

# Assignment-3: Knowledge, Reasoning, and Planning

## KTH, DD2380 Artificial Intelligence

Sevket Melih Zenciroglu (smzen@kth.se)

### 1. ASSEMBLING TASK

#### 1.1. E-LEVEL 10 POINTS

Uploaded.

#### 1.2. D-LEVEL 5 POINTS


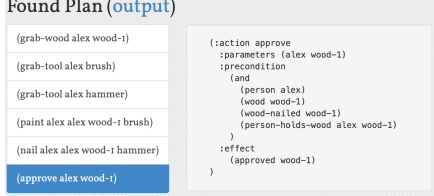

Uploaded.

#### 1.3. C-LEVEL 10 POINTS

**H1:** Ignoring all preconditions led us to the solution immediately but this is not an efficient way of solving a problem because, by this way all the woods have been approved without any action taken. In other words, the woods were not painted and nailed. So, the results are not satisfactory.

**H2:** Ignoring “not” cases everywhere in actions (preconditions and effects) looks like it gives us a better solution.

However, removing all “not” cases is also not a good idea because even if it gives a solution, when we check the details, we see that, e.g. for Problem-1, the simplest one, it expects Alex to do the whole work. Grabbing both brush and hammer.

Problem-1	H1	H2 (remove all “not” cases)	No Heuristics
Cost (# of actions)	1	6	7
Screenshot		<p>Found Plan (output)</p> 	<p>Found Plan (output)</p> 
Problem-2	H1	H2	No Heuristics
Cost (# of actions)	2	10	11

Screenshots	<p>Found Plan (output)</p> <pre> (approve wood-2 wood-2) (approve wood-2 wood-1)  (:action approve :parameters (wood-2 wood-2) :effect   (approved wood-2) )</pre>	<p>Found Plan (output)</p> <pre> (grab-wood alex wood-1) (grab-wood alex wood-2) (grab-tool alex brush) (grab-tool alex hammer) (paint alex alex wood-1 brush) (nail alex alex wood-1 hammer) (approve alex wood-1) (paint alex alex wood-2 brush) (nail alex alex wood-2 hammer) (approve alex wood-2)  (:action grab-wood :parameters (alex wood-1) :precondition   (and     (person alex)     (wood wood-1)     (free-wood wood-1)   ) :effect   (and     (person-holds-wood alex wood-1)     (person-holds-something alex)   ) )</pre>	<p>Found Plan (output)</p> <pre> (grab-tool maggie hammer) (grab-tool jessica brush) (grab-wood alex wood-2) (paint jessica alex wood-2 brush) (nail maggie alex wood-2 hammer) (approve alex wood-2) (drop-wood alex wood-2) (grab-wood alex wood-1) (paint jessica alex wood-1 brush) (nail maggie alex wood-1 hammer) (approve alex wood-1)  (:action grab-tool :parameters (maggie hammer) :precondition   (and     (person maggie)     (tool hammer)     (not       (person-holds-something maggie)     )   ) :effect   (and     (person-holds-tool maggie hammer)     (person-holds-something maggie)     (not       (free-tool hammer)     )   ) )</pre>
Problem-3	H1	H2	No Heuristics
Cost (# of actions)	3	14	15
Screenshots	<p>Found Plan (output)</p> <pre> (approve wood-3 wood-3) (approve wood-3 wood-2) (approve wood-3 wood-1)  (:action approve :parameters (wood-3 wood-3) :effect   (approved wood-3) )</pre>	<p>Found Plan (output)</p> <pre> (grab-wood alex wood-1) (grab-wood alex wood-2) (grab-wood alex wood-3) (grab-tool alex brush-1) (grab-tool alex hammer-1) (paint alex alex wood-1 brush-1) (nail alex alex wood-1 hammer-1) (approve alex wood-1) (paint alex alex wood-2 brush-1) (nail alex alex wood-2 hammer-1) (approve alex wood-2) (paint alex alex wood-3 brush-1) (nail alex alex wood-3 hammer-1) (approve alex wood-3)  (:action grab-wood :parameters (alex wood-1) :precondition   (and     (person alex)     (wood wood-1)     (free-wood wood-1)   ) :effect   (and     (person-holds-wood alex wood-1)     (person-holds-something alex)   ) )</pre>	<p>Found Plan (output)</p> <pre> (grab-tool john hammer-1) (grab-tool maggie brush-1) (grab-wood jessica wood-2) (paint maggie jessica wood-2 brush-1) (nail john jessica wood-2 hammer-1) (approve jessica wood-2) (grab-wood alex wood-3) (paint maggie alex wood-3 brush-1) (nail john alex wood-3 hammer-1) (approve alex wood-3) (drop-wood alex wood-3) (grab-wood alex wood-1) (paint maggie alex wood-1 brush-1) (nail john alex wood-1 hammer-1) (approve alex wood-1)  (:action grab-tool :parameters (john hammer-1) :precondition   (and     (person john)     (tool hammer-1)     (not       (person-holds-something john)     )   ) :effect   (and     (person-holds-tool john hammer-1)     (person-holds-something john)     (not       (free-tool hammer-1)     )   ) )</pre>

