Knowledge, Reasonging, and Planning

This assignment is **individual**.

Deadline: The deadline is **March 6, 2020, 11:59pm** and it is strict.

Submission: Please, submit your solutions in Canvas. Submit your solution to the first exercise 1.1 as a single text file file under **A3.PP.1.PDDL submission** in Assignments. Submit your solution to exercise 2.7 as a single text file under **A3.PP.2.SWISH submission** in Assignments (only if you aim for B or higher). 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. Make sure that the .pdf is clearly signed and organized.

Grade E: You need to receive 31/35 points for the E-level exercises, i.e. from 1.1, 2.1 - 2.3, and 3.1.

Grade D: You need to receive 49/55 points for the E-level and D-level exercises, i.e. from 1.1 - 1.2, 2.1- 2.4, 3.1 - 3.3.

Grade C: You need to receive 76/85 points for the E-level, D-level, and C-level exercises, i.e. from 1.1 - 1.3, 2.1 - 2.6, 3.1 - 3.4.

Grade B: You need to receive 108/120 points for the E-level, D-level, C-level, and B-level exercises, i.e. from 1.1 - 1.3, 2.1 - 2.7, 3.1 - 3.5, 4.1 - 4.2.

Grade A: You need to receive 135/150 points for all the exercises.

1 Assembling Task

In this exercise, we are going to look into encoding a planning domain using PDDL:

Two friends are assembling a piece of furniture. Each one can do the following:

- **Grab tool:** The person is able to grab a tool
- **Drop tool:** The person is able to drop a tool
- **Grab wood:** The person is able to grab a wood
- **Paint:** The person is able to paint a wood
- **Nail**: The person is able to hit a nail on a wood
- **Approve:** The person is able to approve the assembly

1.1 E-level 10 points

Download the package **Assembling.zip** from Canvas. Your task is to complete the file **assembling-domain.pddl** to formalize the planning 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 assembling-domain.pddl will only contain the definition of the *domain*. *The problem instances* including the definition of objects in the world, the initial state and the goal specification are given in separate files. You can find three examples of problem instances for the domain in this exercise in the **Assembling-domain-problems** subfolder named **problem-*i*.pddl**, where i is the number of the problem.

Your task is to complete only the code for each of the five 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. It should.

How to submit: Under the assignment A3.PP.1.PDDL, upload assembling-domain.pddl. Make sure that your submitted file does not contain syntax errors.

1.2 D-level 5 points

In the **Assembling-domain-problems** subfolder, you can find a problem named **problem-with-no-solution**. This problem should not be solvable in the domain you have defined so far. Define a new action drop tool in the domain that will allow the problem to get solved.

How to submit: Under the assignment A3.PP.1.PDDL, upload assembling-domain.pddl. Make sure that your submitted file does not contain syntax errors. You only submit one file that contains your solution to E and D-level exercises.

1.3 C-level 10 points

A PDDL planning domain together with a problem definition can be translated into a state space (as you will do in Exercise 3). A solution to the planning problem can then be found in this state space via forward search. It is important to define a good heuristic function to guide the search and make it efficient.

We will look into 2 different domain-independent heuristics guiding the forward search in the assembling task. For both of them, the value of the heuristic function for a state is the cost (i.e. the number of steps – actions) of the solution from that state in some *relaxed problem*. The two relaxed problems change the domain by:

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

H2 Ignoring all delete lists.

What is the value of the heuristic functions H1 and H2 for the initial state of problem-i.pddl, for $i = 1, 2, 3$? Motivate your answer and explain how you calculated it.

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 Company

Rose, Lily, Frank and Harry are four members of a famous company. They occupy different positions within the company: an intern, a manager, a director, and a CEO. Each of them reports to the its immediate superior, i.e. the intern reports to the manager, the manager reports to the director, and the director reports to the CEO. Each of them sits on a different floor of a skyscraper: three, five, nine and twelve. We can use the following constants

$$\mathit{Rose}, \mathit{Lily}, \mathit{Frank}, \mathit{Harry}, \mathit{3}, \mathit{5}, \mathit{9}, \mathit{12}, \mathit{intern}, \mathit{manager}, \mathit{director}, \mathit{CEO};$$

functions

$$\mathit{Floor}(x), \mathit{Position}(x),$$

and predicate

$$\mathit{ReportsTo}(x, y)$$

meaning that x reports to y. We know the following facts about them.

