Royal Institute of Technology
DD2380 - Artificial Intelligence
A3 - Knowledge, Reasoning, and Planning

Marina Herrera Sarrias
19910601T720
mherrera@kth.se

1 ASSEMBLING TASK

1.1 E-LEVEL

The *assembling-domain.txt* file can be found on canvas.

1.2 D-LEVEL

The problem named *problem-with-no-solution* can be solved after adding the action **drop-wood**. The reason why this problem can not be solved without the **drop-wood** action is due to the fact that there is 3 people for assembling 3 pieces of wood, hence, in order for the domain to work we would need to add a new person to the problem or to define a new action so we can free one of the three members.

1.3 C-LEVEL

Consider the heuristics:

- **H1** Ignoring all preconditions of all actions in the action scheme.
- **H2** Ignoring all delete lists.

> What is the value of the heuristic function H_1 and H_2 for the initial state of *problem-1.pddl*, *problem-2.pddl* and *problem-3.pddl*

The value of H_1 in the initial state for the three problems is 1, 2 and 3 respectively, which correspond to the number of pieces of wood "approved" per problem.

In the case of H_2, if we ignore the delete list from the actions effect only the value of the heuristic will be infinite, as this setting will most likely lead to a dead end.

On the other hand, if we set H_2 to ignore the delete list from the action's preconditions and effects the value of the heuristic will be the following:

- **Problem 1** it will expect Alex to the all the work, i.e *grab wood, grab brush, grab hammer, brush, hammer* and *approve*.The heuristic value is of **6** actions.
- **Problem 2** it will expect Alex to *grab wood 1, grab wood 2, grab brush, grab hammer, brush wood 1, hammer wood 1, approve wood 1, brush wood 2, hammer wood 2* and *approve wood 2*. The heuristic value is of **10** actions.
- **Problem 3** it will expect Alex to *grab wood 1, grab wood 2, grab wood 3, grab brush, grab hammer, brush wood 1, hammer wood 1, approve wood 1, brush wood 2, hammer wood 2, approve wood 2, brush wood 3, hammer wood 3* and *approve wood 3*. The heuristic value is of **14** actions.

2 COMPANY

We know the following facts about the company members:

$S1 : ReportsTo(x, y) \wedge Floor(y) = 3 \implies ReportsTo(Rose, x)$
$S2 : Position(Frank) = Manager$
$S3 : ReportsTo(x, y) \wedge Floor(y) = 12 \implies Floor(y) = 3$
$S4 : Floor(Harry) = 9$
$S5 : ReportsTo(Frank, Lily)$
$S6 : Floor(x) = 3 \implies ReportsTo(x, Harry)$
$S7 : ReportsTo(x, Lily) \implies Floor(x) = 12$

2.1 E-LEVEL

Infer sentence $Floor(Frank) = 12$ using a sequence of Generalized Modus Ponens from the knowledge base including only the above 7 sentences $S1 - S7$.

For inferring the resolvent clause $X3 : Floor(Frank) = 12$ we will use the two sentences $S5$ and $S7$ such that the two clauses are $X1 : ReportsTo(Frank, Lily)$ and $X2 : ReportsTo(x, Lily) \implies Floor(x) = 12$. We derive that $p_1^{'}$ is $ReportsTo(Frank, Lily)$, p_1 is $ReportsTo(x, Lily)$, q is $Floor(x) = 12$ and $\theta = \{x/Frank\}$.

Hence, we can infer $Floor(Frank) = 12$ by applying the Generalized Modus Ponens inference rule as:

$$\frac{X1 : ReportsTo(Frank, Lily), X2 : ReportsTo(x, Lily) \implies Floor(x) = 12}{X3 : Floor(Frank) = 12}$$

2.2 E-LEVEL

Translate the sentences $S1 - S7$ to Conjunctive Normal Form (CNF) sentences $C1 - C7$

$C1 : \neg ReportsTo(x, y) \vee \neg Floor(y) = 3 \vee ReportsTo(Rose, x)$
$C2 : Position(Frank) = Manager$
$C3 : \neg ReportsTo(x, y) \vee \neg Floor(x) = 12 \vee Floor(y) = 3$
$C4 : Floor(Harry) = 9$
$C5 : ReportsTo(Frank, Lily)$
$C6 : \neg Floor(x) = 3 \vee ReportsTo(x, Harry)$
$C7 : \neg ReportsTo(x, Lily) \vee Floor(x) = 12$

2.3 E-LEVEL

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

For obtaining the resolvent clause $ReportsTo(Rose, Frank)$ we will first need to infer the clause $Floor(Lily) = 12$.