If H2 is designed by removing “not” cases from effects only and keeping them at preconditions, it gives a timeout even for Problem-1.

```

Suspected timeout.

--- OK.
Match tree built with 16 nodes.

PDDL problem description loaded:
  Domain: ASSEMBLING-DOMAIN
  Problem: PROBLEM-1
  #Actions: 16
  #Fluents: 13
Landmarks found: 1
Starting search with IW (time budget is 60 secs)...
rel_plan size: 7
#RP_fluents 9
Caption
{#goals, #UNNachieved, #Achieved} -> IW(max_w)

{1/1/0}:IW(1) -> [2][3][4];; NOT I-REACHABLE ;;
Total time: -1.51992e-09
Nodes generated during search: 16
Nodes expanded during search: 16
IW search completed
Starting search with BFS(novel,land,h_add)...
--[4294967295 / 8]--
--[1 / 8]--
--[1 / 5]--
Total time: -1.51992e-09
Nodes generated during search: 13
Nodes expanded during search: 7
Plan found with cost: 8.10971e-39
BFS search completed

```

## 2. COMPANY

### 2.1. E-LEVEL 5 POINTS

$$\frac{S5: ReportsTo(Frank, Lily) \quad S7: ReportsTo(x, Lily) \Rightarrow Floor(x) = 12}{Floor(Frank) = 12}, \theta = \{x/Frank\}$$

### 2.2. E-LEVEL 5 POINTS

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

$$S2: \quad Position(Frank) = manager$$

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

$$S4: \quad Floor(Harry) = 9$$

$$S5: \quad ReportsTo(Frank, Lily)$$

$$S6: \quad Floor(x) = 3 \Rightarrow ReportsTo(x, Harry)$$

$$S7: \quad ReportsTo(x, Lily) \Rightarrow Floor(x) = 12$$

$$C1: \neg (ReportsTo(x, y) \wedge Floor(y) = 3) \vee ReportsTo(Rose, x)$$

$$C1: \neg ReportsTo(x, y) \vee \neg (Floor(y) = 3) \vee ReportsTo(Rose, x)$$

$$C2: Position(Frank)=manager$$

C3:  $\neg(\text{ReportsTo}(x, y) \wedge \text{Floor}(x) = 12) \vee \text{Floor}(y) = 3$   
 C3:  $\neg\text{ReportsTo}(x, y) \vee \neg(\text{Floor}(x) = 12) \vee \text{Floor}(y) = 3$   
 C4:  $\text{Floor}(\text{Harry})=9$   
 C5:  $\text{ReportsTo}(\text{Frank}, \text{Lily})$   
 C6:  $\neg(\text{Floor}(x) = 3) \vee \text{ReportsTo}(x, \text{Harry})$   
 C7:  $\neg(\text{ReportsTo}(x, \text{Lily})) \vee \text{Floor}(x) = 12$

### 2.3. E-LEVEL 5 POINTS

$\text{ReportsTo}(\text{Rose}, \text{Frank})$

From the exercise 2.1, we know that x in C3 and C7 is Frank since  $\text{Floor}(\text{Frank})=12$ . So, we can update the equations accordingly:

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

$\frac{C5:\text{ReportsTo}(\text{Frank}, \text{Lily}) \quad C3:\neg\text{ReportsTo}(\text{Frank}, y) \vee \neg(\text{Floor}(\text{Frank})=12) \vee \text{Floor}(y)=3}{\text{Floor}(\text{Lily})=3}, \text{ where } \theta = \{y/\text{Lily}\}$

Then C1 can be updated as

C1:  $\neg\text{ReportsTo}(x, \text{Lily}) \vee \neg(\text{Floor}(\text{Lily}) = 3) \vee \text{ReportsTo}(\text{Rose}, x)$

$\frac{C5:\text{ReportsTo}(\text{Frank}, \text{Lily}) \quad C1:\neg\text{ReportsTo}(x, \text{Lily}) \vee \neg(\text{Floor}(\text{Lily})=3) \vee \text{ReportsTo}(\text{Rose}, x)}{\text{ReportsTo}(\text{Rose}, \text{Frank})}, \text{ where } \theta = \{x/\text{Frank}\}$

### 2.4. D-LEVEL 5 POINTS

S1: The person who is at the 3<sup>rd</sup> floor is reported by the person who Rose reports.

