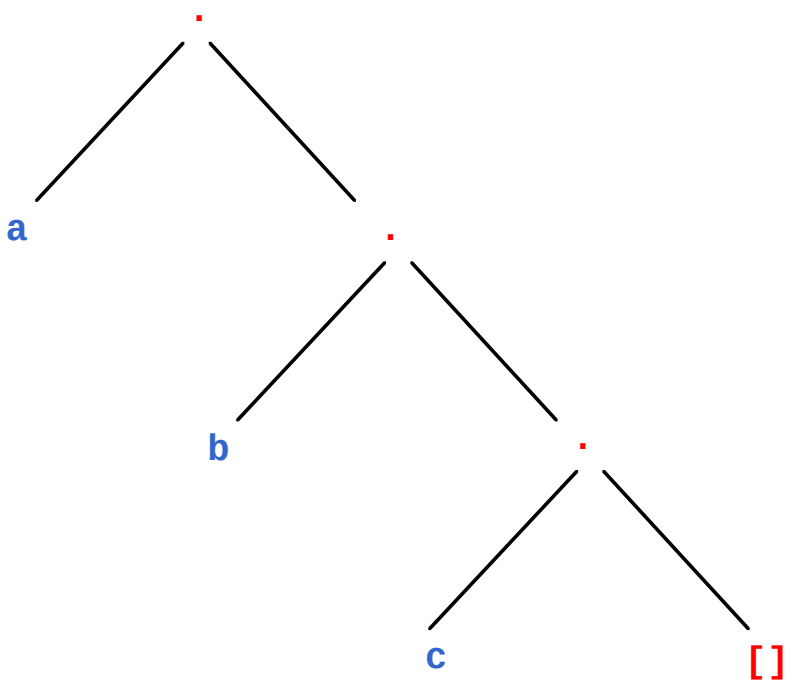


Structured terms

```
reachable(X,Y,[]):-connected(X,Y,L).
reachable(X,Y,[Z|R]):-connected(X,Z,L),
                       reachable(Z,Y,R).

?-reachable(oxford_circus,charing_cross,R).
R = [tottenham_court_road,leicester_square];
R = [piccadilly_circus];
R = [picadilly_circus,leicester_square]
```



Lists



This list can be written in many ways:

```
.(a,.(b,.(c,[])))

[a|[b|[c|[]]]]

[a|[b|[c]]]

[a|[b,c]]

[a,b,c]

[a,b|[c]]

…
```

Lists (2)

- Lists of arbitrary length:

```
list([]).
list([First|Rest]):-list(Rest).
```

- Lists of even length:

```
evenlist([]).
evenlist([First,Second|Rest]):-evenlist(Rest).
```

- Lists of odd length:

```
oddlist([One]).
oddlist([First,Second|Rest]):-oddlist(Rest).
```

or alternatively:

```
oddList([First|Rest]):-evenlist(Rest).
```

Exercise 1.4

- ☞ Prolog has very simple syntax
  - ✓ constants, variables, and structured terms refer to objects
    - variables start with uppercase character
    - functors are never evaluated, but are used for naming
  - ✓ predicates express relations between objects
  - ✓ clauses express true statements
    - each clause independent of other clauses
- ☞ Queries are answered by matching with head of clause
  - ✓ there may be more than one matching clause
    - query answering is search process
  - ✓ query may have 0, 1, or several answers
  - ✓ no pre-determined input/output pattern (usually)

Summary