CS322 Fall 1999
Module 10 (Representing Actions)
Assignment 10

Due: 1:30pm, Friday 19 November 1999.

In the file `~cs322/cilog/delrob_strips.pl` (also available on the web) is a STRIPs representation of the delivery robot. There is also a STRIPs planner in `~cs322/cilog/strips_strips.pl` (also available on the web). The bottom of this file shows some queries you can try, for example, the following is an example query (reformatted to show the plan better):

```
cilog: ask achieve(at(k1,lab2),init,S,6,N).
Answer: achieve(at(k1,lab2),init,
          do(move(rob,o103,lab2),
            do(unlock(rob,door1),
              do(move(rob,mail,o103),
                do(pickup(rob,k1,mail),
                  do(move(rob,o103,mail),
                    do(move(rob,o109,o103),
                      init))))))),6,0).
```

You will need to play with these programs to do this assignment.

Question 1

Suppose that we also have buckets that can be filled by a tap in room o111. Assume there is an action *fill*(*B*) to fill a bucket *B* that the robot is carrying in room *o*101. Bucket *b*1 is initially empty in the storage room.

An autonomous agent can wet (using the action *wet*(*Obj*, *B*)) an object *Obj* by pouring water (from the bucket *B*) onto that object, as long as the agent is carrying the full bucket at the same location as *Obj*. Objects stay wet. Suppose we have the predicate *is_wet*(*Obj*) that is true if *Obj* is wet.

(a) Represent the actions and any needed derived relations, and the initial situation, so you can answer queries such as the following (again we have formatted the output to make it more readable):

```
cilog: ask achieve(is_wet(parcel),init,S,8,R).
Answer: achieve(is_wet(parcel),init,
     do(wet(parcel,b1),
       do(move(rob,o109,storage),
         do(move(rob,o111,o109),
           do(fill(b1),
             do(move(rob,o109,o111),
               do(move(rob,storage,o109),
                  do(pickup(rob,b1,storage),
                    do(move(rob,o109,storage),init)))))))),8,0).
```

(b) Explain why the STRIPS planner can't find the shortest plan. [You have to think about what is the shortest plan.]

Solution

(a) You can get the complete axiomatization in `delrob_strips_wet.pl`.

The relevant parts are:

```
% wet(Obj,B) is the action of wetting the Obj with the water in bucket B
preconditions(wet(Obj,B),
    bucket(B) & autonomous(Ag) & full(B) & carrying(Ag,B) &
    at(Obj,Loc) & at(Ag,Loc)).
addlist(wet(Obj,B),[is_wet(Obj)]).
deletelist(wet(Obj,B),[full(B)]).

% fill(B) is the action of filling bucket B
preconditions(fill(B),
   bucket(B) & autonomous(Ag) & carrying(Ag,B) & at(Ag,o111)).
addlist(fill(B),[full(B)]).
deletelist(fill(B),[]).

primitive(full(_)).
primitive(is_wet(_)).

bucket(b1) <= true.

holds(sitting_at(b1,storage),init).
```

(b) The shortest plan is for rob to go to the storage and pick up both the parcel and the bucket then go to room o111, fill the bucket then immediately wet the parcel. This has one fewer steps than the plan found. It won't find this as moving the parcel isn't considered at all when trying to find either goal.

Note that the regression planner can find this plan.

Question 2

Using SLD resolution directly on the on the situation calculus or STRIPs representation can be very slow. How slow? That is what we'll try to estimate here.

In the file `~cs322/cilog/back_strips.pl` (also available on the web) is a direct meta-interpreter for STRIPs in CILog.

Note that cilog does depth-bounded search. The command *bound N*. sets the depth-bound to N.

Consider the query:

```
ask holds_in(carrying(rob,parcel)& sitting_at(rob,lab2),S).
```

Assume that we are using CILog to do an iterative deepening search. You are to estimate for how long the search takes for the complete search at depth $D-1$ where D is the depth needed for the shortest solution. To do this you should:

(a) Estimate the smallest bound needed to find a plan. Hint: how many actions are needed to solve this problem? How does the number of steps relate to the depth-bound needed? [If you know the shortest plan it's not computationally difficult to find the depth-bound needed to prove it.] Justify your estimate.

