Knowledge, Reasonging, and Planning

This assignment is **individual**.

Deadline: The deadline is **March 6 2019, 8pm** and it is strict.

Submission: Please, submit your solutions in Canvas. Submit your solution to the first exercise as a single .zip file under **A3.PP PDDL zip submission** in Assignments. Submit your solutions to the rest of the exercises as a single .pdf file under **A3.PP pdf submission**. Scanned handwriting is OK as long as it is clearly readable.

Grade E: You need to receive 45/50 points for the E-level exercises, i.e. from 1.1, 2.1, 2.2, 3.1, 3.2, and 3.3.

Grade D: You need to receive 72/80 points for the E-level and D-level exercises, i.e. from 1.1, 1.2, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, and 3.4.

Grade C: You need to receive 108/120 points for the E-level, D-level, and C-level exercises, i.e. from 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, 2.4, 2.5, 3.1, 3.2, 3.3, 3.4, 3.5, and 4.1.

Grade B: You need to receive 144/160 points for all the exercises.

1 Maze Runner

In this exercise, we are going to look into encoding the following domain using PDDL:

A person is trapped in a maze. After long time solving all the riddles the maze put in her way, the person finally arrives to the entrance of the very last puzzle, such as the one depicted in Figure 1.1), where she finds a hall with 6 amulets and two rooms.

A person (from now on called runner) is able to do the following:

- **Move**: She is able to move from one position to another if they are connected.
- **Pick up:** She is able to pick up an amulet.
- **Drop off:** She is able to drop off the amulet.

The goal of the runner is obviously to escape the maze. The requirements for this will be tighter for each grade level.

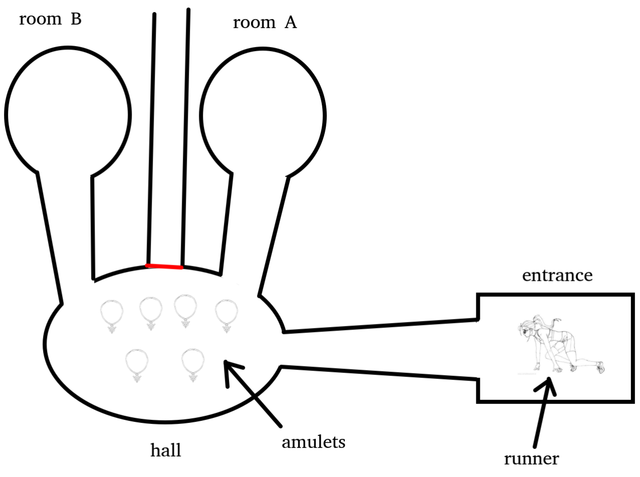

Figure 1.1: Illustration of the problem to solve encoded in the different versions of maze-problem.pddl.

1.1 E-level 10 points

Now, in order to escape, the runner must ensure both rooms are activated. We say a room is activated every time an amulet has been dropped off there (it does not need to stay there for this, it can be picked up again, but the room stays activated).

Download the package **MazeRunner.zip** from Canvas. Your task is to complete the file **maze-domain-E.pddl** to formalize this domain. The syntax used in the file is a standardized syntax used in state-of-the-art PDDL solvers, such as in this on-line editor and solver [2]. There are numerous examples of problems encoded in this syntax under the Import tab in this tool. There are also numerous tutorials on this syntax, for instance this one [1]. The relevant tab to explore there is PDDL Background.

Note that the file maze-domain-E.pddl will only contain the definition of the *domain*. *Problem instances* including the definition of objects in the world, the initial state and the goal specification are given in the file maze-problem-E.pddl. This instance encodes the problem depiced in Figure 1.1.

Your task is to complete only the code for each of the three actions, which involves writing the parameters, the precondition and the effect. Note that each action comes with a comment that gives more details that the brief domain introduction above. All the predicates you are allowed to use are already given in the file. You will not need to define any requirements or functions.

You can use the above mentioned on-line editor and solver [2] to see whether your domain definition allows to find a solution to problem-i.pddl.