S2: Frank is the manager.

S3: The person at the 12<sup>th</sup> floor reports to the person at the 3<sup>rd</sup> floor.

S4: Harry is at 9<sup>th</sup> floor.

S5: Frank reports to Lily

S6: The person at the 3<sup>rd</sup> floor reports to Harry.

S7: The person who reports to Lily is at 12<sup>th</sup> floor.

C8:  $\neg\text{Position}(x) = \text{Manager} \vee \neg\text{ReportsTo}(x, y) \vee \text{Position}(y) = \text{Director}$

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

S8:  $\text{Position}(x) = \text{Manager} \wedge \text{ReportsTo}(x, y) \Rightarrow \text{Position}(y) = \text{Director}$

S8: The person who reports to the director is the manager.

### 2.5. C-LEVEL 5 POINTS

C1:  $\text{Position}(x)=\text{Manager} \vee \text{ReportsTo}(x, y): A \vee B$

C2:  $\neg\text{Position}(x)=\text{Manager} \vee \neg\text{ReportsTo}(x, y): \neg A \vee \neg B$

Resolution-1:  $\frac{A \vee B, \neg A \vee \neg B}{A \vee \neg A}$

Resolution-2:  $\frac{A \vee B, \neg A \vee \neg B}{B \vee \neg B}$

In both cases, complementary literals eliminate each other.

## 2.6. C-LEVEL 5 POINTS

C1: Position(x)=Manager: A

C2:  $\neg \text{Position}(x)=\text{Manager} \vee \neg \text{ReportsTo}(x,y): \neg A \vee \neg B$

C3: Floor(x)=1  $\vee \neg \text{ReportsTo}(x,y): C \vee \neg B$

C4:  $\neg \text{Floor}(x)=1: \neg C$

Resolution-C1, C2:  $\frac{A, \neg A \vee \neg B}{\neg B} \equiv \text{Resolution-C3, C4: } \frac{C \vee \neg B, \neg C}{\neg B}$

## 2.7. B-LEVEL 10 POINTS

Uploaded.

```
% Render the employees term as a nice table.
:- use_rendering(table,
    [header(person('name', 'position', 'floor'))]).

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,_))).

company(Com) :-
    % each person in the list Com of company is represented as:
    % person(name; position; floor).
    length(Com,4),
    reportsTo(person(rose,_,_),person(_,_,_),Com), % 1
    reportsTo(person(_,_,_),person(_,_,3),Com), % 2.1
    member(person(frank,manager,_) , Com), % 2.2
    reportsTo(person(frank,manager,_) , Com), % 3
    reportsTo(person(_,_,12),person(_,_,3),Com), % 4
    member(person(harry,_,9),Com), % 5
    reportsTo(person(frank,_,_),person(lily,_,_),Com), % 6
    reportsTo(person(_,_,3),person(harry,_,_),Com), % 7
    reportsTo(person(_,_,12),person(lily,_,_),Com), % 8
    reportsTo(person(_,manager,_) , person(_,director,_) , Com), % 9
    reportsTo(person(_,intern,_) , person(_,manager,_) , Com), % 10
    reportsTo(person(_,director,_) , person(_,ceo,_) , Com), % 11
    member(person(rose,_,5),Com). % 12
```

name	position	floor
rose	intern	5
frank	manager	12
lily	director	3
harry	ceo	9

Next 10 100 1,000 Stop

company(Com)

name	position	floor
rose	intern	5
frank	manager	12
lily	director	3
harry	ceo	9

Next 10 100 1,000 Stop

?- company(Com)

