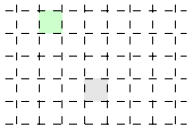
Logic and Functional Programming

Prolog: Searching in an infinite space
A possible solution

Every room can be represented by a pair `(X,Y)` of integers where `X` indicates how far up/down it is from the origin, and `Y` indicates how far left/right it is from the origin. For example, in the figure below the origin is the grey room represented by `(0,0)`, and the green-marked room is represented by `(-2,3)`.



The internal state of the robot (`ProgramData`) could be a tuple

```
(N,(X,Y),S,(XL,YL))
```

where: `N` is the number of boxes that the robot must deliver; `(X,Y)` is the current location of the robot; `S` is `true` if the robot holds a box, and `false` otherwise; and `(XL,YL)` is the last location visited by the robot.

Initially, the robot is in the origin `(0,0)`, it holds no box, and is asked to deliver `N` boxes. Therefore, the initialization step of the robot should be `(N,(0,0),false,(0,0))`.

```
initMyData(N,(N,(0,0),false,(0,0))).
```

1 The actions of the robot

Action `done`

This action is performed when the robot is in the origin, it carries no box, and there are no more boxes to deliver. In this case, the state of the robot is `(0,(0,0),false,P)`.

```
perform((0,(0,0),false,P),_,done,(0,(0,0),false,P)):-!.
```

Action deliverBox

This action is performed when the robot is in the origin and carries a box. In this case, the state of the robot is `(N,(0,0),true,P)` and its new state should be `(N1,(0,0),false,P)` because

1. The number of boxes to be delivered decreases by 1, and
2. after delivering the box the robot has no box in its arms.

```
perform((N, (0,0),true,P),_,deliverBox,
        (N1,(0,0),false,P)):-!,N1 is N-1.
```

Action pickBox

This action is performed when the robot does not carry a box, it is in the last visited room, and there is a box in it. In this case, the state of the robot is `(N,(X,Y),false,P)`, and the new state will be `(N,(X,Y),true,(X,Y))`. The robot will remember that the `(X,Y)` is the last visited room, where he found a box.

```
perform((N,(X,Y),false,(X,Y)),true,pickBox,
        (N,(X,Y),true, (X,Y))):-!.
```

Actions to return to the last visited room

These are `move(D)` actions which are performed when the robot is not yet in the last visited room. The value of `D` depends on the relative positions `(X,Y)` of the robot and `(Xl,Yl)` of the last visited room:

If `X` $<$ `Xl` then `D` $=$ `east`.
If `X` $>$ `Xl` then `D` $=$ `west`.
If `X` $=$ `Xl` and `Y` $<$ `Yl` then `D` $=$ `north`.
If `X` $=$ `Xl` and `Y` $>$ `Yl` then `D` $=$ `south`

The rules which perform these actions are

```
perform((N,(X,Y), false,(XL,YL)),_,move(east),
        (N,(X1,Y),false,(XL,YL))):-X<XL,!,X1 is X+1.
perform((N,(X,Y), false,(XL,YL)),_,move(west),
        (N,(X1,Y),false,(XL,YL))):-X>XL,!,X1 is X-1.
perform((N,(XL,Y), false,(XL,YL)),_,move(north),
        (N,(XL,Y1),false,(XL,YL))):-Y<YL,!,Y1 is Y+1.
perform((N,(XL,Y), false,(XL,YL)),_,move(south),
        (N,(XL,Y1),false,(XL,YL))):-Y>YL,!,Y1 is Y-1.
```

Actions to search box in a new, unvisited room

These are `move(D)` actions with `D` $\in$ {`north`, `south`, `west`, `east`} which are performed when the robot carries no box, it is in the last visited room, and there is no box in the room. These actions move the robot on the spiral suggested in Labwork 11.

```
perform((N,(XL,YL),false,(XL,YL)),false,move(east),
        (N,(X1,YL),false,(X1,YL))):-
     YL=<0,abs(XL)=< -YL,!,X1 is XL+1.
perform((N,(XL,YL),false,(XL,YL)),false,move(north),
        (N,(XL,Y1),false,(XL,Y1))):-
     XL>0,abs(YL)<XL,!,Y1 is YL+1.
perform((N,(XL,YL),false,(XL,YL)),false,move(west),
        (N,(X1,YL),false,(X1,YL))):-
     YL>0,-YL<XL,XL=<YL,!,X1 is XL-1.
perform((N,(XL,YL),false,(XL,YL)),false,move(south),
        (N,(XL,Y1),false,(XL,Y1))):-
     XL<0,XL<YL,YL=< -XL,!,Y1 is YL-1.
```

Actions to bring box to the origin

These are `moveWithBox(D)` actions with `D` $\in$ {`north`, `south`, `west`, `east`} which are performed when the robot holds a box and is not yet in the origin. The value of `D` depends on the relative positions `(X,Y)` of the robot and origin `(0,0)`:

If `X` $<$ `0` then `D` $=$ `east`.
If `X` $>$ `0` then `D` $=$ `west`.
If `X` $=$ `0` and `Y` $<$ `0` then `D` $=$ `north`.
If `X` $=$ `0` and `Y` $>$ `0` then `D` $=$ `south`

The rules which perform these actions are

```
perform((N,(X,Y), true,P),_,moveWithBox(east),
        (N,(X1,Y),true,P)):-X<0,!,X1 is X+1.
perform((N,(X,Y), true,P),_,moveWithBox(west),
        (N,(X1,Y),true,P)):-X>0,!,X1 is X-1.
perform((N,(0,Y), true,P),_,moveWithBox(north),
        (N,(0,Y1),true,P)):-Y<0,!,Y1 is Y+1.
perform((N,(0,Y), true,P),_,moveWithBox(south),
        (N,(0,Y1),true,P)):-Y>0,!,Y1 is Y-1.
```

2 Putting all rules together

The program can be downloaded from here.