02285 AI and MAS, SP19
Exercises for week 6, 12/3-19

Exercise 1 (BDI for the hospital domain)

Consider the single-agent version of the hospital domain (programming project). Below we will consider using version 2 of the BDI architecture (cf. the slides from week 5) to control an agent in this domain.

a) To use the BDI architecture for agents in the hospital domain, we first have to think about what the *intentions* should be in this domain. What would be the consequence of choosing the intentions to be individual move, push and pull actions?

b) What would be the consequence having just one possible intention $SolveLevel$ which is to get all goal cells covered by boxes?

c) As should be clear from the two previous questions, we need intentions to be high-level actions in the sense used in hierarchical tasks networks (HTNs). Devise a set of relevant such high-level actions, that is, define a relevant set Int of all possible intentions. *Note*: There are many possible answers here, depending on how you would like your BDI loop to function.

d) Sketch how version 2 of the BDI architecture could work, given your choice of intention set from the previous question. That is, describe informally what should be implemented in each step of the agent control loop on the slides. In the step $plan(B, I)$, you can assume that you use $A^\star$ with a suitable heuristics. Remember also to describe this heuristics.

e) Consider how your agent control loop would run on a couple of example levels, e.g. this one:

```
+++++++
+  0  +
+A+ +a+
+B+ +b+
+C+ +c+
+++++++
```

f) Explain how your BDI-based solution in most cases reduces on the required search compared to just running $A^\star$ on the level of basic actions.

g) Using $A^\star$ with a heuristics in the step $plan(B, I)$ is likely to still induce too much search to scale well. Sketch a HTN approach where the intentions are refined into other high-level actions before finally being made into plans at the level of basic actions.

h) The BDI approach can clearly be extended to the multi-agent setting. We can let each agent run its own BDI control loop (decentralised solution), and let each agent have its own subset of intentions derived from the intentions considered above (the intentions involving goals of the same color as the agent itself). However, the generalisation to the multi-agent case would require communication, as one agent might need help from another agent to achieve its own intention. Give a simple example of a multi-agent level illustrating this.

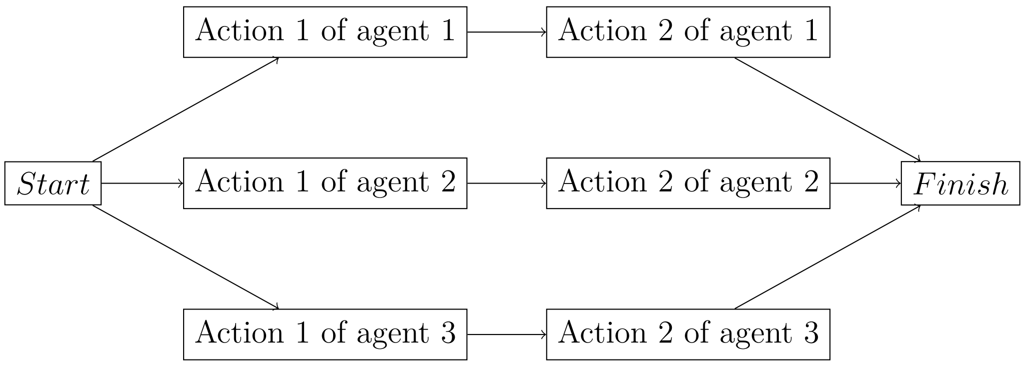
Exercise 2 (Plan merging using POP)

We now consider the multi-agent version of the hospital domain. We assume that each agent is provided with its own planning engine (decentralised solution). For simplicity, we assume that there is at most one agent of each color, so that the goals of the individual agents are disjoint. We also restrict attention to levels where each agent can initially achieve its goals without the help of the other agents.

a) If the agents make individual plans, and afterwards try to execute them concurrently, then resource conflicts are likely to arise. Give a simple example of a level illustrating this.

b) Consider the following idea. To solve a level, each agent first computes its own plan for achieving its own goals—not considering that the other agents will act concurrently. More precisely, each agent A_i produces its own plan $a_1^i, a_2^i, \ldots, a_{m_i}^i$. We now try to do *plan merging* on these individual plans. We first consider what we will call *simple plan merging*: At any step in the joint plan, each agent can choose between executing the next action of its individual plan or do a *NoOp* (do nothing). Simple plan merging doesn't allow agents to change the actions of their individual plans in case of conflicts, but only allows them to attempt to solve conflicts by interleaving their action executions. Provide two example levels, one in which a conflict between individual plans can be solved by simple plan merging, and another level in which it cannot.

c) We want to show that the problem of simple plan merging can be solved using ideas from the POP algorithm. The idea is to combine the individual plans into a partial-order plan consisting of one path for each of the individual agents' plans. In the case of 3 agents all having plans of length 2, it would look like this:



Describe, in pseudocode, an algorithm relying on the POP algorithm that works as follows. As input it takes a set of individual plans $a_1^i, a_2^i, \ldots, a_{m_i}^i$, one for each agent A_i. As output it produces a simple merged plan, if such a plan exists, and otherwise it returns *failure*.

d) Consider how your algorithm would work on a few distinct levels with conflicting individual plans. You don't have to write down the details, just make sure you understand how the algorithm works and how it solves the conflicts.

e) Your algorithm can also be used for the multi-agent blocks world problem considered on today's slides. Show that your algorithm produces the same merged plan as on the slides to the problem where agent 1 has a plan to achieve $On(A, B)$ and agent 2 has a plan to achieve $On(B, C)$.

f) Argue why a merged plan outputted by your algorithm will always be conflict-free. *Hint*: This relates to Exercise 2(f) in the Theory Assignment.

g) Not all individual plans can be merged by simple plan merging. Question (b) illustrates this. However, the POP algorithm can still be helpful in this case, to produce a repaired joint plan. Sketch how.