For obtaining the resolvent clause $X4 : Floor(Lily) = 3$ we will use the conjunctive normal form sentences **C1, C3** and the sentence inferred in **2.1** . The three clauses are $X1 : ReportsTo(Frank, Lily)$, $X2 : Floor(Frank) = 12$ and $X3 : \neg ReportsTo(x, y) \vee \neg Floor(x) = 12 \vee Floor(y) = 3$. l_1 is $ReportsTo(Frank, Lily)$, l_2 is $Floor(Frank) = 12$, m_1 is $ReportsTo(x, y)$, m_2 is $Floor(x) = 12$, m_3 is $Floor(y) = 3$. Where $UNIFY(l_1, m_1) = \{x/Frank, y/Lily\}$, l_1 and m_1 disappear, hence, we need to do

$SUBST(\{x/Frank, y/Lily\}, l_2 \vee l_3 \vee m_2)$. The resolution can be expressed by the Generalized Modus Ponen inference rule as:

$$\frac{X1 : ReportsTo(Frank, Lily), X2 : Floor(Frank) = 12, X3 : \neg ReportsTo(x, y) \vee \neg Floor(x) = 12 \vee Floor(y) = 3}{Floor(Lily) = 3} \tag{1}$$

Now, for obtaining the resolvent clause $X4 : ReportsTo(Rose, Frank)$ we will use the conjunctive normal form sentences **C1, C5** and the sentence inferred in Eq. (1). The three clauses are $X1 : ReportsTo(Frank, Lily)$, $X2 : Floor(Lily) = 3$ and $X3 : \neg ReportsTo(x, y) \vee \neg Floor(y) = 3 \vee ReportsTo(Rose, x)$. l_1 is $ReportsTo(Frank, Lily)$, l_2 is $Floor(Lily) = 3$, m_1 is $ReportsTo(x, y)$, m_2 is $Floor(y) = 3$, m_3 is $ReportsTo(Rose, x)$. Where $UNIFY(l_1, m_1) = \{x/Frank, y/Lily\}$, l_1 and m_1 disappear, hence, we need to do $SUBST(\{x/Frank, y/Lily\}, l_2 \vee l_3 \vee m_2)$.

$$\frac{X1 : ReportsTo(Frank, Lily), X2 : Floor(Lily) = 3, X3 : \neg ReportsTo(x, y) \vee \neg Floor(y) = 3 \vee ReportsTo(Rose, x)}{ReportsTo(Rose, Frank)} \tag{2}$$

2.4 D-LEVEL

Consider the sentence in Conjunctive Normal Form

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

Express $S1 - S8$ in natural English.

$S1$: The member that reports to its immediate superior who sits on the 3^{rd} floor is reported by Rosie.

$S2$: Frank occupies the manager position.

$S3$: The immediate superior that receives reports from the member sitting on the 12^{th} floor sits on the 3^{rd} floor.

$S4$: Harry sits on the 9^{th} floor.

$S5$: Frank reports to Lily.

$S6$: The member sitting on the 3^{rd} floor reports to Harry.

$S7$: The member who reports to Lily sits on the 12^{th} floor.

$S8$: The immediate superior that is reported by the member that occupies the manager position holds the director position.

2.5 C-LEVEL

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.

Consider the two grounded clauses $\neg B \vee A$ and $B \vee \neg A$. If we apply the resolution inference rule, we obtain:

$$\frac{\neg B \vee A, B \vee \neg A}{A \vee \neg A} \tag{3}$$

In Eq. (3), if we suppose B to be true in order for $\neg B \vee A$ to be true A must be true, alternatively, if we suppose B to be false, in order for the clause $B \vee \neg A$ to be true, A must be false. Hence, whenever both clauses hold true, the resolvent clause is true.

If we now change the order of the grounded clauses and apply the commutative law, we obtain the resolvent clause $B \vee \neg B$.

$$\frac{\neg A \vee B, A \vee \neg B}{B \vee \neg B} \tag{4}$$

2.6 C-LEVEL

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 $C4$.

Let's consider our model

$$m = \{BoderdPuppy = True, \quad DestroyedShoes = False, \quad PlayWithPuppy = True\}$$

Consider the knowledge base with four different grounded clauses:

- $C_1 : BoredPuppy$
- $C_2 : BoredPuppy \Rightarrow PlayWithPuppy$
- $C_3 : \neg DestroyedShoes$
- $C_4 : \neg PlayWithPuppy \Rightarrow DestroyedShoes$

For resolving the clauses, We will apply the resolution inference rule. In order to do so we will first need to translate C_2 and C_4 to the Conjunctive Normal Form. By the implication elimination equivalence we transform $C_2 = \neg BoredPuppy \vee PlayWithPuppy$, and by using the contraposition equivalence we convert $C_4 = \neg DestroyedShoes \Rightarrow PlayWithPuppy$, which by the implication elimination equivalence becomes $C_4 = DestroyedShoes \vee PlayWithPuppy$.

Resolving C_1 and C_2 yields to:

$$\frac{C_1 : BoredPuppy;\ C_2 : \neg BoredPuppy \vee PlayWithPuppy}{C_5 : PlayWithPuppy}$$

We can derive that l_1 is $BoredPuppy$, m_1 is $\neg BoredPuppy$, m_2 is $PlayWithPuppy$.

and Resolving C_3 and C_4 yields to:

$$\frac{C_3 : \neg DestroyedShoes;\ C_4 = DestroyedShoes \vee PlayWithPuppy}{C_6 : PlayWithPuppy}$$

We can derive that l_1 is $\neg DestroyedShoes$, m_1 is $DestroyedShoes$, m_2 is $PlayWithPuppy$.

2.7 B-LEVEL

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

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

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

$S11 : Floor(Rose) = 5$

The *einstein.txt* file in canvas contains the puzzle solution.