## 2.8. A-LEVEL 10 POINTS

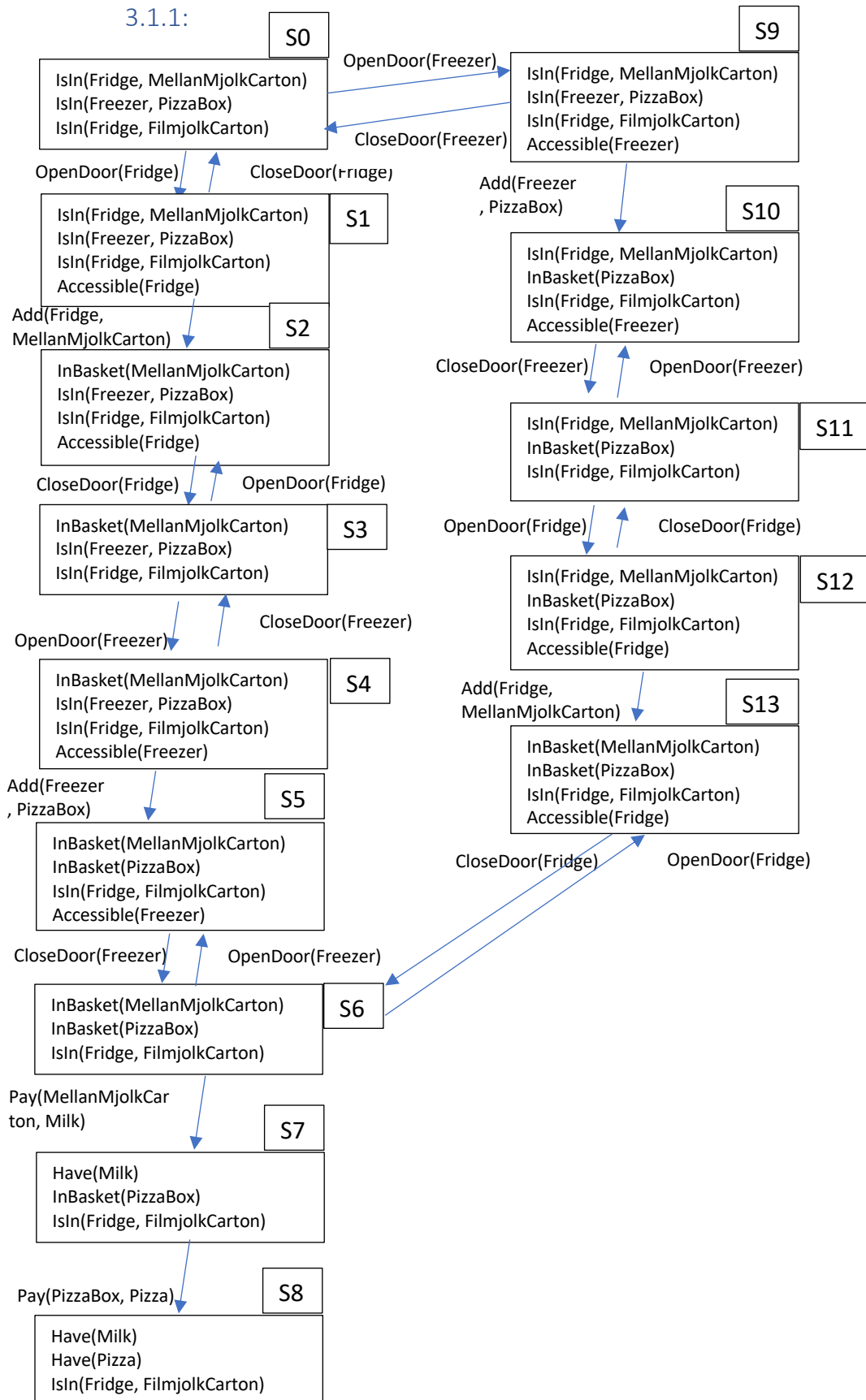
Questions 3 is below



### 3. GROCERY SHOPPING

#### 3.1. E-LEVEL 10 POINTS: STATE SPACE

##### 3.1.1:



### 3.1.2:

There are 3 possible states for the products:  $IsIn(x)$ ,  $InBasket(x)$ ,  $Have(x)$

$x$  can be one of these 3 products: Milk, Pizza, SourMilk

(Since the carton or box will include any of those items related to them, I simplify “ $x$ ” with the product names.)

This gives us  $3^3=27$  possible states.

(A product has to be in one of the states, that’s why “not States” haven’t been listed as an option, i.e.  $\neg IsIn(x)$ . If a product is at the state of  $\neg IsIn(x)$ , it means that it’s either at  $InBasket(x)$  or at  $Have(x)$  state)

There are 2 possible states for storages:  $Accessible(x)$ ,  $\neg Accessible(x)$

There are 2 storages: Fridge, Freezer

This gives us  $2^2=4$  possible states.

(here we must mention the “not State”. Because the storage can be accessible or not, there is no alternative state and also it is not accessible all the time.)

If we multiply those 2 results, we find the  $27*4=108$  different states in the whole reachable state space.

### 3.1.3:

The answer is 8.

We have 2 possible final states of products:

- 1)  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsIn(Fridge, FilmjolkCarton)$
- 2)  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsBasket(Fridge, FilmjolkCarton)$

There are 2 possible states for storages:  $Accessible(x)$ ,  $\neg Accessible(x)$

There are 2 storages: Fridge, Freezer

This gives us  $2^2=4$  possible states.

