Marcantonio Soda Jr
HW1

1)

Behavioral equivalence between a standard agent and state-based agent can be thought of as for any set of percepts as input, both will produce the same actions as output. In other words, the proof must show that for any defined standard agent one must be able to define a state-based agent that generates the same actions based on the same percepts.

Consider the behavior definition of a state-based agent defined in the Woolridge book:

see: $E \rightarrow Per$
next: $IxPer \rightarrow I$
action: $I \rightarrow Ac$

Which shows that the agent updates its internal state every time it sees and the action that it selects is based on the updated internal state.

We can then compare that with the definition of a standard agent:

Ag : R^E -> Ac

Which shows that the agent maps runs (which ends with an environment state) to actions. Therefore a standard agent chooses its action based on the history of the system that it has recorded.

The notion of choosing an action based on the system's history no different than the state-based system consulting its internal state in order to calculate its actions. The two are behaviorally equivalent. Therefore state based agents are equivalent in expressive power to standard agents.

2) a) I assume that the agent begins facing north on cell (0,0) as described in the book example.

Rules:

$In(x,y) \wedge Dirt(x,y) \rightarrow Do(suck)$

$In(x,y) \wedge \neg Dirt(x,y) \wedge \neg wall(forward) \wedge \neg Visited(forward) \rightarrow Set Visited(x,y)\,;\ Do(forward)$

$In(x,y) \wedge \neg Dirt(x,y) \wedge wall(forward) \rightarrow Do(right)$

Predicates:

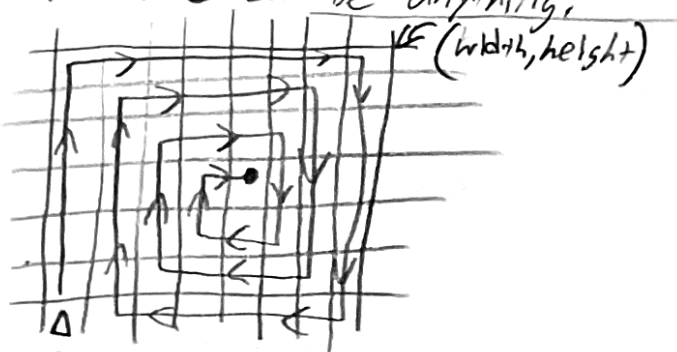
wall(dir): Returns true if there is a wall in the direction (forward, left, right) specified. It is based on the direction that the agent is actually facing, the position of the agent, and the world size.

SetVisited(x,y): Adds the x,y pair to a list of coordinates that represent where the agent has been.

Visited(x,y): Returns true if the x,y pair is listed in the agent's state.

Behavior:

Based on the rules that I provided, the agent should travel forward until it's going to hit a wall or a previously explored position, then it should turn right and repeat. The world size can be anything.



b) I will base my subsumption architecture structure off of the one in the book about 'Steel's Mars Explorer Experiments'.

L1 Dirt Cleaning:
- IF there is dirt on the cell
- THEN suck it

L2 Wall Avoidance:
- IF the agent is facing a wall
- THEN turn right

L3 Exploration
- IF true
- THEN turn right OR go forward

- L3 may seem a bit strange due to the fact that it is randomly moving forward or turning right, but this is the simplest way to ensure the eventual full traversal of the graph due to the limitation of this needing to be a fully reactive agent and thus being unable to track visited cells.
- The answer to the three questions is somewhat nuanced due to the specific nature of the problem

☆ The subsumption-based approach (SBA) is not as "good" as the deductive logic approach (DLA). This is mostly due to the fact that the DLA is able to track state, leading to an optimal solution. The SBA is a purely reactive agent, so state is unable to be tracked. The SBA is far more intuitive and extensible than the DLA. The DLA is simpler. Its simplicity would be easier to illustrate given a more complex problem. There are a great many problems which are infeasible to solve using a DLA which can be solved using a SBA. Please note that a SBA which would be able to track state would be able to perform just as well as my DLA example.

3 closed(d, Do(a(e), s)) $\leftrightarrow$

(a == explode(e) $\wedge$ near(e, d, s)) $\vee$

(a == close(d)) $\vee$

(a != open(d) $\wedge$ closed(d, s))

In english:

The door(d) will be in a closed state after an action(a) on object(e) in s(s) if and only if d is near an exploading object or the action was to close d or the action is not to open d and it was already closed.

4)

Behavior Layer:

The behavior layer is the lowest layer in the InteRRaP architecture. This layer directly receives percepts from the world interface. It also has the information that it has stored in the world model to go off of. Some of the information that might be contained in the world model include the location of obstacles, the location of other agents, the location of detected samples, and any other information that might be useful in describing the current state of the environment. The decisions made by the behavior layer are low-level compared to the other layers. It will use the information from the world model, new percepts, and the plan given to it by the planning layer (which may have been influenced by the cooperation layer) to decide what action to take next. Some actions that it may decide to take are movement-based actions (move north, south, east, west) or actions on a sample (pickup/drop sample). The behavior layer will also pass its observations of the environment to the planning layer for its use in creating/modifying a plan.

Planning Layer:

The planning layer is responsible for devising a plan of action for the agent to take in an effort to reach the design goal. It will use the information from the behavior layer, the information from the cooperation layer, and the information from its own planning knowledge to devise a plan. The planning layer should take into account the relevant information that the behavior layer has in the world model such as locations of obstacles, and information from the cooperation layer such as what another agent's goals might be to calculate a path. The planning knowledge contains information such as agent's goals, and the current plan. It may calculate the path using some sort of path-finding algorithm such as A*. The planning layer will pass its plan down to the behavior layer to help it decide what action to take next and to the cooperation layer to help it effectively communicate its intent to the other layers in the interest of achieving a collective goal.