Alternatively, we provided **solve.py** that calls a PDDL solver. Its usage is the following:

```
$ cd <path-to-folder>
$ python solve.py <path-to-domain> <path-to-problem> <path-to-solution> [-v]
```

If the *-v* option is selected then the path obtained and its cost will be shown in your terminal. NOTE: The **solve.py** shall be executed with Python 2.7.

1.2 D-level 10 points

Now, each of the rooms not only need to be activated but also needs to have 3 or more amulets inside of it in order to allow the runner to escape the maze. As you see, the file maze-problem-D.pddl has already been extended with the declaration of numerical fluents for the elements of the rooms and the hall.

Your second task is to complete the files maze-domain-D.pddl and maze-problem-D.pddl to be able to handle the new requirement from the maze puzzle. This basically consist on:

- In the domain: extending the actions of pick up and drop off, write the functions needed for this extension.
- In the problem: extend the goal to encode also these new part of the puzzle.

Note: Please, note that the provided solver is not able to handle well numerical fluents and that you will not be able to solve the problem using solve.py anymore.

The tutorial [1] we already mentioned above is a good starting point to learn how to add numeric expresions to your PDDL (or, more precisely now PDDL2.1) code. The relevant tabs to explore there are Numeric Expressions, Conditions, and Effects, and also Plan Metrics.

A few hints:

- The existence of paths between each location is encoded by the predicate `connected`.
- Since you will need to work with numeric fluents, (:requirements :fluents) will be added to the domain definition maze-domain.pddl. You should not need any other requirements.

How to submit: Under the assignment A3.PP.PDDL, wrap into .zip that contains maze-problem-g.pddl and maze-domain-g.pddl with g the maximum grade level you attempted. Since each exercise is modular we can see if you did the other things correctly from there. The grade will be shown as the number of points you reached, i.e. x/20. If you attempted E-level only, then you will reach max. 10/20 points.

References

[1] A PDDL 2.1 tutorial, `https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume20/fox03a-html/JAIRpddl.html`, Accessed: 2018-09-26

[2] An online PDDL editor and solver, `http://editor.planning.domains/`, Accessed: 2018-09-26

2 Student Einstein Puzzle

John, Mary, Carol and Sofie are four hard-working students who, besides attending university (either KTH or SU), also work as interns in renowned companies (either W or Z). To commute from university to work every day, each one of them chooses between cycling, walking or taking the subway. Furthermore, we know the following:

S1: John rides a bicycle to work.

S2: Carol rides a bicycle to work.

S3: Carol works at W.

S4: Mary takes the subway to work.

S5: Mary and Carol work together.

S6: Mary and Carol do not study together.

S7: Everyone that cycles to work studies at KTH.

S8: Everyone that studies at KTH is a classmate of Sofie.

S9: Every classmate of Carol that is not Carol herself, works at Z.

S10: Everyone that takes the subway and works at W, studies at SU.

S11: Every coworker of John prefers walking to work.

2.1 E-level 10 points

Write down the first-order logic representations of the above 11 premises, suitable for use with Generalized Modus Ponens. Use constants

$$John, Mary, Carol, Sofie, Bike, Walk, Subway, KTH, SU, W, Z,$$

functions

$$Commute(x), Company(x), University(x),$$

and predicates

$$Coworkers(x, y), Classmates(x, y),$$

meaning that x and y are coworkers, and x and y are classmates, respectively. A hint: you will not need existential quantifier.

2.2 E-level 5 points

Infer sentence $Classmates(John, Sofie)$ using a sequence of Generalized Modus Ponens from the knowledge base including only the above 11 premises S1 - S11. Recall that Generalized Modus Ponens is an inference rule

$$\frac{p_1', p_2', \ldots, p_n',\ (p_1 \wedge p_2 \wedge \ldots p_n \Rightarrow q)}{SUBST(\theta, q)},$$

where θ is a substitution such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, and p_i, p_i' are atomic sentences, for all i.

For instance, consider a different problem where the two premises are X1: $House(Bob) = Red$, and X2: $House(x) = Red \Rightarrow Fear(x) = Spiders$. From this, we derive that p_1' is $House(Bob) = Red$, p_1 is $House(x) = Red$, q is $Fear(x) = Spiders$, and $\theta = \{x/Bob\}$. Notice that $SUBST(\theta, p_1') = SUBST(\theta, p_1)$, which is $House(Bob) = Red$, and $SUBST(\theta, q)$ is $Fear(Bob) = Spiders$.

