Pen and Paper Assignment:
Knowledge, Reasoning, and Planning

ALEXANDROS NICOLAOU
Alexandros | alenic @kth.se

March 6, 2020

1 Assembling Task

1.1 E - Level: Complete the file of assembling domain

As submitted in A3.PP1.PDDL:

einstein.txt

```
;; Domain definition
(define (domain assembling-domain)

  ;; Predicates: Properties of objects that we are interested in (boolean)
  (:predicates
    (PERSON ?x) ; True if x is a person
    (WOOD ?x) ; True if x is a wood
    (TOOL ?x) ; True if x is a tool
    (CAN-PAINT ?x) ; True if x is a tool that can be used for painting
    (CAN-HIT ?x) ; True if x is a tool that can be used for hitting
    (free-tool ?x) ; True if tool x is free for grabs
    (free-wood ?x) ; True if wood x is free for grabs
    (person-holds-something ?x) ; True if person x holds something
    (person-holds-tool ?x ?y) ; True if person x holds tool y
    (person-holds-wood ?x ?y) ; True if person x holds wood y
    (wood-painted ?x) ; True if x is a wood and it is painted
    (wood-nailed ?x) ; True if x is a wood and it contains a nail
    (approved ?x) ; True if x is correctly assembled
  )

  ;; Actions: Ways of changing the state of the world

  ; Person x can grab a tool y if they do not hold anything yet and if the tool is free
  ; As a result they hold the tool, and the tool is not free any more
  ; Parameters:
  ; - x: the person
  ; - y: the tool
  (:action grab-tool
    ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x ?y)
    :precondition (and (PERSON ?x) (TOOL ?y) (free-tool ?y) (not(person-holds-something ?x))
    :effect (and (not(free-tool ?y)) (person-holds-tool ?x ?y) (person-holds-something ?x))
  )

  ; Person x can drop a tool y if they hold the tool
  ; As a result the person stops holding the tool, and the tool becomes free
  ; Parameters:
  ; - x: the person
  ; - y: the tool
  (:action drop-tool
     ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x ?y)
    :precondition (and (PERSON ?x) (TOOL ?y) (person-holds-tool ?x ?y))
    :effect (and (free-tool ?y) (not(person-holds-tool ?x ?y)) (not(person-holds-something ?x
  )

  ; Person x can grab a wood y if they do not hold anything yet and if the wood is free
  ; As a result they hold the wood, and the wood is not free any more
  ; Parameters:
  ; - x: the person
  ; - y: the wood
  (:action grab-wood
    ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x ?y)
    :precondition (and (PERSON ?x) (WOOD ?y) (not(person-holds-something ?x)) (free-wood ?y))
    :effect (and (not(free-wood ?y)) (person-holds-wood ?x ?y) (person-holds-something ?x))
  )

  ; Person x1 can paint wood y with a tool z if person x1 is holding z, if z can be used for
  ; As a result the wood y becomes painted
  ; Parameters:
  ; - x1, x2: the people
  ; - y: the wood
  ; - z: the tool
  (:action paint
     ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x1 ?x2 ?y ?z)
    :precondition (and (PERSON ?x1) (PERSON ?x2) (WOOD ?y) (TOOL ?z) (CAN-PAINT ?z) (person-h
    :effect (wood-painted ?y)
  )

  ; Person x1 can nail wood y with a tool z if x1 is holdind z, if z can be used for hitting,
  ; As a result the wood y has a nail but is still not approved
  ; Parameters:
  ; - x1, x2: the people
  ; - y: the wood
  ; - z: the tool
  (:action nail
     ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x1 ?x2 ?y ?z)
    :precondition (and (PERSON ?x1) (PERSON ?x2) (WOOD ?y) (TOOL ?z) (CAN-HIT ?z) (person-hol
    :effect (and (wood-nailed ?y) (not(approved ?y)))
  )

  ; Person x can inspect wood y if y has been nailed and if person x is holding the wood y
  ; As a result wood y becomes approved
  ; Parameters:
  ; - x: the person
  ; - y: the wood
  (:action approve
    ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x ?y)
    :precondition (and (PERSON ?x) (WOOD ?y) (person-holds-wood ?x ?y) (wood-nailed ?y) )
    :effect (approved ?y)
  )

  ; 1.2. D Level: No solution Problem
  ; Question: Discuss briefly (in 1-3 lines of text) why the problem cannot be solved
```

```
  ; and define one new action in the domain that will allow the problem to get solved.

  ; Answer: In the no-solution problem there is equal number of people, as the number of woo
  ; as opposed to the other 3 problems. However, this leads to a problem since there is no
  ; action to drop the wood and thus no available people to nail or paint the last wood.
  ; To overcome this, we need a drop wood action.

  ; Person x can drop a wood y if they hold the wood
  ; As a result the person stops holding the wood, and the wood becomes free
  ; Parameters:
  ; - x: the person
  ; - y: the wood
  (:action drop-wood
     ; WRITE HERE THE CODE FOR THIS ACTION
    :parameters (?x ?y)
    :precondition (and (PERSON ?x) (WOOD ?y) (person-holds-wood ?x ?y))
    :effect (and (free-wood ?y) (not(person-holds-wood ?x ?y)) (not(person-holds-something ?
  )

)
```

1.2 D - Level: Unsolvable problem

See 1.1.