S1: $\mathit{ReportsTo}(x, y) \wedge \mathit{Floor}(y) = 3 \Rightarrow \mathit{ReportsTo}(\mathit{Rose}, x)$

S2: $\mathit{Position}(\mathit{Frank}) = \mathit{manager}$

S3: $\mathit{ReportsTo}(x, y) \wedge \mathit{Floor}(x) = 12 \Rightarrow \mathit{Floor}(y) = 3$

S4: $\mathit{Floor}(\mathit{Harry}) = 9$

S5: $\mathit{ReportsTo}(\mathit{Frank}, \mathit{Lily})$

S6: $\mathit{Floor}(x) = 3 \Rightarrow \mathit{ReportsTo}(x, \mathit{Harry})$

S7: $\mathit{ReportsTo}(x, \mathit{Lily}) \Rightarrow \mathit{Floor}(x) = 12$

2.1 E-level 5 points

Infer sentence $\mathit{Floor}(\mathit{Frank}) = \mathit{12}$ using a sequence of Generalized Modus Ponens from the knowledge base including only the above 7 sentences S1 - S7. 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 sentences are X1: $\mathit{House}(\mathit{Bob}) = \mathit{Red}$, and X3: $\mathit{House}(x) = \mathit{Red} \Rightarrow \mathit{Fear}(x) = \mathit{Spiders}$. From this, we derive that p'_1 is $\mathit{House}(\mathit{Bob}) = \mathit{Red}$, p_1 is $\mathit{House}(x) = \mathit{Red}$, q is $\mathit{Fear}(x) = \mathit{Spiders}$, and $\theta = \{x/\mathit{Bob}\}$. Notice that $SUBST(\theta, p'_1) = SUBST(\theta, p_1)$, which is $\mathit{House}(\mathit{Bob}) = \mathit{Red}$, and $SUBST(\theta, q)$ is $\mathit{Fear}(\mathit{Bob}) = \mathit{Spiders}$.

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

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

2.2 E-level 5 points

Translate the sentences S1-S7 to CNF sentences C1-C7.

2.3 E-level 5 points

Infer $ReportsTo(Rose, Frank)$ using Resolution from the knowledge base including only the sentences C1 - C7 and the sentences you inferred in Exercise 2.1.

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:

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

2.4 D-level 5 points

Consider the knowledge base containing S1 - S7 and one additional sentence in CNF:

S8: $\neg Position(x) = Manager \vee \neg ReportsTo(x, y) \vee Position(y) = Director.$

Express sentences (S1-S8) in good, natural English. No x's and no y's.

2.5 C-level 5 points

Construct an example of two grounded clauses that can be resolved together in two different ways resulting in $A \vee \neg A$ and $B \vee \neg B$. Show the two different resolutions.

2.6 C-level 5 points

Construct an example of a knowledge base with four *different* grounded clauses $C1, C2, C3, C4$, such that resolving $C1$ with $C2$ gives the very same result as resolving $C3$ with $C4$.

2.7 B-level 10 points

Consider now the knowledge base containing S1 - S8 and three additional sentences in CNF

S9: $\neg Position(x) = Manager \vee \neg ReportsTo(y, x) \vee Position(y) = Intern.$

S10: $\neg Position(x) = Director \vee \neg ReportsTo(x, y) \vee Position(y) = CEO.$

S11: $Floor(Rose) = 5.$

Use the online tool *SWISH* ($https://swish.swi-prolog.org$) to solve this puzzle. You can define a list of company members $company(Com)$ with each employee having three attributes: $person(name, position, floor)$. For syntax, you can look at a similar problem, the Einstein's riddle, in the examples section: $https://swish.swi-prolog.org/example/houses\_puzzle.pl$.