Write down each application of Generalized Modus Ponens following this form, including θ:

$$\frac{X1 : House(Bob) = Red \quad X2 : (House(x) = Red \Rightarrow Fear(x) = Spiders)}{X3 : Fear(Bob) = Spiders},\ \theta = \{x/Bob\}.$$

2.3 D-level 10 points

Convert each premise S1-S11 to the conjunctive normal form (CNF). In other words, convert each of them to a clause, where clause is a disjuction of literals. Following the notation from the slides and the course book, literals are atomic sentences or their negations. Denote the clauses you obtain C1 - C11.

2.4 C-level 15 points

Consider the knowledge base containing C1 - C11 you just obtained and four additional sentences in CNF:

C12: $\neg Coworkers(x, y) \vee \neg Company(x) = z \vee Company(y) = z.$

C13: $\neg Company(x) = z \vee \neg Company(y) = z \vee Coworkers(x, y).$

C14: $\neg Coworkers(x, y) \vee Coworkers(y, x).$

C15: $\neg Classmates(x, y) \vee Classmates(y, x).$

C16: $\neg Classmates(x, y) \vee \neg University(x) = z \vee University(y) = z.$

C17: $Classmates(x, y) \vee \neg University(x) = z \vee \neg University(y) = z.$

- Express sentences C12 - C15 in good, natural English. No x's and no y's.
- Infer which university each of people study at, how they commute, and which company they work for using Resolution from the knowledge base including only the premises C1 - C17 **blue**and the following additional sentences: $C18 : Carol = Carol$, $C19 : Carol \neq John$, $C20 : Carol \neq Mary$, and $C21 : Carol \neq Sofie$. Recall that resolution rule is

  $$\frac{\ell_1 \vee \ldots \vee \ell_k, \; m_1 \vee \ldots \; \vee m_n}{SUBST(\theta, \ell_1 \vee \ldots \ell_{i-1} \vee \ell_{i+1} \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)},$$

  where $UNIFY(\ell_i, \neg m_j) = \theta$, which means that θ satisfies $SUBST(\theta, l_i) = SUBST(\theta, \neg m_j)$.

  For instance, consider a different problem where two clauses are $X1 : House(Bob) = Red$ and $X2 : \neg House(x) = Red \vee Fear(x) = Spiders$. ℓ_1 is $House(Bob) = Red$, m_1 is $\neg House(x) = Red$, m_2 is $Fear(x) = Spiders$, and $UNIFY(\ell_1, \neg m_1) = \theta = \{x/Bob\}$. The resolvent clause is $Fear(Bob) = Spiders$. Write down each resolution following this form, including the unifier θ:

  $$\frac{X1 : House(Bob) = Red, \quad X2 : \neg House(x) = Red \vee Fear(x) = Spider}{X3 : Fear(Bob) = Spiders}, \text{ where } \theta = \{x/Bob\}. \tag{2.1}$$

2.5 C-level 5 points

Construct an example of two *grounded* clauses that can be resolved together in two different ways giving two different outcomes. Show the two different resolutions.

3 Florence + The Machine at Ericsson Globe

Florence + The Machine play a concert at Ericsson Globe at 11th March 2019. You were lucky and a friend living at Mariatorget managed to buy a ticket for you. Unfortunately, you already know that the tunnelbana system will be under construction on that day. Therefore, you want to plan well ahead which "Ersättningsbussar" you need to take in order to get from KTH to Globen. The tunnelbana lines are replaced by corresponding bus lines. You found information on TheRedBus which goes from KTH to Slussen and from Slussen to Mariatorget. TheGreenBus brings you from Slussen to Gullmarsplan and also from Gullmarsplan to Globen. TheGreenBus always passes Gullmarsplan so if you do not leave the bus you will be able to return to Gullmarsplan. Before the concert, you first have to meet your friend to get the ticket. The story can be formalized in PDDL:

Init (*Travel(**T**ekniska **h**ögskolan, **Sl**ussen, The**R**ed**B**us)* ∧
*Travel(**Sl**ussen, **M**aria**t**orget, The**R**ed**B**us)* ∧
*Travel(**Sl**ussen,**G**ullmars**p**lan, The**G**reen**B**us)* ∧
*Travel(**G**ullmars**p**lan, **G**ullmars**p**lan, The**G**reen**B**us)* ∧
*Travel(**G**ullmars**p**lan, **Gl**oben, The**G**reen**B**us)* ∧
*BusAt(The**R**ed**B**us, **T**ekniska **h**ögskolan)* ∧ *BusAt(The**G**reen**B**us, **Sl**ussen)* ∧
*YouAt(**T**ekniska **h**ögskolan)* ∧ *TicketAtFriend(**M**aria**t**orget)*)
Goal (*YouAt(**Gl**oben)* ∧ *YouHaveTicket*)

$$\mathit{Action}\ \big(\mathit{GoForth}(\mathit{from}, \mathit{to})$$
$$\text{PRECOND}: \mathit{Travel}(\mathit{from}, \mathit{to}, \mathit{bus}) \wedge \mathit{YouAt}(\mathit{bus}) \wedge \mathit{BusAt}(\mathit{bus}, \mathit{from})$$
$$\text{EFFECT}: \neg \mathit{BusAt}(\mathit{bus}, \mathit{from}) \wedge \mathit{BusAt}(\mathit{bus}, \mathit{to})\big)$$
$$\mathit{Action}\ \big(\mathit{GoBack}(\mathit{from}, \mathit{to})$$
$$\text{PRECOND}: \mathit{Travel}(\mathit{to}, \mathit{from}, \mathit{bus}) \wedge \mathit{YouAt}(\mathit{bus}) \wedge \mathit{BusAt}(\mathit{bus}, \mathit{from})$$
$$\text{EFFECT}: \neg \mathit{BusAt}(\mathit{bus}, \mathit{from}) \wedge \mathit{BusAt}(\mathit{bus}, \mathit{to})\big)$$
$$\mathit{Action}\ \big(\mathit{GetOnBus}(\mathit{bus}, \mathit{station})$$
$$\text{PRECOND}: \mathit{BusAt}(\mathit{bus}, \mathit{station}) \wedge \mathit{YouAt}(\mathit{station})$$
$$\text{EFFECT}: \neg \mathit{YouAt}(\mathit{station}) \wedge \mathit{YouAt}(\mathit{bus}))$$
$$\mathit{Action}\ \big(\mathit{GetOffBus}(\mathit{bus}, \mathit{station})$$
$$\text{PRECOND}: \mathit{BusAt}(\mathit{bus}, \mathit{station}) \wedge \mathit{YouAt}(\mathit{bus})$$
$$\text{EFFECT}: \neg \mathit{YouAt}(\mathit{bus}) \wedge \mathit{YouAt}(\mathit{station})\big)$$
$$\mathit{Action}\ \big(\mathit{PickTicket}(\mathit{station})$$
$$\text{PRECOND}: \mathit{YouAt}(\mathit{station}) \wedge \mathit{TicketAtFriend}(\mathit{station})$$
$$\text{EFFECT}: \neg \mathit{TicketAtFriend}(\mathit{station}) \wedge \mathit{YouHaveTicket}\big)$$

3.1 E-level 10 points

We will look into converting the planning problem to planning in a state space. In the following, we will you abbreviations for the stations according to the letters written in bold. Draw a part of the reachable space of all states containing at least 10 different states. Start from the initial state given in Figure 3.1. Represent each state as the list of fluents (ground, functionless atoms) that are true therein. In classical planning, we make the *closed-world assumption* (database semantics) meaning that any fluent not mentioned in the state is false. However, because $\mathit{Travel}(\mathit{Th}, \mathit{Sl}, \mathit{RB})$, $\mathit{Travel}(\mathit{Sl}, \mathit{Mt}, \mathit{RB})$, $\mathit{Travel}(\mathit{Sl}, \mathit{Gp}, \mathit{GB})$, $\mathit{Travel}(\mathit{Gp}, \mathit{Gp}, \mathit{GB})$ and $\mathit{Travel}(\mathit{Gp}, \mathit{Gl}, \mathit{GB})$ are always true, for simplicity of presentation, you do not have to list them explicitly as labels in each state. Label each transition in the state space with the corresponding ground action.