1.3 C - Level: Evaluate the heuristics of the problems

Since there we don't take into account the preconditions for the $h1$, then the cost will be equal to the number of "approve" actions for each problem. Thus, the cost will be 1 for the 1st problem, 2 for the 2nd, and 3 for the 3rd.

For the $h2$ heuristic, the cost will be infinite since the deleted lists are ignored and thus the people in each of the problem, will try to pick an object before dropping the previous one, leading to a dead end.

2 Company

2.1 E - Level: Infer $Floor(Frank) = 12$ using a sequence of Generalized Modus Ponens

From $S5$ and $S7$ we get:

$$\frac{S5: ReportsTo(Frank, Lily),\ S7: (ReportsTo(x, Lily) \Rightarrow Floor(x) = 12)}{S8: Floor(Frank) = 12},\ where\ \theta = \{x/Frank\}$$

2.2 E - Level: Translate the sentences to CNF

To translate the $S1 - S7$ sentences to CNF, we must make use of propositional logic equivalences:

$$Implication\ elimination: (\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \tag{1}$$

$$DeMorgan: \neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \tag{2}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \tag{3}$$

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

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

C2: $Position(Frank) = manager$

C3: $\neg[ReportsTo(x,y) \wedge Floor(x) = 12] \vee Floor(y) = 3$
$\Rightarrow$
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: $\neg ReportsTo(x,Lily) \vee Floor(x) = 12$

C8: $Floor(Frank) = 12 \quad (from 2.1)$

2.3 E - Level: Infer $ReportsTo(Rose,Frank)$ using Resolution

First resolution:
By combining C1 and C5,

$$l1\ is\ ReportsTo(Frank,Lily)\ and\ m1\ is\ \neg ReportsTo(x,y).$$
$$UNIFY(l1,\neg m1) = \theta = \{x/Frank,\ y/Lily\}$$
$$\frac{C1 : \neg ReportsTo(x,y) \vee \neg Floor(y) = 3 \vee ReportsTo(Rose,x),\ C5 : ReportsTo(Frank,Lily)}{C9 : \neg Floor(Lily) = 3 \vee ReportsTo(Rose,Frank)},$$
$$where\ \theta = \{x/Frank,\ y/Lily\}$$

Second resolution:
By combining C3 and C5,

$$l2\ is\ ReportsTo(Frank,Lily)\ and\ m2\ is\ \neg ReportsTo(x,y).$$
$$UNIFY(l2,\neg m2) = \theta = \{x/Frank,\ y/Lily\}$$
$$\frac{C3 : \neg ReportsTo(x,y) \vee \neg Floor(x) = 12 \vee Floor(y) = 3,\ C5 : ReportsTo(Frank,Lily)}{C10 : \neg Floor(Frank) = 12 \vee Floor(Lily) = 3},$$
$$where\ \theta = \{x/Frank,\ y/Lily\}$$

Third resolution:
By combining C8 and C10,

$$l3\ is\ Floor(Frank) = 12\ and\ m3\ is\ \neg Floor(Frank) = 12.$$
$$UNIFY(l3,\neg m3) = \theta = \{Frank\}$$
$$\frac{C8 : Floor(Frank) = 12,\ C10 : \neg Floor(Frank) = 12 \vee Floor(Lily) = 3}{C11 : Floor(Lily) = 3},$$
$$where\ \theta = \{Frank\}$$

Fourth resolution:

By combining C8 and C10,

$$l4 \text{ is } Floor(Lily) = 3 \text{ and } m4 \text{ is } \neg Floor(Lily) = 3.$$
$$UNIFY(l4, \neg m4) = \theta = \{Lily\}$$
$$\frac{C9 : \neg Floor(Lily) = 3 \vee ReportsTo(Rose, Frank),\ C11 : Floor(Lily) = 3}{C12 : ReportsTo(Rose, Frank)},$$
$$where\ \theta = \{Lily\}$$

2.4 D - Level: Express sentences in natural English

$S1$: If a member reports to another member that stays in the third floor, then Rose only reports to the former member.

$S2$: Frank's position is manager.

$S3$: If a member stays in the twelfth floor, then he/she only reports to a member that stays in the third floor.

$S4$: Harry stays in the ninth floor.

$S5$: Frank only reports to Lily.

$S6$: The member that stays in the third floor only reports to Harry.

$S7$: The member that reports to Lily, stays at the twelfth floor.