For instance, to add sentence S2, you can write
$member(person(Frank, manager, \_), Com)$ which means that there is an employee in the list Com that has the name "Frank" and is "manager". For the reports relation you can define it as follows:

$$
\begin{aligned}
reportsto(X, Y, Ls) :- \\
&member(X, Ls), \\
&member(Y, Ls), \\
&((X = person(\_, intern, \_), Y = person(\_, manager, \_)); \\
&(X = person(\_, manager, \_), Y = person(\_, director, \_)); \\
&(X = person(\_, director, \_), Y = person(\_, CEO, \_))).
\end{aligned}
\tag{2.2}
$$

Note that ',' is used to represent conjunction and ';' is used to represent disjunction. Also note that in the definition of the function "reports", we are taking into account the initial puzzle description that states that there is exactly one person per floor.

How to submit 2.7: Upload your solution to this part of the assignment as a text file einstein.txt under assignment A3.PP.2.SWISH.

2.8 A-level 10 points

Consider the following claim: "Two grounded clauses cannot be solved together in two different ways and result in $A \vee B$ and $\neg A \vee \neg B$". Either prove that this claim holds true for any two grounded clauses, or find a counter-example. Motivate you answer properly and be rigorous.

3 Grocery shopping

3.1 E-level 10 points: state space

Let us go grocery shopping at a fancy supermarket: There is a basket to carry your shopping items. In the supermarket you further have a fridge with a MellanmjölkCarton and a FilmjölkCarton, and a freezer with a pizza. Unfrotunately, only one of each of the items is left You can open the fridge and the freezer and you can add the MellanmjölkCarton, FilmjölkCarton and PizzaBox to your basket. You can also pay for the cartons and the box. You would like to have milk and pizza.

Grocery Shopping Version 1

Init $\big($*IsIn(Fridge, MellanmjölkCarton)* $\wedge$ *IsIn(Freezer, PizzaBox)* $\wedge$ *IsIn(Fridge, FilmjölkCarton)* $\wedge$
Shelf(Fridge) $\wedge$ *Shelf(Freezer)* $\wedge$ *Contains(FilmjölkCarton, SourMilk)*,
Contains(MellanmjölkCarton, Milk) $\wedge$ *Contains(PizzaBox, Pizza)*$\big)$

Goal $\big($*Have(Milk)* $\wedge$ *Have(Pizza)* $\wedge \neg$*Have(SourMilk)*$\big)$

Action $\big($*OpenDoor*(x)
PRECOND :$\neg$*Accessible*(x) $\wedge$ *Shelf*(x)
EFFECT : *Accessible*$(x)\big)$

Action $\big($*CloseDoor*(x)
PRECOND :*Accessible*(x) $\wedge$ *Shelf*(x)
EFFECT : $\neg$*Accessible*$(x)\big)$

Action $\big($*Add*(x, y)
PRECOND :*Accessible*(x) $\wedge$ *IsIn(x,y)*
EFFECT : *InBasket*(y) $\wedge \neg$*IsIn(x,y)*$\big)$

Action $\big($*Pay*(x, y)
PRECOND :*InBasket*(x) $\wedge$ *Contains(x,y)*
EFFECT : $\neg$*InBasket*(x) $\wedge$ *Have(y)*$\big)$

3.1.1 We will look into converting the planning problem to planning in a state space. Draw a connected part of the *reachable state space* with at least 7 different states. Start from the initial state given in Figure 3.1 and draw states that can be reached from it in one or more steps. Represent each state as the *list of fluents (ground, functionless atoms) that are true therein.* However, because *Shelf*$($*Fridge*$)$, *Shelf*$($*Freezer*$)$, *Shelf*$($*Fridge*$)$ and *Contains*$($*FilmjlkCarton*, *SourMilk*$)$, *Contains*$($*MellanmjlkCarton*, *Milk*$)$, *Contains*$($*PizzaBox*, *Pizza*$)$ are always true, for simplicity of presentation, you do not have to list them explicitly as labels in each state, just as they are not listed in Figure 3.1. In classical planning, we make the *closed-world assumption* meaning that any fluent not mentioned in the state is false. Label each transition in the state space with the corresponding *ground action.*
Answer the following questions and *provide the motivation* for your answers:

3.1.2 How many different states are there in the whole reachable state space?

3.1.3 How many different goal states satisfying the goal condition are there in the reachable state space?

3.1.4 How many different plans (sequences of actions) that lead to the satisfaction of the goal condition are there?

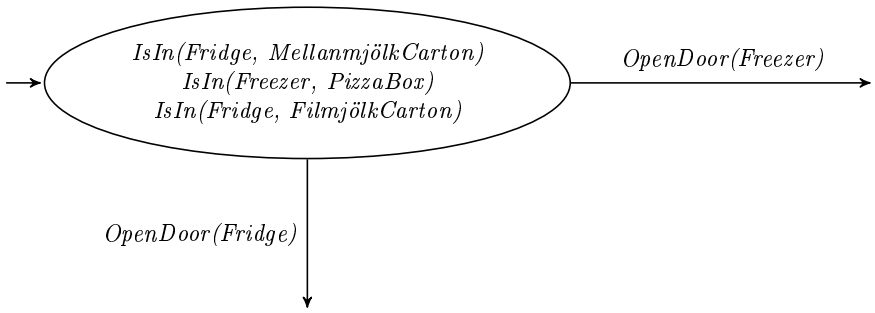

Figure 3.1: A part of the state space.

3.1.5 If there are multiple such plans, what is the optimal plan in terms of the number of actions? Give the plan as a sequence of ground actions.

3.2 D-level 5 points: alternative state space properties

Consider an alternative formalization of the problem in PDDL:

Grocery Shopping Version 2: Sustainable choices in an empty supermarket
Imagine now that you are going shopping with sustainability in mind and only want to open each door once. Your supermarket still has only one of each item you want to have left.

$$
\begin{aligned}
\mathit{Init}\ (&\mathit{IsIn(Fridge,\ MellanmjölkCarton)} \wedge \mathit{IsIn(Freezer,\ PizzaBox)} \wedge \\
&\mathit{IsIn(Fridge,\ FilmjölkCarton)} \wedge \\
&\mathit{Shelf(Fridge)} \wedge \mathit{Shelf(Freezer)} \wedge \mathit{Contains(FilmjölkCarton,\ SourMilk)} \wedge \\
&\mathit{Contains(MellanmjölkCarton,\ Milk)} \wedge \mathit{Contains(PizzaBox,\ Pizza)}) \\
\mathit{Goal}\ (&\mathit{Have(Milk)} \wedge \mathit{Have(Pizza)} \wedge \neg \mathit{Have(SourMilk)}) \\
\mathit{Action}\ (&\mathit{OpenDoor}(x) \\
&\text{PRECOND} : \neg \mathit{DoorOpened}(x) \wedge \neg \mathit{Accessible}(x) \wedge \mathit{Shelf}(x) \\
&\text{EFFECT} : \mathit{Accessible}(x) \wedge \mathit{DoorOpened}(x)) \\
\mathit{Action}\ (&\mathit{CloseDoor}(x) \\
&\text{PRECOND} : \mathit{Accessible}(x) \wedge \mathit{Shelf}(x) \\
&\text{EFFECT} : \neg \mathit{Accessible}(x)) \\
\mathit{Action}\ (&\mathit{Add}(x, y) \\
&\text{PRECOND} : \mathit{Accessible}(x) \wedge \mathit{IsIn(x,y)} \\
&\text{EFFECT} : \mathit{InBasket}(y) \wedge \neg \mathit{IsIn(x,y)}) \\
\mathit{Action}\ (&\mathit{Pay}(x, y) \\
&\text{PRECOND} : \mathit{InBasket}(x) \wedge \mathit{Contains(x,y)} \\
&\text{EFFECT} : \neg \mathit{InBasket}(x) \wedge \mathit{Have(y)})
\end{aligned}
$$