(as explained in [Question-3.1.2](#))

If we multiply these 2 results= $2*4=8$

State0:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsIn(Fridge, FilmjolkCarton)$

State1:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsIn(Fridge, FilmjolkCarton)$ ,  $Accessible(Fridge)$

State2:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsIn(Fridge, FilmjolkCarton)$ ,  $Accessible(Freeze)$

State3:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsIn(Fridge, FilmjolkCarton)$ ,  $Accessible(Fridge)$ ,  $Accessible(Freeze)$

State4:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsBasket(Fridge, FilmjolkCarton)$

State5:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsBasket(Fridge, FilmjolkCarton)$ ,  $Accessible(Fridge)$

State6:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsBasket(Fridge, FilmjolkCarton)$ ,  $Accessible(Freeze)$

State7:  $Have(Milk)$ ,  $Have(Pizza)$ ,  $IsBasket(Fridge, FilmjolkCarton)$ ,  $Accessible(Fridge)$ ,  $Accessible(Freeze)$

### 3.1.4:

Since there’s no limitation of opening and closing the doors, it should be infinite.

If we are not allowed to play open/close doors but instead (leave the door open and move forward) or (close the door and move forward, don’t go back to open it again):

**without FilmjölKCarton is considered:**

leave both doors open= $C(6,3)=20$

close both doors= $C(8, 3)=56$

close freezer door= $C(7, 3)=35$

close fridge door= $C(7, 3)=35$

$=20+56+42+42=146$  different plans can lead to the satisfaction of the goal condition.

**With FilmjölKCarton is considered:**

with FilmjölKCarton is considered

[

3 more actions added:

Add(Fridge, FilmjölKCarton)

Open(Fridge) for FilmjölKCarton

Close(Fridge) for FilmjölKCarton

]

leave both doors open= $C(9,3)=84$

close both doors= $C(11, 3)=165$

close freezer door= $C(10, 3)=120$

close fridge door= $C(10, 3)=120$

$=84+165+120+120=489$  different plans

**With FilmjölKCarton is considered-not closing the fridge:**

with FilmjölKCarton is considered

[

3 more actions added:

Add(Fridge, FilmjölKCarton)

Open(Fridge) for FilmjölKCarton

]

leave both doors open= $C(8,3)=56$

close both doors= $C(10, 3)=120$

close freezer door= $C(9, 3)=84$

close fridge door= $C(9, 3)=84$

$=56+120+84+84=344$  different plans

**With FilmjölKCarton is considered-opening the fridge once for both milks:**

with FilmjölKCarton is considered

[

3 more actions added:

Add(Fridge, FilmjölKCarton)

]

leave both doors open= $C(7,3)=35$

close both doors= $C(9, 3)=84$

close freezer door= $C(8, 3)=56$

close fridge door= $C(8, 3)=56$

$=35+84+56+56=231$  different plans



SUM=146+489+344+231=1290

Below shows the sample for both doors open:

	A	B	C	D	E	F	G	H
1	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Fridge, MellanMjolkCarton)	Add(Freezer, PizzaBox)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)	leave both doors open=C(6,3)=20 close both doors=C(8, 3)=56 close freezer door=C(7, 3)=42 close fridge door=C(7, 3)=42  =20+56+42+42=160 different plans	
2	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Fridge, MellanMjolkCarton)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
3	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)		
4	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)		
5	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Add(Fridge, MellanMjolkCarton)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
6	OpenDoor(Fridge)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)		
7	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)		
8	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
9	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	OpenDoor(Freezer)	Pay(MellanMjolkCarton, Milk)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)		
10	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)		
11	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Add(Freezer, PizzaBox)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)		
12	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
13	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)		
14	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Freezer, PizzaBox)	Add(Fridge, MellanMjolkCarton)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
15	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Freezer, PizzaBox)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)		
16	OpenDoor(Freezer)	OpenDoor(Fridge)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)		
17	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)	Pay(PizzaBox, Pizza)		
18	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Pay(PizzaBox, Pizza)	Pay(MellanMjolkCarton, Milk)		
19	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	OpenDoor(Fridge)	Pay(PizzaBox, Pizza)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)		
20	OpenDoor(Freezer)	Add(Freezer, PizzaBox)	Pay(PizzaBox, Pizza)	OpenDoor(Fridge)	Add(Fridge, MellanMjolkCarton)	Pay(MellanMjolkCarton, Milk)		
21				combination(6,3)				
22				6=total number of actions				
23				3=number of satisfactory states    Have(Milk), Have(Pizza), not Have(SourMilk)				

3.1.5:

One of the optimal plans would be (*assuming that we don't have to close any of the doors*):  
OpenDoor(Fridge) >> Add(Fridge, MellanMjolkCarton) >> OpenDoor(Freezer) >> Add(Freezer, PizzaBox) >>  
Pay(MellanMjolkCarton, Milk) >> Pay(PizzaBox, Pizza)

*If we have to close all open doors* before proceeding to the next action, then one of the optimal plans would be:  
OpenDoor(Fridge) >> Add(Fridge, MellanMjolkCarton) >> CloseDoor(Fridge) >> OpenDoor(Freezer) >>  
Add(Freezer, PizzaBox) >> CloseDoor(Freezer) >> Pay(MellanMjolkCarton, Milk) >> Pay(PizzaBox, Pizza)

I say one of the because some of the orders above can be changed but the total number of actions taken will not change.

3.2. D-LEVEL 5 POINTS: ALTERNATIVE STATE SPACE PROPERTIES

3.2.1:

This time, we can't open and close the door for each type of Milk which reduces the number of possible plans but not the number of states. CloseDoor(x) affects the status of Accessible(x) and makes it not Accessible(x) whereas it doesn't change the status of DoorOpened(x).

There are 3 possible states for the products: IsIn(x), InBasket(x), Have(x)  
x can be one of these 3 products: Milk, Pizza, SourMilk  
(Since the carton or box will include any of those items related to them, I simplify "x" with the product names.)  
This gives us 3^3=27 possible states.  
(A product has to be in one of the states, that's why "not States" haven't been listed as an option, i.e. ¬IsIn(x). If a product is at the state of ¬IsIn(x), it means that it's either at InBasket(x) or at Have(x) state)

There are 2 possible states for storages: Accessible(x), ¬Accessible(x)  
There are 2 storages: Fridge, Freezer

This gives us  $2^2=4$  possible states.

(here we must mention the “not State”. Because the storage can be accessible or not, there is no alternative state and also it is not accessible all the time.)

If we multiply those 2 results, we find the  $27*4=108$  different states in the whole reachable state space.

### 3.2.2:

The number of goal states shouldn't change since Accessible(x) and DoorOpened(x) occurs together. Only, we will need to add DoorOpened(x) everywhere we see Accessible(x) in [Question-3.1.3](#).

The answer is 8.

We have 2 possible final states of products:

- 1) Have(Milk), Have(Pizza), IsIn(Fridge, FilmjolkCarton)
- 2) Have(Milk), Have(Pizza), IsBasket(Fridge, FilmjolkCarton)

There are 2 possible states for storages: Accessible(x),  $\neg$ Accessible(x)

There are 2 storages: Fridge, Freezer

This gives us  $2^2=4$  possible states.

(as explained in [Question-3.1.2](#))

If we multiply these 2 results= $2*4=8$  is the answer.

### 3.2.3:

With the same logic in [Question-3.1.4](#), this time we can't open the fridge door more than once: without FilmjolkCarton is considered

leave both doors open= $C(6,3)=20$

close both doors= $C(8, 3)=56$

close freezer door= $C(7, 3)=35$

close fridge door= $C(7, 3)=35$

= $20+56+35+35=146$  different plans

with FilmjolkCarton is considered

[

1 more action added:

Add(Fridge, FilmjolkCarton)

]

leave both doors open= $C(7,3)=35$

close both doors= $C(9, 3)=84$

close freezer door= $C(8, 3)=56$

close fridge door= $C(8, 3)=56$

= $35+84+56+56=231$  different plans

SUM= $146+231=377$

### 3.3. D-LEVEL 5 POINTS: YET ANOTHER ALTERNATIVE STATE SPACE PROPERTIES

#### 3.3.1:

Same as [3.2.1](#), which is 108. Because, the positive states we are looking don't change. The only difference will be not having "IsIn(x,y)" state anymore. But we already don't show not cases. So, it doesn't affect the number of states.

#### 3.3.2:

The answer is 12.

This one changes because the previous goal state in 3.2.2 was Have(Milk) and Have(Pizza) and **not(Have(SourMilk))**

Now:

Have(Milk) and Have(Pizza)

That means, we can buy SourMilk before we buy Milk and Pizza. Then the conditions are updated as:

We have 3 possible final states of products:

- 1) Have(Milk), Have(Pizza), IsIn(Fridge, FilmjolkCarton)
- 2) Have(Milk), Have(Pizza), IsBasket(Fridge, FilmjolkCarton)
- 3) Have(Milk), Have(Pizza), Have(SourMilk)

There are 2 possible states for storages: Accessible(x),  $\neg$ Accessible(x)

There are 2 storages: Fridge, Freezer

This gives us  $2^2=4$  possible states.