The $S8 : \neg Position(x) = Manager \vee \neg ReportsTo(x,y) \vee Position(y) = Director$,
can be translated to:
$\neg[Position(x) = Manager \wedge ReportsTo(x,y)] \vee Position(y) = Director$
$\Rightarrow$
Position(x)= Manager $\wedge ReportsTo(x,y) \Rightarrow Position(y) = Director$
Thus,
$S8$: The manager only reports to director.

2.5 C - Level: Construct an example of two grounded clauses that can be resolved together in two different ways resulting in A∨¬A and B∨¬B. Show the two different resolutions.

Consider the clauses:

C1: $SkyColour(Blue) \vee SkyColour(Red)$
C2: $\neg SkyColour(Blue) \vee \neg SkyColour(Red)$
By combining C1 and C2 we get A $\vee \neg A$,

$$l1 \text{ is } SkyColour(Red) \text{ and } m1 \text{ is } \neg SkyColour(Red).$$
$$UNIFY(l1, \neg m1) = \theta = \{Red\}$$
$$\frac{C1 : SkyColour(Blue) \vee SkyColour(Red),\ C2 : \neg SkyColour(Blue) \vee \neg SkyColour(Red)}{C3 : SkyColour(Blue) \vee \neg SkyColour(Blue)},$$
$$where\ \theta = \{Red\}$$

Alternatively to get B $\vee \neg B$:

$$l2 \text{ is } SkyColour(Blue) \text{ and } m2 \text{ is } \neg SkyColour(Blue).$$
$$UNIFY(l2, \neg m2) = \theta = \{Blue\}$$
$$\frac{C1 : SkyColour(Blue) \vee SkyColour(Red),\ C2 : \neg SkyColour(Blue) \vee \neg SkyColour(Red)}{C3 : SkyColour(Red) \vee \neg SkyColour(Red)},$$
$$where\ \theta = \{Blue\}$$

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.

Consider the clauses:

C1: $SkyColour(Blue)$
C2: $\neg Weather(Bad)$
C3: $SkyColour(Blue) \Rightarrow Wear(Glasses)$
C4: $\neg Wear(Glasses) \Rightarrow Weather(Bad)$
To turn C3 and C4 to CNF:
C3: $\neg SkyColour(Blue) \vee Wear(Glasses)$
C4: $\neg Weather(Bad) \Rightarrow Wear(Glasses)$
C4: $Weather(Bad) \vee Wear(Glasses)$

By using Resolution we reach the same solution:

$$l1 \text{ is } SkyColour(Blue) \text{ and } m1 \text{ is } \neg SkyColour(Blue).$$
$$UNIFY(l1, \neg m1) = \theta = \{Blue\}$$
$$\frac{C1 : SkyColour(Blue),\ C3 : \neg SkyColour(Blue) \vee Wear(Glasses)}{C5 : Wear(Glasses)},$$
$$\text{where } \theta = \{Blue\}$$

$$l2 \text{ is } Weather(Bad) \text{ and } m2 \text{ is } \neg Weather(Bad).$$
$$UNIFY(l2, \neg m2) = \theta = \{Bad\}$$
$$\frac{C2 : \neg Weather(Bad),\ C4 : Weather(Bad) \vee Wear(Glasses)}{C6 : Wear(Glasses)},$$
$$\text{where } \theta = \{Bad\}$$

2.7 B - Level: Solve the company puzzle using SWISH

As submitted in A3.PP2.SWISH:

To translate the $S8 - S10$ sentences from CNF to logical implications, we must make use of the propositional logical equivalences (1-3).

We already translated S8 in 2.4 to make it easier express it in natural English. Now we do the same for S9 and S10.

S9: $\neg Position(x) = Manager \vee (ReportsTo(y,x) \Rightarrow Position(y) = Intern)$.
$\Rightarrow$
S9: $Position(x) = Manager \Rightarrow (ReportsTo(y,x) \Rightarrow Position(y) = Intern)$.
And,
S10: $\neg Position(x) = Director \vee (ReportsTo(x,y) \Rightarrow Position(y) = CEO)$.
$\Rightarrow$
S10: $Position(x) = Director \Rightarrow (ReportsTo(x,y) \Rightarrow Position(y) = CEO)$.
Therefore:

einstein.txt

```
% company(Company).
%   @param  Solution is a list of members that satisfy all constraints.
```

```
/* Company logical puzzle:

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
Rose; Lily; Frank; Harry; 3 ; 5 ; 9 ; 12 ; intern; manager; director;CEO;

functions
Floor(x); Position(x);

and predicate
ReportsTo(x; y)
meaning that x reports to y. We know the following facts about them.

S1: ReportsTo(x, y) ^ Floor(y) = 3 => ReportsTo(Rose, x)
S2: Position(Frank) = manager
S3: ReportsTo(x, y) ^ Floor(x) = 12 => Floor(y) = 3
S4: Floor(Harry) = 9
S5: ReportsTo(Frank, Lily)
S6: Floor(x) = 3 => ReportsTo(x, Harry)
S7: ReportsTo(x, Lily) => Floor(x) = 12
S8: \negPosition(x) = Manager OR \negReportsTo(x, y) OR Position(y) = Director
S9: \negPosition(x) = Manager OR \negReportsTo(y, x) OR Position(y) = Intern
S10: \negPosition(x) = Director OR \negReportsTo(x, y) OR Position(y) = CEO
S11: Floor(Rose) = 5

Solve the puzzle
*/

% Render the members term as a nice table.
:- use_rendering(table,
                 [header(person(’Name’, ’Position’, ’Floor’))]).

company(Com) :-
        % each member in the list Com of company is represented as:
        %      person(Name, Position, Floor)
        length(Com, 4),

    % S1
    reportsto(X1, Y1, Com), floor(three, Y1,Com), reportsto(person(rose,_,_), X1, Com),
    % S2
    member(person(frank,manager,_), Com),
    % S3
    reportsto(X3, Y3, Com), floor(twelve, X3,Com), floor(three, Y3,Com),
    % S4
    member(person(harry,_,nine), Com),
    % S5
        reportsto(person(frank,_,_), person(lily,_,_), Com),
    % S6
    floor(three, X6,Com), reportsto(X6, person(harry,_,_), Com),
    % S7
    reportsto(X7, person(lily,_,_), Com), floor(twelve, X7,Com),
    % S8
    position(manager, X8, Com), reportsto(X8, Y8, Com), position(director, Y8, Com),
    % S9
    position(manager, X9, Com), reportsto(Y9, X9, Com), position(intern, Y9, Com),
    % S10
    position(director, X10, Com), reportsto(X10, Y10, Com), position(ceo, Y10, Com),
    % S11
    member(person(rose,_,five), Com).

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, _))
    ).

floor(M, X, Ls) :-
    member(X, Ls),
    X = person(_, _, M).

position(N, X, Ls) :-
    member(X,Ls),
    X=person(_,N,_).

/** <examples>
?- company(Com).
*/
```

2.8 A - Level: Consider the following claim: "Two grounded clauses cannot be solved together in two different ways and result in A ∨ B and ¬ A ∨ ¬ 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.

In order to reach the $A \vee B$ and $\neg A \vee \neg B$ clauses, we will make use of a third literal C, and the resolution method.

- C1: $A \vee B \vee C$, and C2: $\neg C$

- C1: $A \vee C$, and C2: $B \vee \neg C$

- C1: $A \vee \neg C$, and C2: $B \vee C$

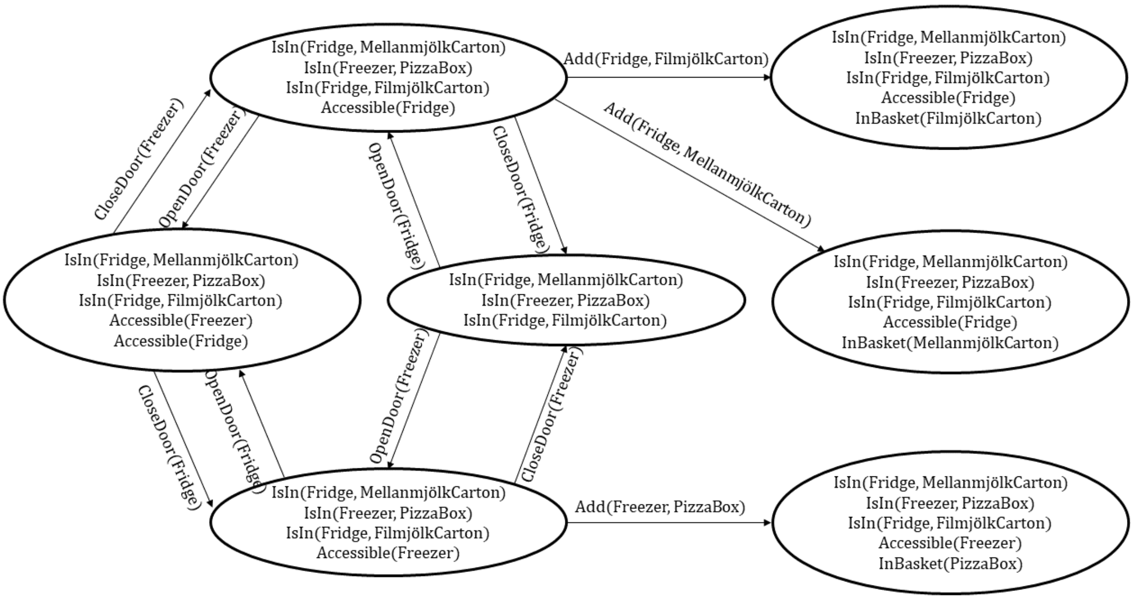
- C1: C, and C2: $A \vee B \vee \neg C$

From this, one can notice that there is no way to solve any of these clauses, and result in a resolvent $\neg A \vee \neg B$, no matter what the C literal is.