Cooperation Layer:

The cooperation layer is responsible for ensuring that all of the agents are cooperating in such a way that allows the collective goal to be reached the most efficiently. It will take the plan from the planning layer and broadcast the agent's intent to the other agents. It will also receive broadcasts from the other agents and pass the relevant information along to the planning layer to ensure that the plan accounts for the other agent's goals. For example, if the agent plans to go to a particular sample, but the cooperation layer receives a broadcast from an agent that is closer to the sample that it intends to collect it, the cooperation layer should "advise" the planning layer to plot a course to a more economic sample. The social knowledge had by the cooperation layer may contain information about the other agents in the system such as their capabilities (if they are not all identical) and goals. It will also contain information on how the agents should best cooperate.

5a)

Predicates:

In(obj, x,y): True if the obj (agent or sample) is in position x,y.
Obstacle(x,y): True if an obstacle exists at location x,y.
Holding(): True if the agent is holding a sample.
InMothership(): True if the agent is at the mothership.

Rules:

GoNorth(agent,x,y):
pre: {In(x,y), ~Obstacle(x,y+1)}
del: {In(x,y)}
add: {In(x,y+1)}

GoSouth(agent,x,y):
pre: {In(x,y), ~Obstacle(x,y-1)}
del: {In(x,y)}
add: {In(x,y-1)}

GoEast(agent,x,y):
pre: {In(x,y), ~Obstacle(x+1,y)}
del: {In(x,y)}
add: {In(x+1,y)}

GoWest(agent,x,y):
pre: {In(x,y), ~Obstacle(x-1,y)}
del: {In(x,y)}
add: {In(x-1,y)}

PickupSample(agent,sample,x,y):
pre: {In(agent,x,y), In(sample,x,y), ~Holding}
del: {In(sample,x,y)}
add: {Holding()}

DropSample(agent,sample,x,y):
pre: {In(agent,x,y), Holding, InMothership()}
del: {Holding()}
add: {In(sample,x,y)}

5b)

```
PLAN {
        NAME: move_randomly
        GOAL: at_sample
        CONTEXT: ~carrying_sample()
        BODY:
                direction dir
                infinite_loop:
                        dir = random dir
                        if Obstacle(dir) continue
                        else break
                Go(dir)
}
PLAN {
        NAME: goto_mothership
        GOAL: at_mothership
        CONTEXT:
                carrying_sample()
                ~at_mothership()
        BODY:
                direction dir
                if gradient(north): dir = north
                elif gradient(south): dir =  south
                elif gradient(east): dir =  east
                elif gradient(west): dir =  west
                loop until at_mothership():
                        if Obstacle(dir):
                                Go(random direction not dir)
                        Go(dir)
}
PLAN {
        NAME: pickup_sample
        GOAL: carrying_sample
        CONTEXT:
                at_sample()
                ~carrying_sample()
        BODY: pickup_sample()
}
PLAN {
        NAME: drop_sample
        GOAL: ~carrying_sample
        CONTEXT:
                at_mothership()
                carrying_sample()
        BODY: drop_sample()
}
```

Some plans were difficult to convey without taking some liberties in extending the syntax described in the Jam example. For example, it would have been more cumbersome and verbose (although not impossible) to convey the goto_mothership plan without the existence of variable declarations and looping. I also decided to add this notion of a “random direction” which is just a shorthand of me trying to describe that a random direction should be chosen. This is easy to convey to a person on paper, but this syntax is definitely too expressive for an actual application.

6)

To effectively come up with a hypothesis as to why behavior-based agents are generally more robust than plan-based, cooperative agents, it is important to establish a concrete definition of robustness in the context of the paper. The definition given by the authors of the paper is, “Robustness is measured by the success ration p, which is the ratio of successfully finished tasks to the total number of tasks given to the agent” Robustness can be defined as a metric which measures the ability of an agent to recover from failures, rather than being overly sensitive to disturbances or minor environmental events. The paper identifies three sources of failures, failures due to local maxima, deadlock situations caused by conflicts, and failures due to multiple conflicts that could not be adequately recognized and handled by agents. It stands to reason that the main culprit for the higher number of failures in plan-based agents compared to behavior based agents is deadlocks. The paper states that randomness is a powerful tool for resolving deadlocks. Behavior based strategies have an inherit degree of randomness baked in while cooperative strategies do not. If plan-based agents are too “stubborn” and fail to come to a consensus on a collective action to take, a deadlock will cause the task to fail. Deadlocks are therefore much less likely to plague behavior-based agents because they are less likely to suffer from cooperation-based conflicts. The two behavior-based agents that the paper defines are BCR, BCH, and RWK. BCR uses randomness to resolve its conflicts, meaning that deadlocks are very unlikely if not impossible. BCH employs a dodging technique which is not random but is able to avoid (in most, but not all cases) deadlocking due to its simple and efficient nature. RWK’s actions are completely random which makes deadlocks reasonably infeasible. In contrast with LCH and LCC, deadlocks are inherently much less likely leading to a more robust agent.

Succinctly, I hypothesize that behavior-based strategies are more robust than plan-based agents because behavior-based agents are less likely to fail on a given task due to a cooperation-caused deadlock due to the fact that most of the behavior based agents have some degree of randomness baked into their conflict resolution measures.