If we multiply these 2 results= $3*4=12$  is the answer

#### 3.3.3:

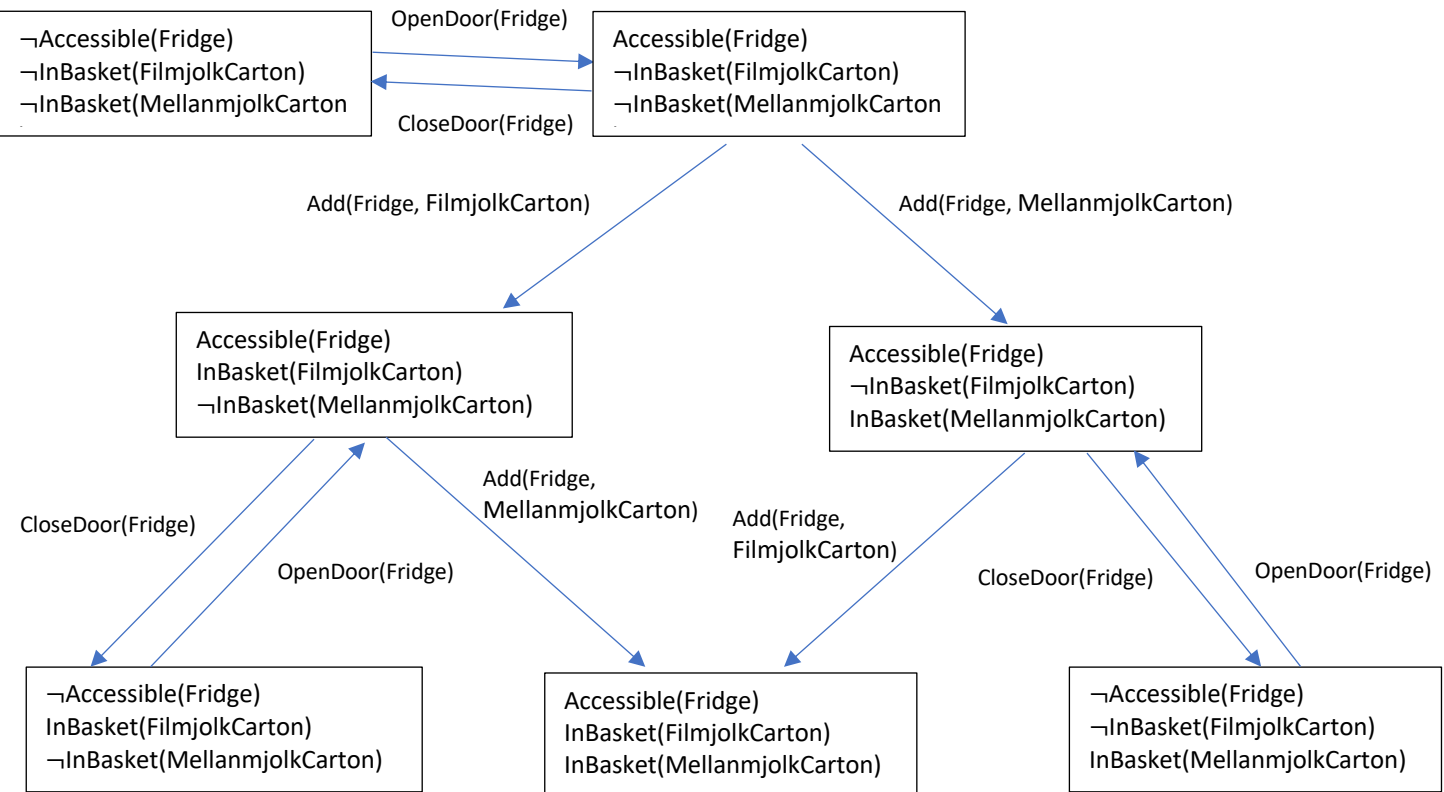
Since there is no limit of adding products, it is infinite.

If there is a limit, the calculations in [3.2.3](#) would be valid.



### 3.4. C-LEVEL 10 POINTS: BELIEF STATE SPACE

#### 3.4.1:



Since we don't have Contains(x, y) in Init and any of the actions gives us that, we cannot satisfy the preconditions for Pay(x, y) and stuck here.

#### 3.4.2:

If the cartons can contain other drinks, as well, there will be 2 different cartons on the shelf with 3 possible content in them.