2.8 A-LEVEL

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.

By using a counter-example We would show that two grounded clauses cannot be solved together.

In order to do so we will introduce a new literal C and try to solve the following clauses using the resolution rule:

- $C_1 = C \quad , \quad C_2 = A \vee B \vee \neg C$

$$\frac{C, A \vee B \vee \neg C}{A \vee B} \tag{5}$$

- $C_1 = A \vee B \vee C \quad , \quad C_2 = \neg C$

$$\frac{A \vee B \vee C, \neg C}{A \vee B} \tag{6}$$

- $C_1 = B \vee C \quad , \quad C_2 = A \vee \neg C$

$$\frac{B \vee C, A \vee \neg C}{A \vee B} \tag{7}$$

- $C_1 = B \vee \neg C \quad , \quad C_2 = A \vee C$

$$\frac{B \vee \neg C, A \vee C}{A \vee B} \tag{8}$$

Notice that it does not matter what values the new literal C takes, the resolution will always be the same. $A \vee B$ and $\neg A \vee \neg B$ do not have the same value. Which can also be verified by looking at the true table:

A	B	$A \vee B$	$\neg A \vee \neg B$
T	T	T	F
T	F	T	T
F	T	T	T
F	F	F	T

3 GROCERY SHOPPING

3.1 E-LEVEL: STATE SPACE

The following state-space planning provides a structure in which all possible solutions in the reachable state space are represented. There are certain aspects we should take into consideration for defining this space.

- From the initial state the only action we can take is *OpenDoor(x)* for $x \in \{fridge, freezer\}$.
- After performing the action *OpenDoor(x)* we can proceed to *CloseDoor(x)* or *Add(x, y)*.
- Following the previous point, whenever $x = fridge$, $y \in \{MellanmjölkCarton, FilmjölkCarton\}$, or if $x = fridge$, $y = PizzaBox$.
- There is no restriction in the number of times we perform the actions *OpenDoor(x)* and *CloseDoor(x)*.
- After taking the action *OpenDoor(x)* we can complete our grocery shopping without taking the action *CloseDoor(x)*. Meaning that we can complete our shopping without closing the fridge/freezer door.
- We can perform the action *Pay(x,y)* before *CloseDoor(x)* (if we ever do).
- The action *Add(x, y)* can be performed with a maximum of three times per sequence of actions (plan).
- After performing the action *Add(x, y)* we can proceed to the action *Pay(x,y)*, *CloseDoor(x)* or if it is the case in which $x = fridge$ we can repeat the action *Add(x, y)* once more.

3.1.1 Draw a connected part of the *reachable state space* with at least 7 different states.

A part of the reachable state space containing 9 different states can be found below. For practical purposes I denoted MellanmjölkCarton, FilmjölkCarton and PizzaBox, with MC, FC and PB respectively. Also, the dots "..." indicates that it leads outside the 7 requested states.