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

3.2.1 How many different states are there in the whole reachable state space?

3.2.2 How many different goal states satisfying the goal condition are there in the reachable state space?

3.2.3 How many different plans (sequences of actions) that lead to the satisfaction of the goal condition are there?

3.3 D-level 5 points: yet another alternative state space properties

Consider yet another alternative formalization of the problem in PDDL, luckily your supermarket is now fully (infinitely) stocked (change in Action *Add*):

Grocery Shopping Version 3: Do I try Filmjölk? Changed goal

$$
\begin{aligned}
\mathit{Init}\ &\big(\mathit{IsIn(Fridge,\ MellanmjölkCarton)} \wedge \mathit{IsIn(Freezer,\ PizzaBox)} \wedge \mathit{IsIn(Fridge,\ FilmjölkCarton)} \wedge \\
&\quad \mathit{Shelf(Fridge)} \wedge \mathit{Shelf(Freezer)} \wedge \mathit{Contains(FilmjölkCarton,\ SourMilk)}, \\
&\quad \mathit{Contains(MellanmjölkCarton,\ Milk)} \wedge \mathit{Contains(PizzaBox,\ Pizza)}\big) \\
\mathit{Goal}\ &\big(\mathit{Have(Milk)} \wedge \mathit{Have(Pizza)}\big) \\
\mathit{Action}\ &\big(\mathit{OpenDoor}(x) \\
&\quad \text{PRECOND} : \neg \mathit{DoorOpened}(x) \wedge \neg \mathit{Accessible}(x) \wedge \mathit{Shelf}(x) \\
&\quad \text{EFFECT} : \mathit{Accessible}(x) \wedge \mathit{DoorOpened}(x)\big) \\
\mathit{Action}\ &\big(\mathit{CloseDoor}(x) \\
&\quad \text{PRECOND} : \mathit{Accessible}(x) \wedge \mathit{Shelf}(x) \\
&\quad \text{EFFECT} : \neg \mathit{Accessible}(x)\big) \\
\mathit{Action}\ &\big(\mathit{Add}(x, y) \\
&\quad \text{PRECOND} : \mathit{Accessible}(x) \wedge \mathit{IsIn(x,y)} \\
&\quad \text{EFFECT} : \mathit{InBasket}(y)\big) \\
\mathit{Action}\ &\big(\mathit{Pay}(x, y) \\
&\quad \text{PRECOND} : \mathit{InBasket}(x) \wedge \mathit{Contains(x,y)} \\
&\quad \text{EFFECT} : \neg \mathit{InBasket}(x) \wedge \mathit{Have(y)}\big)
\end{aligned}
$$

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

3.3.1 How many different states are there in the whole reachable state space?

3.3.2 How many different goal states satisfying the goal condition are there in the reachable state space?

3.3.3 How many different plans (sequences of actions) that lead to the satisfaction of the goal condition are there?

3.4 C-level 10 points: belief state space

Consider another version of the grocery shopping problem, where there is no PizzaBox nor a freezer, and you only want to have milk. However, as you are new to Sweden and you do not know the Swedish word for milk yet you cannot observe the content of the cartons; they can contain milk, or sour milk, or something different. However, you can observe the truth values of all the other fluents. This means that the environment is partially observable, and 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 truth value of the fluent is unknown. On the other hand, every known fluent, positive or negative, appears in the belief state.

The situation can be formalized in PDDL as follows. Notice, how the description of the initial state changed now that we have the open-world assumption.
Further note, that *Contains(FilmjölkCarton, SourMilk)*, *Contains(MellanmjölkCarton, Milk)* are the only unknown fluents, i.e. the only ones that are not mentioned in positive or negative form.

