MAS: Activity 2 – Single BDI agent

Alexandru Sorici

15.03.2021

The problem: In the **blocks world**[1] an initial state is given. Blocks are identified by capital letters. Blocks can be placed one on top of the other ($ON(A, B)$), or on the table ($ONTABLE(A)$) (which can hold an unlimited number of stacks). The blocks and the table form a **station**. Blocks can be moved by an **arm** situated above the station. It can pick blocks and put them down, one block at a time. The arm is controlled by an agent, which is assigned a final goal state. The problem is solved when, by using legal operations (see below), the initial state is transformed into the final state. The environment, however

is dynamic and **does not** allow for a deterministic planning. The dynamicity of the environment simulates the possible presence of other agents in the system, which our agent is unable to perceive or model. What can happen in the environment:

- blocks may disappear from the top of a stack (including blocks that are on the table);
- blocks that have previously disappeared may appear again, on top of an existing stack;
- blocks may fall down to the table, from the top of a stack;
- blocks may teleport (instantaneously) from the top of one stack to the top of another.

NOTES:

1. the order of stacks on the table is not relevant, only relations between blocks are;
2. dynamic actions are only performed using legal operations (taking blocks from the top of stacks or from the table; putting blocks on the table or on top of stacks);
3. dynamic actions will only affect stacks that are not **currently observed** by any agent(s);

Example

```
*1               *1                 *1          *1                *1
<>               <A>                <>          <B>               <>


[A]                                                               [B]
[B]  [C]         [B]  [C]           [B]  [C]  [A]   [C]  [A]      [C]  [A]
--------------------------------------------------------------------------
initial state    Unstack(A, B)      Putdown(A)  Pickup(B)         Stack(B, C)
                                                                  final state
```

The solution: use a **single-minded BDI** agent to solve the problem. The agent receives as perception the current state of the station (the station does **not** contain the block held by the arm), as a list of FOPL predicates ($ON(X, Y)$, $ONTABLE(X)$, or $CLEAR(X)$). The agent controls the arm to take a block ($Pickup(C)$ from the table or $Unstack(A, B)$) or to put it down ($Putdown(A)$ on the table or $Stack(B, C)$). The agent can also *lock* a block into place ($Lock(C)$), such that the dynamic actions of the environment can no longer affect it. A locked blocked *cannot* be *unlocked*.

[1] http://en.wikipedia.org/wiki/Blocks_world

The state of the arm can be checked using predicates $ARMEMPTY$ and $HOLD(X)$. If more blocks are stacked one on top of the other, only the top block can be unstacked. A block can only be on the table, stacked on another block, or held by the arm.

Tasks:

- Devise a **planning method** to solve the given problem. (see Page 3 for some ideas - easiest one is UNSTACK-STACK)
- Implement the algorithm of the agent as a BDI process, by implementing the `reviseBeliefs()` and `plan()` methods.
  - use the `reviseBeliefs` function to update the agent's memory of the world.
  - use ***desires*** to model meta-actions such as "Find block A", "Build tower ABC"
  - instantiate your desires as ***intentions*** by creating **partial plans** to avoid replanning when a needed block disappears;
  - adapt the current plan being executed to adjust for blocks that have moved: either *re-plan* for same goal, or *switch to* other partial plan
- Test the planning onto the various tests; make plans short (no obviously redundant actions). Display your plan using the `statusString()` method.
- Use the `statusString()` method of the agent to monitor the agent.

BDI elements:

- Beliefs – what the agent believes about the world;
- Desires – what the agent would want to be true;
- Intentions – what the agent plans for;
- brf – belief revision;
- options – what non-contradicting, possible goals the agent can set itself;
- filter – goals the agent should plan for (commit to);

Heads-up:

- `Blocks` contains all methods needed for reading the state of a station and change it. Read the documentation of `contains` and `exists` and remember the difference.

Testing: Change the test suite by changing `TEST_SUITE` in `MyTester`.

Planning algorithms:

Unstack-Stack, Gupta and Nau, 1 and 2:
http://ai.cs.unibas.ch/_files/teaching/hs12/search-opt-seminar/slides/08_blocks_world_revisited.pdf
SATPLAN: http://www.inf.unibz.it/~fillottrani/Class4p.pdf
Some examples of planning: http://www.cs.cf.ac.uk/Dave/AI2/node116.html
Genetic algorithms: http://www.inf.ed.ac.uk/teaching/courses/gagp/slides06/4gagplect10.pdf
STRIPS, Graphplan and other algorithms:
http://www.ailab.si/ivan/datoteke/dokumenti/31/100_60_planning_postgraduate_course_2010.pdf