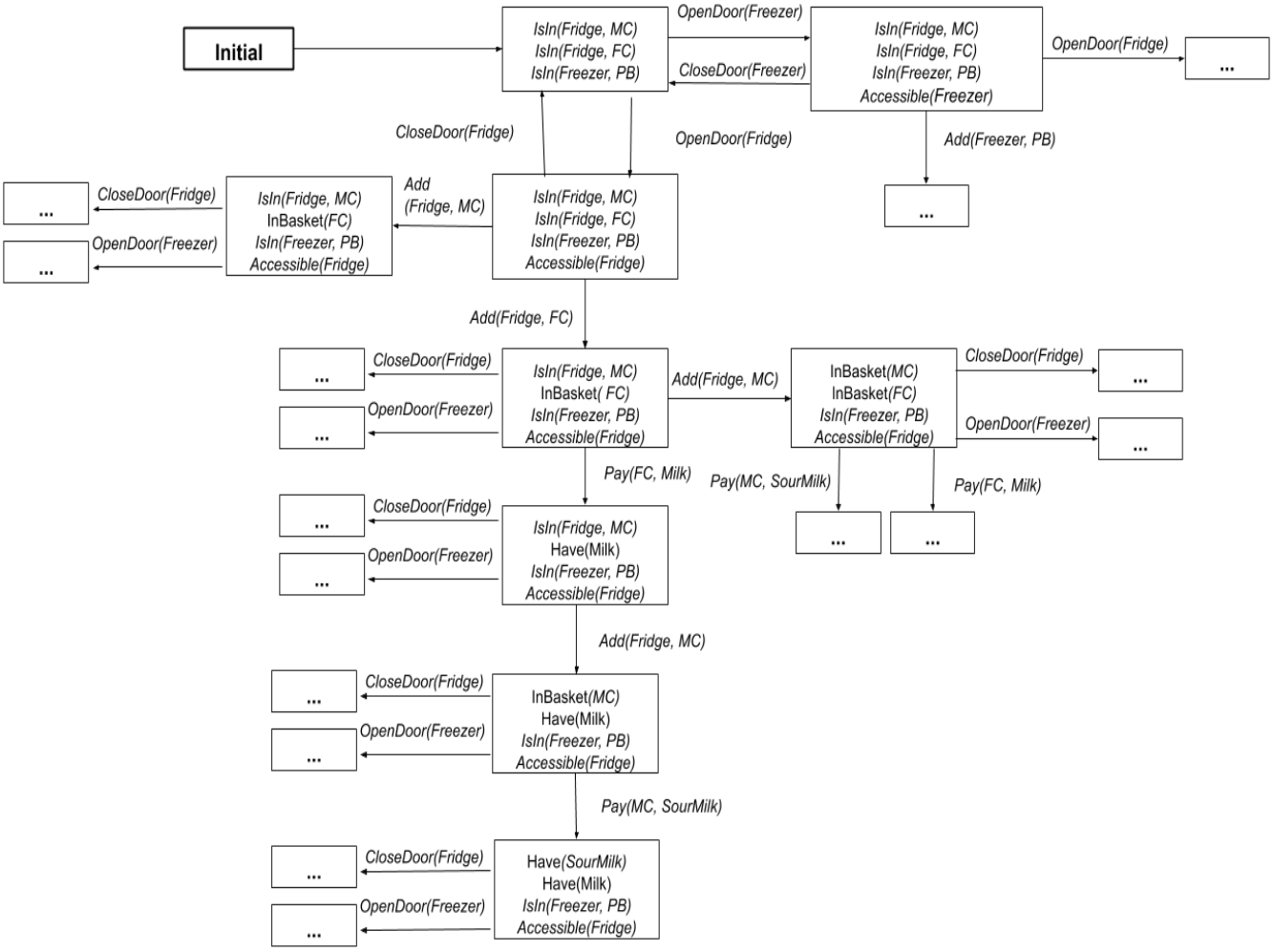

Figure 1: A part of the reachable state space

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

For computing the number of reachable states we will need to consider:

- The **fridge** could be either closed or open.
- The **freezer** can be either closed or open.
- The **pizza Box** can be either in the freezer, basket or paid.
- The **milk** carton can be either in the fridge, basket or paid.
- The **sour milk** carton can be either in the fridge, basket or paid.

Hence, the total number of possible states is obtain by multiplying $2^2 \cdot 3^3$ which results in **108** different reachable states.

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

There are **8** different goal states in the reachable state space satisfying the goal condition of buying milk and pizza only. $(Have(Milk) \wedge Have(Pizza) \wedge \neg Have(SourMilk))$

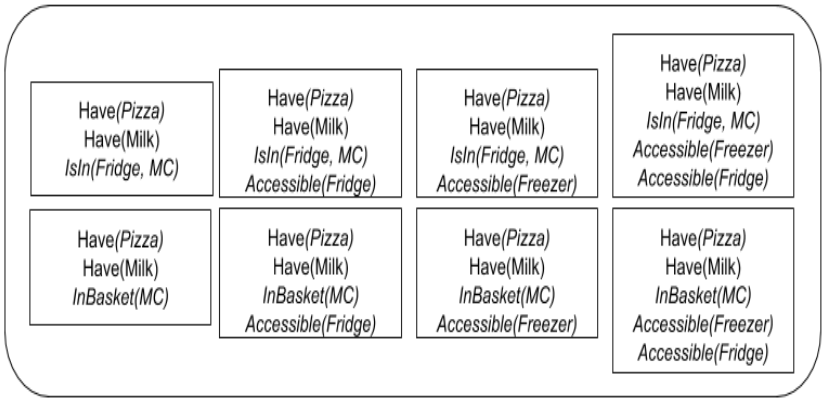

Figure 2: States satisfying the goal condition

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

There are infinite many plans leading to the goal condition, the reason for this is that despite the fact that there is only one available item per product (milk, sour milk, pizza), there is not a restriction in how many times we can open and close the fridge/freezer doors. Hence, these actions can be performed infinitely many times, leading to infinite number of plans satisfying the goal condition.

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.

Assuming that the reachable state space previously defined is correct. The optimal state-space plan satisfying the goal condition is achieved after a sequence of 6 actions. One of these paths can be:

$OpenDoor(Fridge) \rightarrow add(Fridge, MellanmjölkCarton) \rightarrow OpenDoor(Freezer) \rightarrow add(Freezer, PizzaBox) \rightarrow Pay(MellanmjölkCarton) \rightarrow Pay(PizzaBox)$

Mind that the order of actions taken can change i.e if we first open the fridge or the freezer, if we first pay the milk or the pizza. Although the sequence of actions are the same.

3.2 D-LEVEL: ALTERNATIVE STATE SPACE PROPERTIES

Grocery Shopping Version 2

We are now in a new environment where some of the considerations we made in the previous section have changed. We will now consider the following:

- We consider the new state *ClosedAfterOpeningDoor(x)* as we are only allowed to perform once the action *OpenDoor(x)*, which will mean that the state *CloseDoor(x)* will only concern to the initial state.
- Following the previous point, whenever $x = fridge$ $y \in \{MellanmjölkCarton, FilmjölkCarton\}$, or if $x = fridge$, $y = PizzaBox$.
- After performing the action *OpenDoor(x)* we can proceed to *ClosedAfterOpeningDoor(x)* or *Add(x, y)* but mind that we will not be able to perform the action *OpenDoor(x)*.
- After taking the action *OpenDoor(x)* we can complete our grocery shopping without taking the action *ClosedAfterOpeningDoor(x)*. Meaning that we can complete our shopping without closing the fridge/freezer door.
- We can perform the action *Pay(x,y)* before *ClosedAfterOpeningDoor(x)* (if we ever do).
- The action *Add(x, y)* can be performed with a maximum of three times per sequence of actions (plan).
- After performing the action *Add(x, y)* we can proceed to the action *Pay(x,y)*, *ClosedAfterOpeningDoor(x)* or if it is the case in which $x = fridge$ we can repeat the action *Add(x, y)* once more.