Grocery Shopping Version 4: sensorless

$Init$ $\big(Shelf(Fridge) \wedge \neg Shelf(FilmjölkCarton) \wedge \neg Shelf(MellanmjölkCarton) \wedge$
$\neg Accessible(Fridge) \wedge \neg Accessible(FilmjölkCarton) \wedge$
$\neg Accessible(MellanmjölkCarton) \wedge \neg InBasket(Fridge) \wedge \neg InBasket(FilmjölkCarton) \wedge$
$\neg InBasket(MellanmjölkCarton) \wedge \neg IsIn(Fridge, Fridge) \wedge$
$\neg IsIn(MellanmjölkCarton,\ Fridge) \wedge \neg IsIn(FilmjölkCarton,\ Fridge) \wedge$
$IsIn(Fridge,\ MellanmjölkCarton) \wedge IsIn(Fridge,\ FilmjölkCarton)$

$Goal$ $\big(Have(Milk) \wedge \neg Have(SourMilk)\big)$

$Action$ $\big(OpenDoor(x)$
PRECOND : $\neg Accessible(x) \wedge Shelf(x)$
EFFECT : $Accessible(x)\big)$

$Action$ $\big(CloseDoor(x)$
PRECOND :$Accessible(x) \wedge Shelf(x)$
EFFECT : $\neg Accessible(x)\big)$

$Action$ $\big(Add(x, y)$
PRECOND :$Accessible(x) \wedge IsIn(x, y)$
EFFECT : $InBasket(y) \wedge \neg IsIn(x, y)\big)$

$Action$ $\big(Pay(x, y)$
PRECOND :$InBasket(x) \wedge Contains(x,y)$
EFFECT : $\neg InBasket(x) \wedge Have(y)\big)$

3.4.1 Draw the space of *all belief states* that are reachable from the initial one (in one or more steps), assuming the above action scheme. Start from the initial state given in Figure 3.2 and draw belief states that can be reached from it in one or more steps. Similarly as in Question 3.1, for simplicity of the presentation, you do not have to explicitly list the fluents that are stay either always true or always false, just as we do in Figure 3.2. 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.*

A TIP: We say that you should label transitions with ground actions. This means that variables x, y in $Add(x, y)$ has to be substituted with a constant. 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?

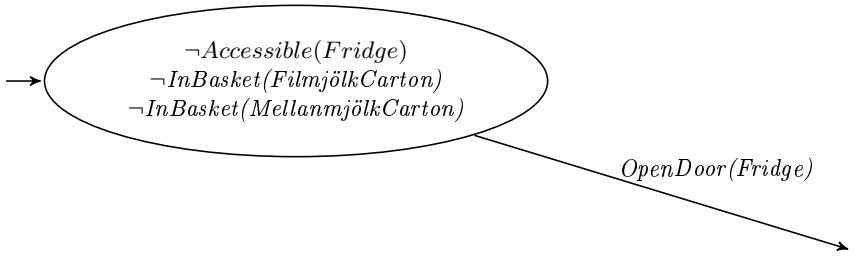

Figure 3.2: A part of the belief state space.

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

3.4.2 How many actual physical states does the initial belief state contain?

3.4.3 How many different plans that lead to the satisfaction of the goal are there?

3.5 B-level 10 points

Consider now yet another variant of the problem, with the action scheme from Section 3.4, you again cannot observe what the content of the cartons is, but can walk with your basket to a corner in a supermarket with good coverage and use google translate on your phone to find out what mellanmjölk and filmjölk are. The problem is not sensorless anymore. Formally, we add the following to the domain description:

$$\begin{aligned} \mathit{Percept}\ &\big(\mathit{Contains}(x, y) \\ &\quad \text{PRECOND}: \mathit{InBasket}(x) \end{aligned}$$

3.5.1 Draw a connected subgraph of the belief state space containing at least 7 different belief states. Again, for simplicity of presentation, you do not have to explicitly list the fluents that are either always true or always false, you can start from the initial state depicted in Figure 3.2. 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 know that a carton is $InBasket$, there should be two transitions, one with *Contains(FilmjölkCarton, Milk)* and one with *Contains(MellanmjölkCarton, SourMilk)* as a result of *Percept*.

3.5.2 Does there exist a *contingent plan* achieving the goal? If yes, find such a contingent plan and represent it as a sequence of actions and if-then-else structures. If no, explain why. An example of a contingent plan (that however does not satisfy the goal) is

[*OpenDoor*(*Fridge*),
Add(Fridge, FilmjölkCarton),
if *Contains(FilmjölkCarton, milk)*
then [*Pay(MellanmjölkCarton, milk)*]
else [*NoOp*]].