3 Grocery Shopping

3.1 E - Level: Construction of the state space

3.1.1 Convert the planning problem to planning in a state space


IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
Add(Fridge, FilmjölkCarton)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
InBasket(FilmjölkCarton)
Add(Fridge, MellanmjölkCarton)
CloseDoor(Freezer)
OpenDoor(Freezer)
OpenDoor(Fridge)
CloseDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Freezer)
Accessible(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
InBasket(MellanmjölkCarton)
CloseDoor(Fridge)
OpenDoor(Fridge)
OpenDoor(Freezer)
CloseDoor(Freezer)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Freezer)
Add(Freezer, PizzaBox)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Freezer, PizzaBox)
IsIn(Fridge, FilmjölkCarton)
Accessible(Freezer)
InBasket(PizzaBox)

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

For each of the three objects, MellanmjölkCarton, FilmjölkCarton and PizzaBox there are 3 different states: be in the fridge for the cartons/freezer for the box, be in the basket, or have it. Thus $3x3x3 = 27$ possible states. Furthermore, the Fridge and the Freezer can both be either Accessible or not. Thus $2x2 = 4$ possible states. Thus, in total, there are $27x4 = 108$ possible states in the whole reachable world.

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

The goal conditions can be satisfied in 2 states, either when 1. I have the milk, the pizza and the sour milk is in the fridge, or 2. I have the milk, the pizza and the sour milk is in the basket. Furthermore, the Fridge and the Freezer can both be either Accessible or not. Thus $2x2 = 4$ possible states. Thus, in total, there are $2x4 = 8$ different goal states, satisfying the goal condition 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?

Since there is no restriction on how many times we can open and close the door of the fridge or the freezer, we can conclude that there are infinite number of possible plans that can lead to the satisfaction of 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.

The number of the actions of the optimal plan is 6 (considering that we don't really need to close the door, since it is not mentioned in the goal state), however there are multiple reachable optimal plans we can follow to reach the goal. An example of a sequence of ground actions of an optimal plan is:

1. OpenDoor(Fridge)

2. Add(Fridge,MellanmjölkCarton)
3. OpenDoor(Freezer)
4. Add(Freezer,PizzaBox)
5. Pay(PizzaBox,Pizza)
6. Pay(MellanmjölkCarton,Milk)

3.2 D - Level: Alternative state space properties

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

There are 4 possible combinations of states, considering whether a door had opened or not:

- We don't open any door: There is only 1 state since we didn't put anything in the basket, thus, we don't have any of the objects.
- We open the freezer door, but not the fridge door: Then we can either leave the door open, or close it, thus 2 states. And then there is the probability that the pizzabox is either in the freezer, or in the basket, or we have it, thus 3 states. Thus, in total, there are $2x3 = 6$ states.
- We open the fridge door, but not the freezer door: Then we can either leave the door open, or close it, thus 2 states. And then there is the probability that the milk is either in the MellanmjölkCarton or the FilmjölkCarton, and the carton is either in the fridge, or in the basket, or we have it, thus $3^2 = 9$ states. Thus, in total, there are $2x9 = 18$ states.
- We open the fridge door and the freezer door: Then we have all the 108 states combinations we had in 3.1.2..

Therefore, overall, we have $1+6+18+108 = 133$ different states 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?

The goal states in that case are the same as in 3.1.3.:

The goal conditions can be satisfied in 2 states, either when 1. I have the milk, the pizza and the sour milk is in the fridge, or 2. I have the milk, the pizza and the sour milk is in the basket. Furthermore, the Fridge and the Freezer can both be either Accessible or not. Thus $2x2 = 4$ possible states. Thus, in total, there are $2x4 = 8$ different goal states, satisfying the goal condition 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?

In this case, there is no infinite number of different plans that lead to the goal, since we can only open a door once. Then, the minimum number of actions to reach the goal, is 6, as stated in 3.1.5., and the maximum number of actions is 9, if we also add the unnecessary actions of *a*. closing the fridge door, *b*. closing the freezer door and *c*. also adding the FilmjölkCarton in the basket.

Thus, we have to find all the possible sequence combinations for the 6-, 7-, 8-, or 9-action sequence plan.

This is a tedious task since no all the sequence ordering can be performed, because for example we cannot add an object in the basket, if we don't open a door first.

Therefore, to find the number of different plans, we must exclude from the number of all the possible sequence combinations $x!$, $where\ x = 6,\ 7,\ 8,\ 9$, the sequences that can't be performed due to restrictions. For this, instead of calculating the exact number of all possible different plans, I prefer to just mention the restrictions (constrains) we must satisfy in order to find the solution. Then, we could use these restrictions to find the answer using a coding language.