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

For computing the number of reachable states we will need to consider:

- The fridge could be either *closed*, *open* or *closedAfterOpening*.
- The **freezer** can be either *closed*, *open* or *closedAfterOpening*.
- The **pizza Box** can be either in the freezer, basket , or paid.
- The **milk** carton can be either in the fridge, basket or paid.
- The **sour milk** carton can be either in the fridge, basket or paid.

Hence, the total number of possible states is obtain by multiplying $2^3 \cdot 3^3$ which results in **216** different reachable states.

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

There will be 8 different goal states satisfying the goal condition, same as in Grocery Shopping Version 1.

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: YET ANOTHER ALTERNATIVE STATE SPACE PROPERTIES

We are now in a new environment where some of the considerations we made in the previous sections (Grocery Shopping Version 1 and 2) have changed. We will now consider the following:

- The number of times we perform the actions *OpenDoor(x)* and *ClosedAfterOpeningDoor(x)* is restricted to one.
- As the supermarket has now infinite stock, we can pay for an item and then add a new one. Thus, we will introduce a new state for the already bought items in the basket.
- Following the previous point, whenever $x = fridge$ $y \in \{MellanmjölkCarton, FilmjölkCarton\}$, or if $x = fridge$, $y = PizzaBox$.
- After performing the action *OpenDoor(x)* we can proceed to *ClosedAfterOpeningDoor(x)* or *Add(x, y)* but mind that we will not be able to perform the action *OpenDoor(x)*.
- After taking the action*OpenDoor(x)* we can complete our grocery shopping without taking the action *CloseDoor(x)*. Meaning that we can complete our shopping without closing the fridge/freezer door.
- We can perform the action *Pay(x,y)* before *ClosedAfterOpeningDoor(x)* (if we ever do).
- After taking the action *Pay(x,y)* we can perform the actions *Add(x, y)* , *OpenDoor(x)* (If x has not been opened before), *ClosedAfterOpeningDoor(x)* (if we will not need to access the fridge/freezer again) or *Pay(x,y)* (if there are still more items different than the one we have already paid).
- When we perform the action *Pay(x,y)* we are paying for all products in the basket that are MellanmjölkCarton, FilmjölkCarton or PizzaBox. For instance, if we have two PizzaBox we only perform the action $Pay(PizzaBox, Pizza)$ once.
- The action *Add(x, y)* can be performed infinitely many times per sequence of actions (plans), as long as the fridge/freezer do not require to be opened again.
- We will not be able to perform the action *Add(x, y)* if one of the prior actions was *ClosedAfterOpening-Door(x)*.
- After performing the action *Add(x, y)* we can proceed to the action *Pay(x,y)*,*ClosedAfterOpeningDoor(x)* or *Add(x, y)* (considering we will not need to perform *OpenDoor(x)* again).

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

For computing the number of reachable states we will need to consider:

- The fridge could be either *closed*, *open* or *closedAfterOpening*.
- The **freezer** can be either *closed*, *open* or *closedAfterOpening*.
- The **pizza Box** can be either in the freezer, basket , paid, or paid in basket.
- The **milk** carton can be either in the fridge, basket paid, or paid in basket.
- The **sour milk** carton can be either in the fridge, basket, paid, or paid in basket..

Hence, the total number of possible states is obtain by multiplying $2^3 \cdot 3^4$ which results in **648** different reachable states.

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

There will be **8** different goal states satisfying the goal condition, same as in Grocery Shopping Version 1 and 2. The goal states can be found in Figure 2.

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

There are infinite many plans leading to the goal condition, the reason for this is that despite the fact that the fridge/freezer doors can only be opened once, there is not a restriction in how many items can be added in the basket. Hence, adding items (milk, pizza) into the basket can be performed infinitely many times, leading to infinite number of plans satisfying the goal condition.

3.4 C-LEVEL: BELIEF STATE SPACE

3.4.1 Draw the space of *all belief states* that are reachable from the initial one.

The reachable belief state space contains 8 states.

Below can be found the physical states of the initial belief state and the space of all beliefs states that are reachable from the initial one. Notice that *Contain* is the only unknown fluent, hence in our belief state we should consider all possible *Contain* states while the rest of fluent remain the same as they are all known. (Fluents that are always true or always false are not included in the belief states, I hesitated about adding $IsIn$ but as there is only one item per product I did not considered this fluent to always be true.)