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

4 Robot

The following PDDL describes a robot in a 10x10 grid-like world that can move, pick objects, and drop them. The robot can hold only one object at a time and its goal is to gather all objects in cell $C_{1,1}$.

$$
\begin{aligned}
\mathit{Init}\ \big(&\mathit{Free}(\mathit{Robot})\ \wedge\ \mathit{At}(\mathit{Robot}, C_{1,1}) \wedge \\
&\mathit{At}(\mathit{Ball}, C_{4,6}) \wedge \mathit{At}(\mathit{Cube}, C_{3,7}) \wedge \mathit{At}(\mathit{Basket}, C_{10,2}) \wedge \\
&\mathit{Object}(\mathit{Ball}) \wedge \mathit{Object}(\mathit{Cube}) \wedge \mathit{Object}(\mathit{Basket}) \wedge \\
&\mathit{Next}(C_{1,1}, C_{1,2}) \wedge \mathit{Next}(C_{1,1}, C_{2,1}) \wedge \\
&\mathit{Next}(C_{1,2}, C_{1,1}) \wedge \mathit{Next}(C_{1,2}, C_{2,2}) \wedge \mathit{Next}(C_{1,2}, C_{1,3}) \wedge \ldots \wedge \\
&\mathit{Next}(C_{1,10}, C_{1,9}) \wedge \mathit{Next}(C_{1,10}, C_{2,10}) \wedge \ldots \wedge \\
&\mathit{Next}(C_{10,10}, C_{10,9}) \wedge \mathit{Next}(C_{10,10}, C_{9,10})\big) \\
\mathit{Goal}\ \big(&\mathit{At}(\mathit{Robot}, C_{1,1})\ \wedge\ \mathit{At}(\mathit{Ball}, C_{1,1})\ \wedge\ \mathit{At}(\mathit{Cube}, C_{1,1})\ \wedge\ \mathit{At}(\mathit{Basket}, C_{1,1})\big) \\
\mathit{Action}\ \big(&\mathit{Move}(\mathit{from}, \mathit{to}) \\
&\text{PRECOND}: \mathit{Next}(\mathit{from}, \mathit{to}) \wedge \mathit{At}(\mathit{Robot}, \mathit{from}) \\
&\text{EFFECT}: \neg \mathit{At}(\mathit{Robot}, \mathit{from}) \wedge \mathit{At}(\mathit{Robot}, \mathit{to})\big) \\
\mathit{Action}\ \big(&\mathit{Pick}(\mathit{thing}, \mathit{place}) \\
&\text{PRECOND}: \mathit{At}(\mathit{Robot}, \mathit{place}) \wedge \mathit{At}(\mathit{thing}, \mathit{place}) \wedge \mathit{Object}(\mathit{thing}) \wedge \mathit{Free}(\mathit{Robot}) \\
&\text{EFFECT}: \neg \mathit{At}(\mathit{thing}, \mathit{place}) \wedge \neg \mathit{Free}(\mathit{Robot}) \wedge \mathit{Holds}(\mathit{Robot}, \mathit{thing})) \\
\mathit{Action}\ \big(&\mathit{Drop}(\mathit{thing}, \mathit{place}) \\
&\text{PRECOND}: \mathit{At}(\mathit{Robot}, \mathit{place}) \wedge \mathit{Holds}(\mathit{Robot}, \mathit{thing}) \wedge \mathit{Object}(\mathit{thing}) \\
&\text{EFFECT}: \mathit{At}(\mathit{thing}, \mathit{place}) \wedge \mathit{Free}(\mathit{Robot}) \wedge \neg \mathit{Holds}(\mathit{Robot}, \mathit{thing}))\big)
\end{aligned}
$$

4.1 B-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 $H1 - H2$ define heuristics $h_1 - h_2$ 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?
- Are h_1 and h_2 admissible?