Thus, the restrictions are:

- Opening fridge/freezer door must precedes adding a corresponding object.
- Adding must precedes paying the object.
- Closing door must come after adding the corresponding object.

Taking this into account, many possible different plans can be found, using 6-, 7-, 8-, or a 9-action sequence.

3.3 D - Level: Yet another alternative state space properties

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

For this case we follow the exact same approach as we did in 3.2.1., with the difference that now, another one possible state is being added. The state that we have an object, and it is also in the basket, at the same time, since the objects are ”infinite”. Then, there are 4 possible combinations of states, considering whether a door had opened or not:

- We don't open any door: There is only 1 state since we didn't put anything in the basket, thus, we don't have any of the objects.
- We open the freezer door, but not the fridge door: Then we can either leave the door open, or close it, thus 2 states. And then there is the probability that the pizzabox is either in the freezer, or in the basket, or we have it, or both have it and in the basket, thus 4 states. Thus, in total, there are $2x4 = 8$ states.
- We open the fridge door, but not the freezer door: Then we can either leave the door open, or close it, thus 2 states. And then there is the probability that the milk is either in the MellanmjölkCarton or the FilmjölkCarton, and the carton is either in the fridge, or in the basket, or we have it, or both have it and in the basket, thus $4^2 = 16$ states. Thus, in total, there are $2x16 = 32$ states.
- We open the fridge door and the freezer door: Then, similar to 3.1.2. we have $4x4^3 = 256$ possible states combinations.

Therefore, overall, we have $1+8+32+256 = 297$ different states in the whole reachable state space.

Assumption: That I only pay for an object, only after I bought all of its kind. Otherwise it would result in infinite possible reachable states.

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

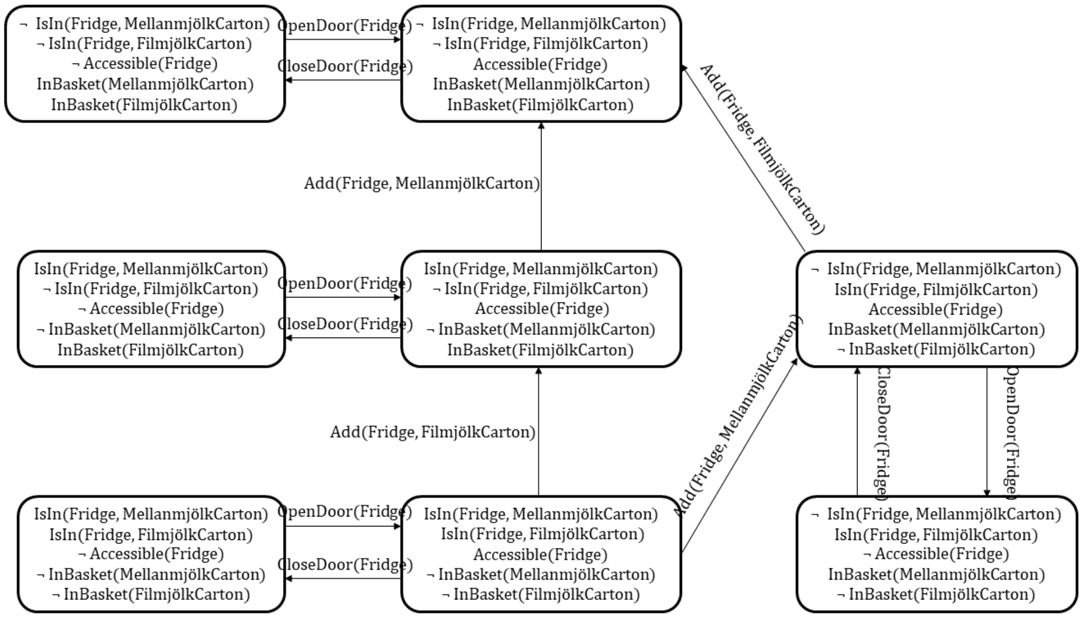
The goal conditions are to have milk and pizza. Both of them can have 2 states: 1. I have it and is not in the basket, or 2. I have it and is in the basket. Also, I can have the sour milk or not and be in the basket or not. Thus $2x2 = 4$ possible sour milk states. Furthermore, the Fridge and the Freezer can both be either Accessible or not. Thus $2x2 = 4$ possible states. Therefore, in total, there are $2x2x4x4 = 64$ different goal states, satisfying the goal condition 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?

Since there is no restriction on how many times we can add (and pay) an object, we can conclude that there are infinite number of possible plans that can lead to the satisfaction of the goal condition.