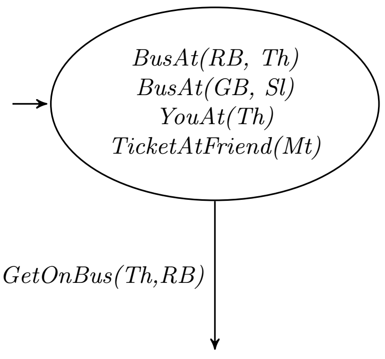

Figure 3.1: A part of the state space.

3.2 E-level 5 points

Answer the following questions and *provide the motivation* for your answers:

- How many different plans (sequences of actions) that lead to the satisfaction of the goal are there?
- Assume that TheGreenBus does not go to Gullmarsplan over and over, that from Gullmarsplan you can only travel Globen or Slussen. In other words, the initial state changed as follows:

  $$\textit{Init}\ \big(\textit{Travel(Th, Sl, RB)} \wedge \textit{Travel(Sl, Mt, RB)} \wedge \textit{Travel(Sl, Gp, GB)} \wedge \textit{Travel(Gp, Gl, GB)} \wedge \\ \textit{BusAt(RB, Th)} \wedge \textit{BusAt(GB, Sl)} \wedge \textit{YouAt(Th)} \wedge \textit{TicketAtFriend(Mt)}\big)$$

  How many different plans (sequences of actions) that lead to the satisfaction of the goal are there now?
- If there are multiple plans, what is the optimal plan in terms of the number of actions? Give the plan as a sequence of ground actions.

3.3 E-level 10 points

Consider a variant of the bus travel problem, where the busses information system does not work and the substitutional stations do not have any station names on them. You do not know whether you are at Tekniska Högskolan, Slussen, Mariatorget, Gullmarsplan or Globen. You know however that you are not on the bus. This means that the environment is partially observable, we are facing sensorless planning, and we need to work with the belief-state space instead. Instead of representing a belief state as an explicit enumeration of the set of states the world can be in, we can use use logical formulas. We make the *open-world assumption* meaning that representation of a belief state can contain both positive and negative fluents, and if a fluent does not appear in the belief state, the fluent is unknown.

Draw the space of all belief states that are reachable from the initial one, assuming the same action schema as above and that predicates $Travel$, $BusAt$, and $TicketAtFriend$ have the same truth value as in the initial state in the schema above. The (partially) unknown is $YouAt$. Similarly as above, for simplicity of the presentation, you do not have to explicitly list the fluents that are either always true or always false. Label each transition in the belief state space with the corresponding ground action. Notice, that an action can be applied in a belief state only if its preconditions are satisfied by every state in that belief state.

TIP: We say that you should label transitions with ground actions. This means that both variables from and to in GoForth(from,to) have to be substituted with constants. If you do that, can you actually make sure that the precondition of that action is satisfied in every single state in your belief state?

Answer the following questions and *provide the motivation* for your answers:

- Give 3 examples of fluents that are always true and 3 examples of fluents that are always false in the given environment.
- How many actual physical states does the initial belief state contain?
- How many different plans that lead to the satisfaction of the goal are there?
- If there are multiple plans, what is the optimal plan in terms of the number of actions? Give the plan as a sequence of ground actions.

3.4 D-level 10 points

Consider now that you can travel to anywhere from anywhere and we only have one bus line in a partially observable environment. Initially, you know only what is written down in the initial state below, i.e., you for instance cannot observe whether you are on a bus, and whether you have a ticket. The alternative schema is defined as follows:

$$
\begin{aligned}
&\mathit{Init}\ \big(\mathit{YouAt(Th)} \land\ \mathit{TicketAtFriend(Mt)}\big)\\
&\mathit{Goal}\ \big(\mathit{YouAt(Gl)} \land\ \mathit{YouHaveTicket}\big)\\
&\mathit{Action}\ \big(\mathit{GoToTh}\\
&\qquad \text{PRECOND} : \neg\, \mathit{YouAt}(\mathit{Th}) \land\ \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \mathit{YouAt}(\mathit{Th}) \land \neg\, \mathit{YouAt}(\mathit{Mt}) \land \neg\, \mathit{YouAt}(\mathit{Gl}) \land \neg\, \mathit{YouAt}(\mathit{Gp}) \land \neg\, \mathit{YouAt}(\mathit{Sl})\big)\\
&\mathit{Action}\ \big(\mathit{GoToSl}\\
&\qquad \text{PRECOND} : \neg\, \mathit{YouAt}(\mathit{Sl}) \land\ \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \mathit{YouAt}(\mathit{Sl}) \land \neg\, \mathit{YouAt}(\mathit{Mt}) \land \neg\, \mathit{YouAt}(\mathit{Gl}) \land \neg\, \mathit{YouAt}(\mathit{Gp}) \land \neg\, \mathit{YouAt}(\mathit{Th})\big)\\
&\mathit{Action}\ \big(\mathit{GoToGl}\\
&\qquad \text{PRECOND} : \neg\, \mathit{YouAt}(\mathit{Gl}) \land\ \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \mathit{YouAt}(\mathit{Gl}) \land \neg\, \mathit{YouAt}(\mathit{Mt}) \land \neg\, \mathit{YouAt}(\mathit{Sl}) \land \neg\, \mathit{YouAt}(\mathit{Gp}) \land \neg\, \mathit{YouAt}(\mathit{Th})\big)\\
&\mathit{Action}\ \big(\mathit{GoToGp}\\
&\qquad \text{PRECOND} : \neg\, \mathit{YouAt}(\mathit{Gp}) \land\ \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \mathit{YouAt}(\mathit{Gp}) \land \neg\, \mathit{YouAt}(\mathit{Mt}) \land \neg\, \mathit{YouAt}(\mathit{Gl}) \land \neg\, \mathit{YouAt}(\mathit{Sl}) \land \neg\, \mathit{YouAt}(\mathit{Th})\big)\\
&\mathit{Action}\ \big(\mathit{GoToMt}\\
&\qquad \text{PRECOND} : \neg\, \mathit{YouAt}(\mathit{Mt}) \land\ \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \mathit{YouAt}(\mathit{Mt}) \land \neg\, \mathit{YouAt}(\mathit{Sl}) \land \neg\, \mathit{YouAt}(\mathit{Gl}) \land \neg\, \mathit{YouAt}(\mathit{Gp}) \land \neg\, \mathit{YouAt}(\mathit{Th})\big)\\
&\mathit{Action}\ \big(\mathit{GetOnBus}(\mathit{station})\\
&\qquad \text{PRECOND} : \mathit{YouAt}(\mathit{station})\\
&\qquad \text{EFFECT} : \neg\, \mathit{YouAt}(\mathit{station}) \land\ \mathit{YouOnBus})\\
&\mathit{Action}\ \big(\mathit{GetOffBus}(\mathit{station})\\
&\qquad \text{PRECOND} : \mathit{YouOnBus}\\
&\qquad \text{EFFECT} : \neg\, \mathit{YouOnBus} \land\ \mathit{YouAt}(\mathit{station})\big)\\
&\mathit{Action}\ \big(\mathit{PickTicket}(\mathit{station})\\
&\qquad \text{PRECOND} : \mathit{YouAt}(\mathit{station}) \land\ \mathit{TicketAtFriend}(\mathit{station})\\
&\qquad \text{EFFECT} : \neg\, \mathit{TicketAtFriend}(\mathit{station}) \land\ \mathit{YouHaveTicket}\big)
\end{aligned}
$$

Answer the questions from Sec. 3.2 and Sec. 3.3 and *motivate your answers*, i.e.

- Draw a part of the reachable belief state space containing at least 5 states and a draw all their outgoing edges labeled with ground actions even if they lead outside your 5 states (in such case, just make them lead to three dots "..." to indicate it is beyond the depicted 5 states). Again, for simplicity of presentation, you do not have to explicitly list the fluents that are either always true or always false.
- How many actual physical states does the initial belief state contain?
- How many different plans that lead to the satisfaction of the goal are there?
- If there are multiple plans, what is the optimal plan in terms of the number of actions? Give the plan as a sequence of ground actions.