- Compare heuristics h_1 and h_2. Does one of them dominate the other?
- Compare heuristics H_1 and H_2 for a general problem expressed in PDDL. Does one heuristic always dominate the other one? Consider two cases: 1- when none of the fluents that appear in the *goal* are deleted by any of the actions 2- when there are some actions that delete some of the fluents that appear in the goal.

4.2 B-level 5 points

For the following questions, overview paper [1] and focus on understanding the following terms:

- *goal distance,*
- *dead end,*
- *recognized dead end,*
- *unrecognized dead end.*
- *local minimum*
- *maximal local minimum exit distance (mlmed)*

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

4.3 A-level 10 points

Draw a 4x4 grid world, where the initial position of the robot is the bottom-left corner cell and the goal is for the robot to get to the top-right corner cell. The robot can move to a cell that has a common edge with its current cell (i.e. no diagonal movement). Imagine that there are different costs for crossing different edges. For this problem:

- Make your own heuristic function. Assign heuristic values to each cell such that there are local minima in the state space. The heuristic should return zero if and only if the goal is reached.
- What is the *mlmed* in the state space with your designed heuristic?

4.4 A-level 10 points

Let us now consider a modified version of the robot in the grid, where the world is split into a city and a dark forest. Once robot enters the forest, it can only get out with a flashlight.

$Init\ \big(Free(Robot)\ \wedge\ At(Robot, C_{1,1})\wedge$
$At(Ball, C_{4,6}) \wedge At(Cube, C_{3,7}) \wedge At(Basket, C_{10,2})\wedge$
$Object(Ball) \wedge Object(Cube) \wedge Object(Basket)\wedge$
$Next(C_{1,1}, C_{1,2}) \wedge Next(C_{1,1}, C_{2,1})\wedge$
$Next(C_{1,2}, C_{1,1}) \wedge Next(C_{1,2}, C_{2,2}) \wedge Next(C_{1,2}, C_{1,3}) \wedge \ldots \wedge$
$Next(C_{1,10}, C_{1,9}) \wedge Next(C_{1,10}, C_{2,10}) \wedge \ldots \wedge$
$Next(C_{10,10}, C_{10,9}) \wedge Next(C_{10,10}, C_{9,10})\wedge$
$City(C_{1,1}) \wedge City(C_{1,2}) \wedge \ldots City(C_{1,5}) \wedge \ldots City(C_{2,1}) \ldots City(C_{10,5})\wedge$
$Forest(C_{1,6}) \wedge Forest(C_{1,7}) \wedge \ldots Forest(C_{1,10}) \wedge \ldots Forest(C_{2,6}) \ldots Forest(C_{10,10})\big)$
$Goal\ \big(At(Robot, C_{1,1})\ \wedge\ At(Ball, C_{1,1})\ \wedge\ At(Cube, C_{1,1})\ \wedge\ At(Basket, C_{1,1})\big)$
$Action\ \big(GrabFlashlight$
PRECOND :$At(Robot, C_{1,1})$
EFFECT :$HasFlashlight(Robot)\big)$
$Action\ \big(MoveInCity(from, to)$
PRECOND :$City(from) \wedge Next(from, to) \wedge At(Robot, from)$
EFFECT :$\neg At(Robot, from) \wedge At(Robot, to)\big)$
$Action\ \big(MoveInForest(from, to)$
PRECOND :$Forest(from) \wedge HasFlashlight(Robot) \wedge Next(from, to) \wedge At(Robot, from)$
EFFECT :$\neg At(Robot, from) \wedge At(Robot, to)\big)$
$Action\ \big(Pick(thing, place)$
PRECOND :$At(Robot, place) \wedge At(thing, place) \wedge Object(thing) \wedge Free(Robot)$
EFFECT :$\neg At(thing, place) \wedge \neg Free(Robot) \wedge Holds(Robot, thing))$
$Action\ \big(Drop(thing, place)$
PRECOND :$At(Robot, place) \wedge Holds(Robot, thing) \wedge Object(thing)$
EFFECT :$At(thing, place) \wedge Free(Robot) \wedge \neg Holds(Robot, thing))\big)$

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

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

4.5 A-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.