3.4 C - Level: Construction of the belief state space

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


¬ IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
OpenDoor(Fridge)
CloseDoor(Fridge)
¬ IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
Add(Fridge, FilmjölkCarton)
Add(Fridge, MellanmjölkCarton)
IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
OpenDoor(Fridge)
CloseDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
¬ IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
Add(Fridge, FilmjölkCarton)
Add(Fridge, MellanmjölkCarton)
CloseDoor(Fridge)
OpenDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
OpenDoor(Fridge)
CloseDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
¬ IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)

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

Assuming that ONLY $Contains(FilmjolkCarton, SoyaMilk)$ and $Contains(MellanmjolkCarton, Milk)$ are unknown fluents, then there are 4 physical states:

- $Contains(FilmjolkCarton,\ SoyaMilk) \wedge\ Contains(MellanmjolkCarton,\ Milk)$
- $\neg\ Contains(FilmjolkCarton,\ SoyaMilk) \wedge\ Contains(MellanmjolkCarton,\ Milk)$
- $Contains(FilmjolkCarton,\ SoyaMilk) \wedge \neg\ Contains(MellanmjolkCarton,\ Milk)$
- $\neg\ Contains(FilmjolkCarton,\ SoyaMilk) \wedge \neg\ Contains(MellanmjolkCarton,\ Milk)$

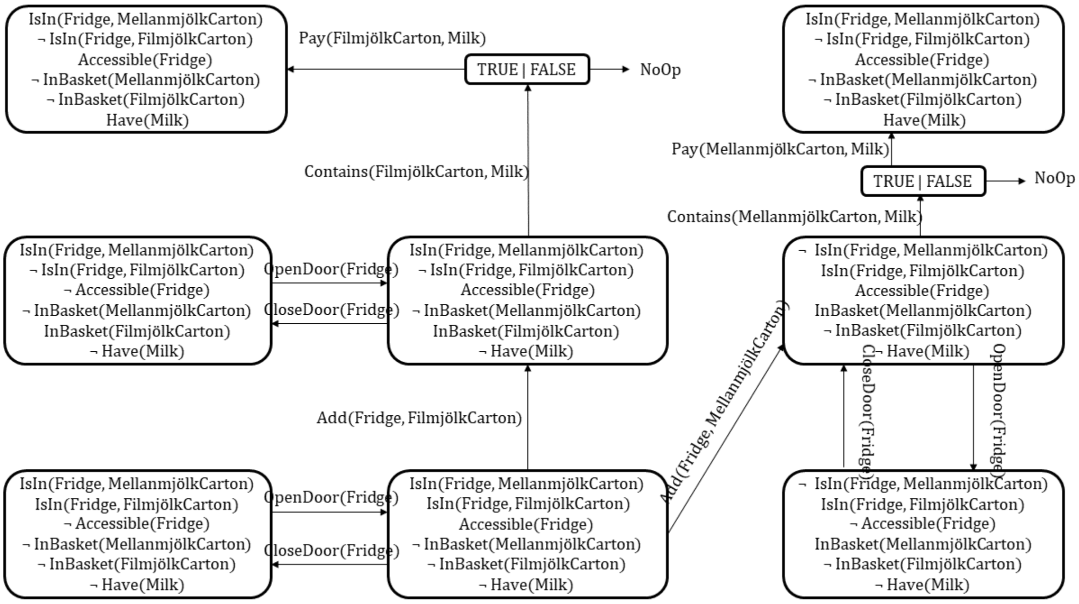
3.4.3 How many different plans that lead to the satisfaction of the goal are there?

Since we are working in a belief state, there is no specific plan that is 100% able to reach the goal. But even if we had such a plan, there is 50% probability that we can actually reach the goal, by following the physical states where MellanmjolkCarton contains milk. But even then, there can be infinite number of them since we can open and close the door as many times as we want, as mentioned in 3.1.4..

3.5 B - Level: What if we add perception to our problem?

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

Assuming that there are two transitions $Contains(FilmjölkCarton, Milk)$ and $Contains(MellanmjolkCarton, Milk)$ this is a possible subgraph. Also, the True/False statements, indicate the fluent of perception.


IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
Have(Milk)
Pay(FilmjölkCarton, Milk)
TRUE | FALSE
NoOp
Contains(FilmjölkCarton, Milk)
IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
Have(Milk)
Pay(MellanmjölkCarton, Milk)
TRUE | FALSE
NoOp
Contains(MellanmjölkCarton, Milk)
IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
¬ Have(Milk)
OpenDoor(Fridge)
CloseDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
¬ IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
InBasket(FilmjölkCarton)
¬ Have(Milk)
¬ IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
¬ Have(Milk)
Add(Fridge, FilmjölkCarton)
Add(Fridge, MellanmjölkCarton)
CloseDoor(Fridge)
OpenDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
¬ Have(Milk)
OpenDoor(Fridge)
CloseDoor(Fridge)
IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
Accessible(Fridge)
¬ InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
¬ Have(Milk)
¬ IsIn(Fridge, MellanmjölkCarton)
IsIn(Fridge, FilmjölkCarton)
¬ Accessible(Fridge)
InBasket(MellanmjölkCarton)
¬ InBasket(FilmjölkCarton)
¬ Have(Milk)

3.5.2 Does there exist a contingent plan achieving the goal?

The definition of the problem implies that there is a possibility that not any carton contains milk. However, in case milk is contained in a milk, a contingent plan would be:

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

4 Robot

4.1 B - Level: Evaluate the heuristics of the problem

- What is the value of h1 for the initial state?