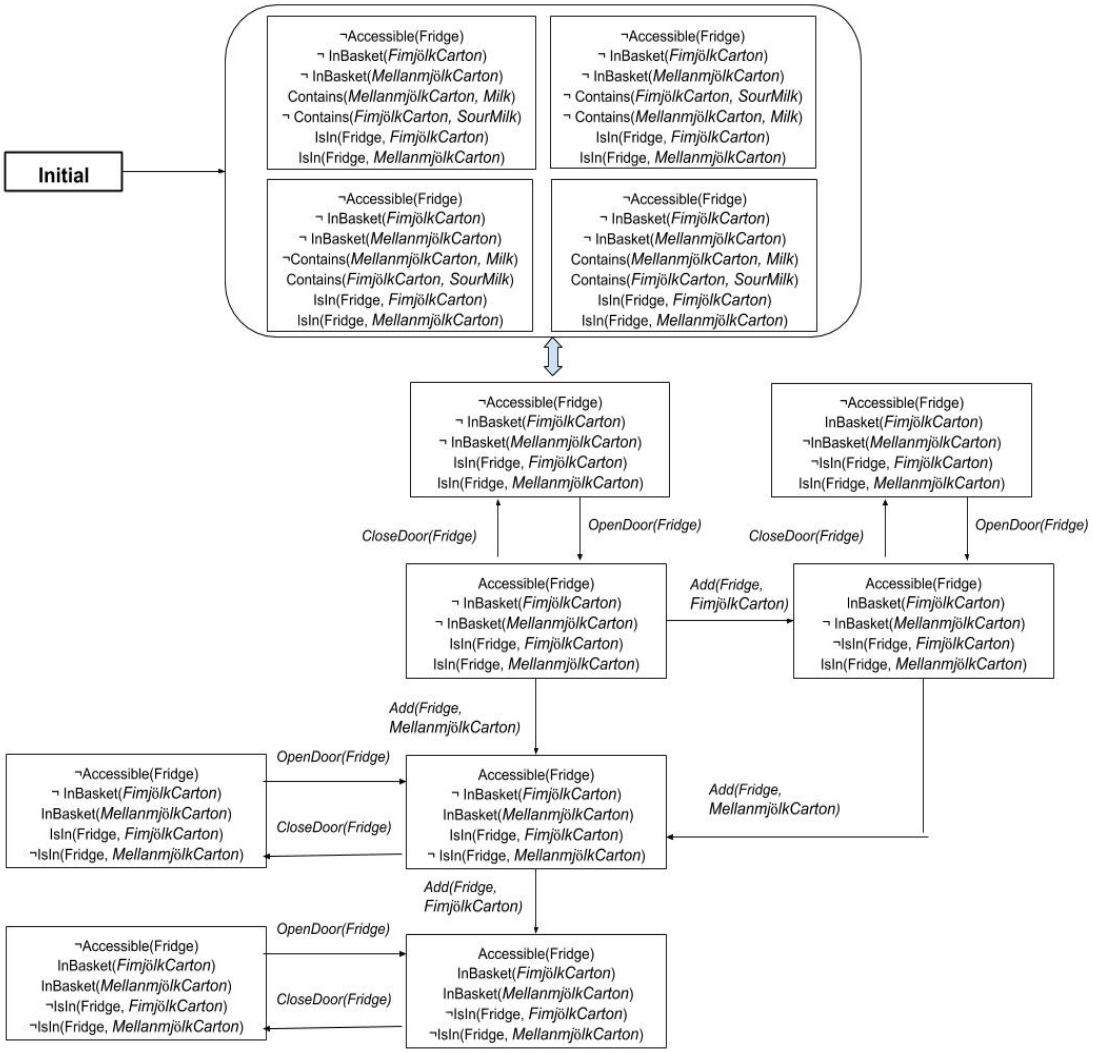

Figure 3: A part of the belief state space

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

There are **four** physical states indicating whether the cartons contain: milk only, sour milk only, both of them or something different. There is one physical state per each belief state. Typically the initial state is the set of all physical states.

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

There is no path to the goal state as we do not know what is the content of the carton we will never be able to satisfy the precondition of the pay action meaning that there will be no path to the goal.

3.5 B-LEVEL

3.5.1 Draw a connected sub-graph of the belief state space containing at least 7 different belief states.

For the following sub-graph we consider that the agent is in a partially-observable environment in which he can add a carton to the basket but without any knowledge of what the content is, i.e whether it contains

milk or Sour-Milk. *Percept* helps us to perceive the real value of the fluent, making the problem not to be sensor-less anymore. We start from the fact that we really do not know the meaning of FilmjölkCarton and MellanmjolkCarton. For instance, if our agent adds to the basket FilmjölkCarton, we will like to verify whether $Contains(FilmjölkCarton, milk)$ or $Contains(FilmjölkCarton, SourMilk)$, the latter I denoted it as $\neg Contains(FilmjölkCarton, milk)$. A part of the reachable state space containing 7 different states can be found below.