3.5 C-level 10 points

Consider now yet another variant of the original bus travel problem, where your friend forgot to tell you at which of the stations she lives. However, when you leave the bus your friend promised to wave at you so that you could spot her directly. The problem is not sensorless anymore. Formally, we add the following to the domain description:

$$\begin{aligned}&\mathit{Percept}\ \big(\mathit{TicketAtFriend}(at)\\&\qquad\qquad \text{PRECOND}: \mathit{YouAt}(at)\big)\end{aligned}$$

- Draw a connected subgraph of the belief state space containing at least 10 different belief states assuming that you initially do not know where your friend lives. Use the action schemas from the beginning of Sec. 3 and again, for simplicity of presentation, you do not have to explicitly list the fluents that are either always true or always false. Each transition in your belief state space should be labeled either with ground action or with perceived fluent in case *Percept* was made. For instance, from the belief state, where you are at a station, according to the domain specification, you can either get on a bus or perceive whether your friend with the ticket lives at this station. An example of the transitions labels from this state are given in Fig. 3.2.

  TIP: There are two important assumptions that we make in this exercise: your friend lives somewhere and there is only one friend that has the ticket. Now practically that means that belief states that have for instance TicketAtFriend(Gp) true and TicketAtFriend(Sl) true are not feasible. It also means that if Percept returns that TicketAtFriend(Gp) is true, you also know that TicketAtFriend(Sl) is false. It also means that once TicketAtFriend(Gp) is false, making Percept will always return that it is false. So this way, when drawing only the feasible states, you can prune the belief state space significantly.

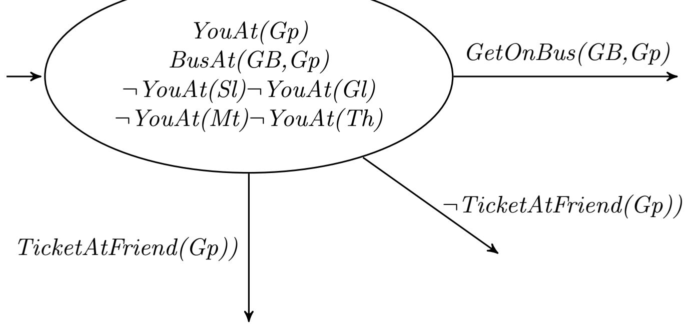

Figure 3.2: Possible transitions from one of the belief states.

- You may notice that there does not exist a sequence of actions that leads to satisfying the goal. However, there exists a *contingent plan* doing so. Find such a contingent plan and represent it as a sequence of actions and if-then-else structures. An example of a contingent plan (that however does not satisfy the goal) is

```
[ $GetOffBus(GB, Gp)$,
 **if** $TicketAtFriend(Gp)$
   **then** [ $PickTicket(Gp)$, $GetOnBus(GB, Gp)$ ]
   **else** [ $GetOnBus(GB, Gp)$, $GetOffBus(GB, Gp)$, **if** $TicketAtFriend(Gp)$
                                   **then** [ $PickTicket(Gp)$, $GetOnBus(GB, Gp)$ ]
                                   **else** [$NoOp$]]].
```

  NoOp is used to denote that no action is going to be taken.

  TIP: Note that with the two assumptions – that there is a friend with a ticket and that there is only one friend with a ticket in our world – you will be able to find a contingent plan, but without assuming that the ticket is somewhere, you won't.

3.6 B-level 15 points

Assume an arbitrary PDDL planning domain, not necessarily the one described before. We will examine two general cases. The first case is fully observable, meaning that for each fluent we always know whether it is true or false. The second case is sensorless, meaning that at any moment, the value of each fluent can be true, false, or unknown. Suppose that in this second case, in the initial state, the value of some of the fluents are unknown.

- How does the size of the state space in the first fully observable case compare in general to the size of the belief state space in the second sensorless case? Is the state space always larger, always smaller or sometimes larger in size and sometimes smaller than the belief state space? If always larger or always smaller, give a proof. Otherwise, give examples of domains and initial states, for which the state space is smaller, respectively larger than the belief state space.