(b) Estimate the branching factor of the search tree. To do this you should look at the time for a complete search at level $k+1$ versus a complete search at level k. You should justify your answer both experimentally (by running the program) and theoretically (by considering what is the branching factor). You don't need to run cases with a large runtime to do this.

(c) Based on your answers to parts 1 and 2, and the time you found for some run of the program for a small bound, estimate the time for a complete search of the search tree at the highest depth that you won't find a solution. Justify your estimate. [Also say what computer you are using.]

Solution

(a) Estimate the smallest bound needed to find a plan.

We try to work out the shortest plan. The one from STRIPS looks pretty good. So we can find out at which depth bound it fails. If you try the query:

```
ask holds_in(carrying(rob,parcel)& sitting_at(rob,lab2),
    do(move(rob,o103,lab2),do(unlock(rob,door1),
      do(move(rob,mail,o103),do(pickup(rob,k1,mail),
        do(move(rob,o103,mail),do(move(rob,o109,o103),
          do(move(rob,storage,o109),do(pickup(rob,parcel,storage),
            do(move(rob,o109,storage),init)))))))))    ).
```

You find out that it fails at depth bound 38 and succeeds at depth bound 39. You can do find the depth-bound by binary search.

So we want to estimate the run time to find all solutions for depth bound 38.

(b) Estimate the branching factor of the search tree.

```
cilog: bound 10.
cilog: ask holds_in(carrying(rob,parcel)& sitting_at(rob,lab2),S).
Query failed due to depth-bound 10.
 Runtime since last report: 110 ms.
     [New-depth-bound,where,ok,help]: 11.
Query failed due to depth-bound 11.
 Runtime since last report: 220 ms.
     [New-depth-bound,where,ok,help]: 12.
Query failed due to depth-bound 12.
 Runtime since last report: 500 ms.
     [New-depth-bound,where,ok,help]: 13.
Query failed due to depth-bound 13.
 Runtime since last report: 1050 ms.
     [New-depth-bound,where,ok,help]: 14.
Query failed due to depth-bound 14.
 Runtime since last report: 1870 ms.
     [New-depth-bound,where,ok,help]: 15.
Query failed due to depth-bound 15.
 Runtime since last report: 3400 ms.
     [New-depth-bound,where,ok,help]: 16.
```

```
Query failed due to depth-bound 16.
 Runtime since last report: 6480 ms.
     [New-depth-bound,where,ok,help]: 17.
Query failed due to depth-bound 17.
 Runtime since last report: 11810 ms.
     [New-depth-bound,where,ok,help]: 18.
Query failed due to depth-bound 18.
 Runtime since last report: 21580 ms.
     [New-depth-bound,where,ok,help]:
```

If you look at the ratios of times of the runtime for depth k divided by the runtime for depth $k-1$, then the last three ratios are, in reverse order: 1.827, 1.822, 1.906. Empirically it seems as though the branching ratio is about 1.8.

(c) Based on your answers to parts 1 and 2, and the time you found for some run of the program for a small bound, estimate the time for a complete search of the search tree at the highest depth that you won't find a solution.

So we would like to estimate how long it takes to find a solution at depth 38. Note that we expect exponential grown (as can be seen for the previous runtimes).

We need 20 more steps than the failing 18 steps which took 20 seconds.

So we would expect it to take about $1.8^{20} * 20 seconds = 127482 * 20 = seconds = 2549640 seconds$, which is about 29 days.

I was running this on a Pentium 266 using CILog in Sicstus prolog. Pender is about 4 times as slow, so will take about 115 days of runtime. CILog in SWI Prolog takes about 50% longer, so would take about 44 days.

Question 3

[Bonus] Estimating values is an important skill.

How many rain drops fall on Vancouver in a typical November?

You can get 2 marks for being within one order of magnitude and 1 mark for being within 2 orders of magnitude.

Solution

The average rainfall in Vancouver for November is 170mm.
Source: http://www.cmc.ec.gc.ca/climate/normals/BCV004.HTM.

The area of Vancouver is 113.1 square km
Source: http://www.city.vancouver.bc.ca/ctyclerk/info_depts_boards/statistics.html.

Raindrop size: raindrop is about 1 cubic mm.
Source: http://www.gi.alaska.edu/ScienceForum/ASF2/236.html
http://wwwga.usgs.gov/edu/raindropsizes.html

$$113 * 10^6 * 10^6 * 170mm \approx 10^{16} = 10000000000000000$$