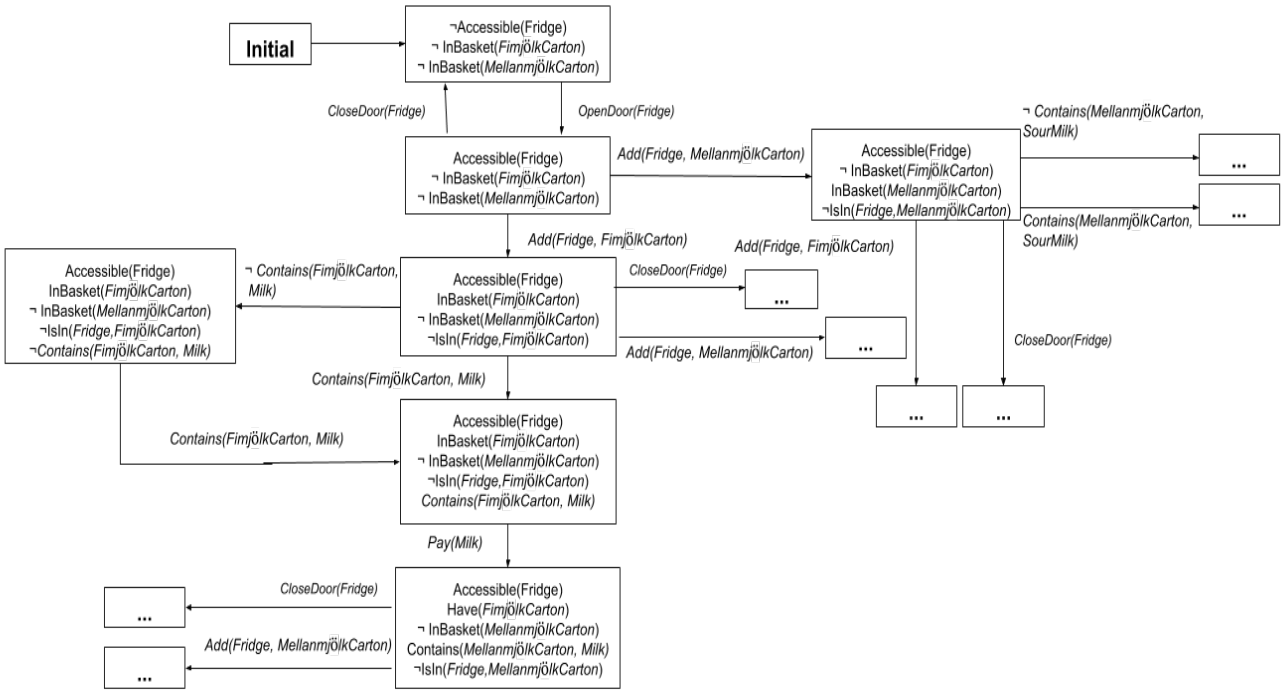

Figure 4: A part of the belief state space

3.5.2 Does there exist a *contingent plan* achieving the goal?

We now propose a *contingent plan* satisfying the goal condition $(Have(Milk) \wedge \neg Have(SourMilk))$ represented as a sequence of actions. Mind that we do not have knowledge of whether the FilmjölkCarton or MellanmjolkCarton carton contains Milk or SourMilk.

```
1  [OpenDoor(Fridge),
2  Add(Fridge, MellanmjolkCarton),
3  if Contains(MellanmjolkCarton, milk)
4      then[Pay(MellanmjolkCarton)]
5      else Add(Fridge, FilmjolkCarton)
6          if Contains(FilmjolkCarton, milk)
7              then[Pay(FilmjolkCarton)]
8              else[NoOp]].
```

4 ROBOT

4.1 B-LEVEL

What is the value of h_1 for the initial state?

Ignoring all preconditions of all actions in the action scheme means that the robot will be able to perform all actions in every state, meaning that he can perform the drop action regardless of where the robot is or if he is holding something. Therefore, the value of our heuristic will be **3** which is the number of times the action "Drop" needs to be performed in order to satisfy the goal (1 per item).

What is the value of h_2 for the initial state?

Ignoring delete lists means to ignore all effects that worsen the current situation i.e, removing all negative literals from effects (of the actions). Hence, the Robot will be able to re-visit all of those places she has already been, she will always be free and able to hold multiple objects at the same time. Hence, in order to compute the cost value of h_2 we should account for the following actions:

- A *pick* and a *drop* per object: **6** moves.
- Moves in order to reach the goal: **17**

Which yields to a heuristic value of $h_2 = \mathbf{23}$.

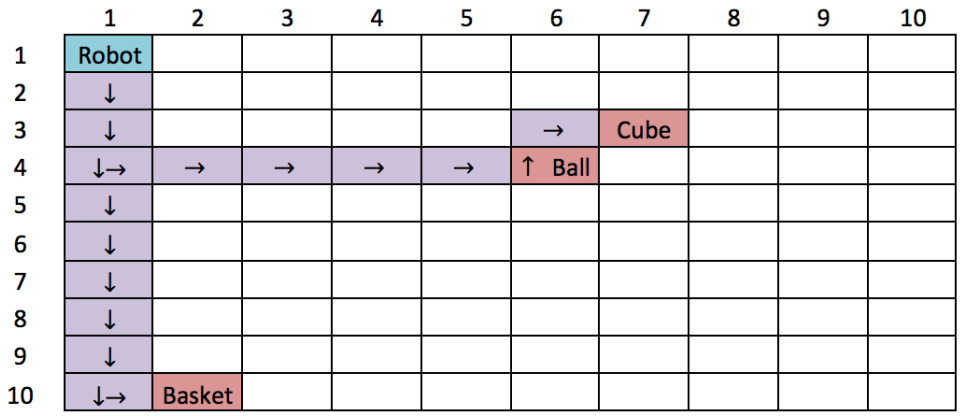

Figure 5: A possible Robot trajectory

Are h_1 and h_2 admissible?