- Compare the size of the representation of the inital state in the first fully observable case and the initial belief state in the second sensorless case. Count the size of the representation as the number of fluents listed in the state and assume that the fluents, whose value is always true or always false are not explicitly listed. Similarly as above, if the state is always larger in size or always smaller than the belief state, give a proof. Otherwise, give examples of domains and initial states, for which the state is smaller, respectively larger than the belief state space.

4 Sweet Cup

The following PDDL describes a robot in a kitchen that can fill it's teaspoon with sugar, pour sugar in the cups, and stir sugar in the cups. The cups are already filled with tea. The teaspoon gets wet if used for stirring and cannot be filled with sugar when wet.

$$
\begin{aligned}
&\quad\ \ \mathit{Init}\ \big(\mathit{Empty}(\mathit{Teaspoon}) \land \mathit{Dry}(\mathit{Teaspoon}) \land \mathit{Cup}(\mathit{Cup}1) \land \mathit{Cup}(\mathit{Cup}2)\big)\\
&\quad\ \ \mathit{Goal}\ \big(\mathit{Sweet}(\mathit{Cup1}) \ \land\ \mathit{Sweet}(\mathit{Cup2})\big)\\
&\mathit{Action}\ \big(\mathit{FillTeaspoon}()\\
&\qquad\quad \text{PRECOND :}\mathit{Empty}(\mathit{Teaspoon}) \land \mathit{Dry}(\mathit{Teaspoon})\\
&\qquad\quad \text{EFFECT :}\neg \mathit{Empty}(\mathit{Teaspoon}) \land \mathit{HasSugar}(\mathit{Teaspoon})\big)\\
&\mathit{Action}\ \big(\mathit{PourSugar}(\mathit{cup})\\
&\qquad\quad \text{PRECOND :}\mathit{HasSugar}(\mathit{Teaspoon}) \land \mathit{Cup}(\mathit{cup})\\
&\qquad\quad \text{EFFECT :}\neg \mathit{HasSugar}(\mathit{Teaspoon}) \land \mathit{Empty}(\mathit{Teaspoon}) \land \mathit{HasSugar}(\mathit{cup})\big)\\
&\mathit{Action}\ \big(\mathit{Stir}(\mathit{cup})\\
&\qquad\quad \text{PRECOND :}\mathit{Cup}(\mathit{cup}) \land \mathit{HasSugar}(\mathit{cup}) \land \mathit{Empty}(\mathit{Teaspoon})\\
&\qquad\quad \text{EFFECT :}\neg \mathit{Dry}(\mathit{Teaspoon}) \land \mathit{Sweet}(\mathit{cup}))\big)
\end{aligned}
$$

4.1 C-level 10 points

We will look into and compare different domain-independent heuristics guiding forward search when the PDDL planning problem is translated into a state space search problem. Denote the state space (S, T), where S is the set of states and T is a set of transitions (edges) between the states. Consider:

H1 Ignoring all preconditions of all actions in the action scheme.

H2 Ignoring all delete lists.

Each of the heuristics defines a new PDDL planning domain and a new state space (S', T'). The minimal number of transitions between a state $s \in S'$ in this state space and a goal state is $h(s)$, and h can be used as a heuristics to guide search in (S, T). This means that $H1$ and $H2$ define heuristics h_1 and h_2 for search in the state space (S, T).

Answer the following questions and *provide motivation for each of the answers*:

- What is the value of h_1 for the initial state?
- What is the value of h_2 for the initial state?
- Compare heuristics h_1 and h_2 in general (i.e. not only for this problem.). Does one of them dominate the other?

4.2 B-level 20 points

Overview paper [1] and focus on understanding the following terms:

- *goal distance,*

- *dead end,*
- *recognized dead end,*
- *unrecognized dead end.*

Answer the following questions for above robot in the grid problem and *provide motivation for each of the answers*:.

- Does there exist a reachable recognized dead end for H1?
- Does there exist a reachable recognized dead end for H2?
- Does there exist a reachable unrecognized dead end for H1?
- Does there exist a reachable unrecognized dead end for H2?

4.3 B-level 5 points

Suppose that you have a heuristics with the property that causes reachable unrecognized dead ends. Will that prevent finding a solution? Why?

References

[1] Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.