Cartons: FilmjolkCarton, MellanmjolkCarton

Contents: SourMilk, Milk, Others

Then the answer is  $3^2=9$

If we don't consider "others" as an option for contents:

Cartons: FilmjolkCarton, MellanmjolkCarton

Contents: SourMilk, Milk

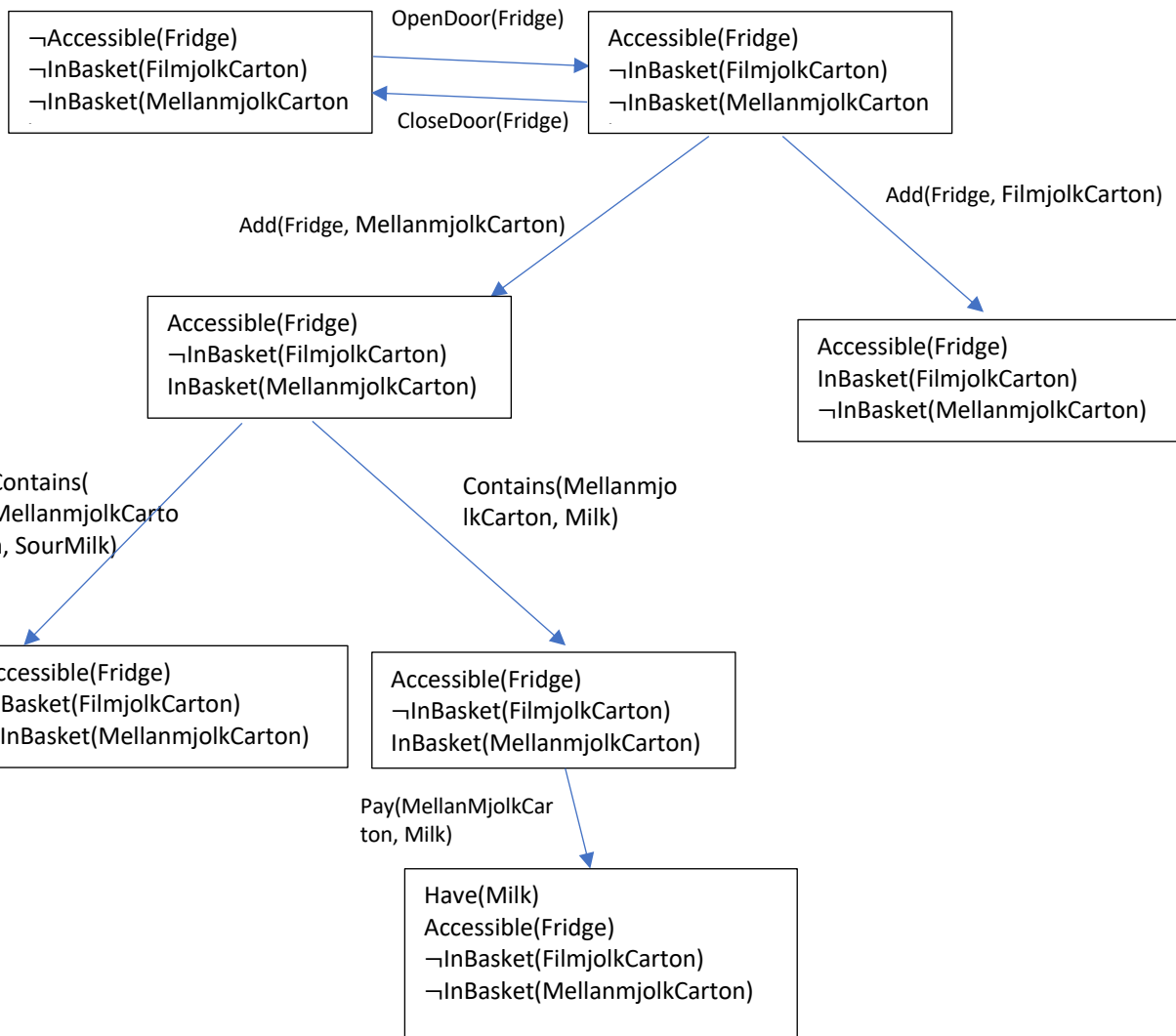
Then the answer is  $2^2=4$

#### 3.4.3:

We cannot reach our goal Have(Milk) and not-Have(SourMilk) since we cannot satisfy the precondition for Pay(x, y) as stated in 3.4.1. That means, there is no plan available.

### 3.5. B-LEVEL 10 POINTS

#### 3.5.1:



#### 3.5.2:

```

OpenDoor(Fridge);
Add(Fridge, MellanmjolkCarton);
If Contains(MellanmjolkCarton, milk)
then [ Pay(MellanmjolkCarton, milk) ]
else [ NoOp ]:

```