Yes, They are both admissible as both of them do not overestimate the cost of reaching the goal. i.e, both heuristics reach the goal in less moves than what it actually takes.

Compare heuristics h_1 and h_2. Does one of them dominate the other?

Yes, h_2 dominates h_1. As h_2 has a higher heuristic value, since in h_1 we only need to *drop*, while for h_2 we need to *move, pick* and *drop* each item.

Compare heuristics H_1 and H_2 for a general problem expressed in PDDL. 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.

In the general case, h_2 will always dominate h_1.

4.2 B-LEVEL

Does there exist a reachable recognized dead end for H2?

In this problem there should be no dead ends, as there does not exists any negated predicates in any of the preconditions of the actions. Hence there could not be any recognized dead end for H2. (Which is the opposite as in **Question 1.3**)

Does there exist a reachable unrecognized dead end for H2?

Yes, we could reach unrecognized dead ends in H2. For example, if we take a look at the action *pick* if we ignore the effect $\neg Free(Robot)$ it would not consider the fact that the robot might be carrying something already another example could be the robot trying to drop an item without holding one. Carrying too many items could cause for instance a crash, from which the goal will be unreachable.

4.3 A-LEVEL

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:

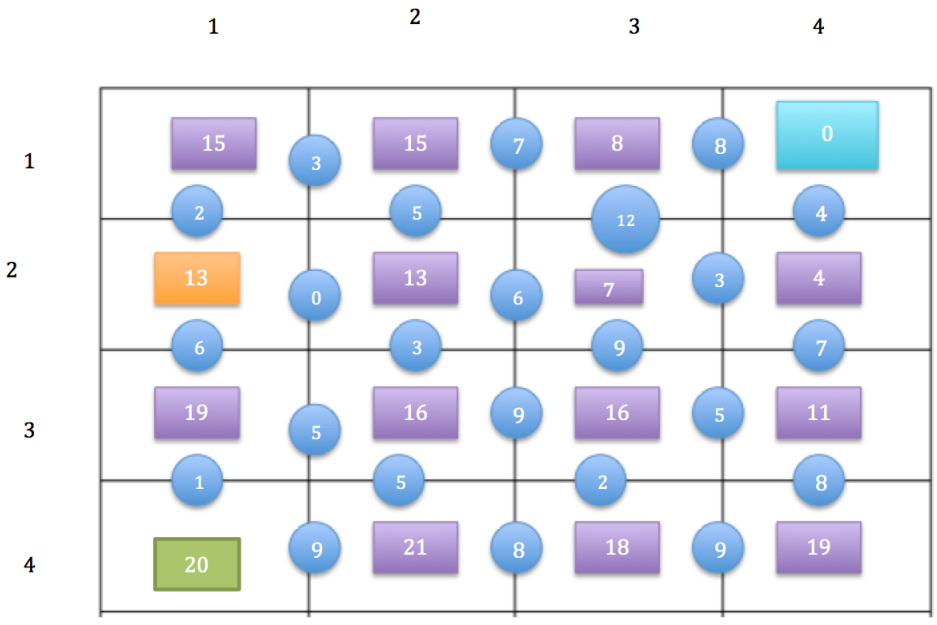

Figure 6: 4x4 grid world

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.

The heuristic I used is the output of the *Dijkstra's algorithm* computed with the cost for crossing the different edges. So for instance, the heuristic value of the initial position $(4, 1)$ will be the minimum between 19+1, coming from $(3, 1)$ and $21+9$ coming from $(4, 2)$, . We compute the heuristic value for all squares starting from the goal.

We can also notice that there is only one local minimum in $(2, 1)$ in our state space.

What is the *mlmed* in the state space with your designed heuristic?

The *maximal local minimum exit distance* is defined to be the maximum over the exit distances of all states on local minimum. As in our case there is only one local minimum the *mlmed* will be 13.

4.4 A-LEVEL

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.

Does there exist a reachable recognized dead end for H1?

No, there is not. As we are ignoring all preconditions of all actions in the action scheme, the goal can be reached in a single action so it is impossible to end up in a recognized dead end, meaning that whether we reach the goal or there is no solution for that goal.

Does there exist a reachable unrecognized dead end for H1?

Yes, because as we are ignoring preconditions whenever we performed the action $Moveinforest$ we ignore if the robot has a flashlight or not, which could lead to an unrecognized dead end.

Does there exist a reachable recognized dead end for H2?

No, as there are not negated predicates in any of the preconditions of the actions.

Does there exist a reachable unrecognized dead end for H2?

Yes, same reason as in *Question 4.2*, the robot might try to pick more items than (perhaps) what he can actually handle, or drop items without carrying any.

4.5 A-LEVEL

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

I think it really depends if the unrecognized dead ends are reachable or not, as we could set a precondition that will somehow avoid us reaching the unrecognized dead end. Otherwise if we manage to reach a dead end the goal will become unreachable.