  The H1 heuristic function ignores the preconditions of all actions in the action scheme. This results in every action becoming applicable in every state. Thus, we don't need the "Move" nor the "Pick" actions since the "Drop" action is applicable no matter where the robot, the object, or the drop place are. And neither do we need to grab an object in order to change its position. Then, we only need to perform the "Drop" action once for each object, and the heuristic value in this case will be 3.

- What is the value of h2 for the initial state?

  By ignoring the deleted lists no action will undo the progress of another action towards the goal. In our case, the robot will always be in all the places it already visited simultaneously and will always be free. This means that it doesn't need to drop an object before picking another one, and that it doesn't need to move in order to go to the drop place. Considering these, the heuristic value will be 22: a pick and a drop per object plus 16 total moves to reach the goal.

- Are h1 and h2 admissible?

  Yes, both are admissible since they reach the goal in less moves than what is actually needed (they don't overestimate).

- Compare heuristics h1 and h2. Does one of them dominate the other?

  In order for a heuristic to dominate another, it must result in higher heuristic value for every state. However this is not the case for this problem. It could happen that the robot needs to move and drop an object for the h1, whereas would just need to drop the object for h2 (if for example it already visited C1,1.

- Compare heuristics H1 and H2 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.

  I already showed in the previous question that H1 cannot dominate H2, let alone in the general case. For the first case, the H1 will be very close to the goal-go-cost whereas H2 not necessarily. Then H1 will most probably be higher than H2. For the second case, H2 will most probably be dominant to H1, since ignoring preconditions will lead to the solution directly.

4.2 B - Level: Answer the questions

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

  Dead end are states if the goal distance is infinity. Thus, if there is no such a path that can lead to the goal. In this problem, there shouldn't be dead end, since we can always find a possible plan to reach it. But even if there was a dead end, then the answer for h1 would still be no, because we would ignore all the negative literals in the effects of the actions, and the preconditions, for the robot problem are positive fluents.

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

  Given what I said about the dead ends, given that there is a dead end in this problem, then the answer for h2 would be yes. By using the H2, the robot is probable to move from the first object, to pick up the second, without dropping the first one. Perhaps the robot won't be able to do that, resulting in an unrecognized dead end.

4.3 A - Level: Create your own heuristic function for the problem

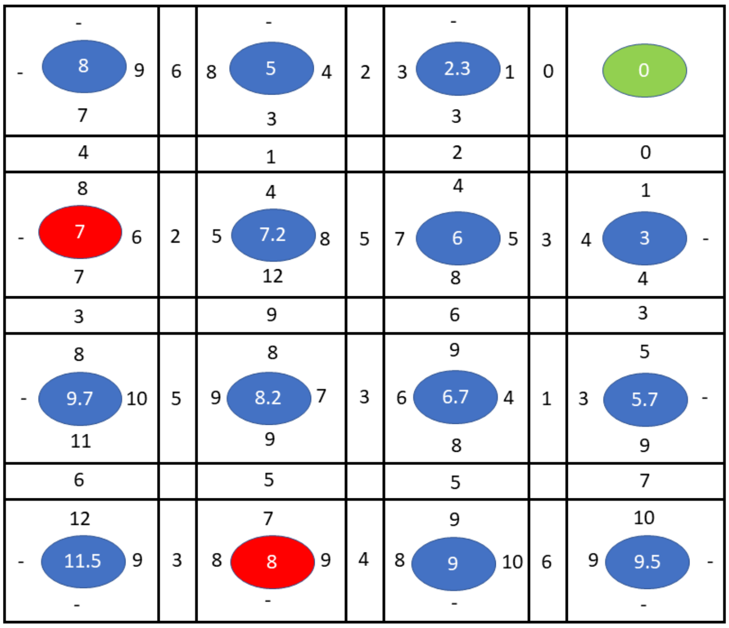
1. Make your own heuristic function. Assign heuristic values to each cell such that there are local minima in the state space. The heuristic should return zero if and only if the goal is reached.

   In order to create a grid with local minima, that means it shouldn't satisfy a strictly increasing / decreasing function. This implies that the gradient changes non-homogeneously according to the location of the cell, forcing it to go uphill or downhill.

   We assume the heuristic:

   heuristic = [distance to goal] - [move cost].

   Thus, the cost of each cell will be the average of all the possible paths that lead to this cell. This is shown in the next figure.



As shown in the figure, there are two local minima at the cells (2,1) and (1,3) (considering that the left-down cell is 1,1). However, the global minima is the (4,4) cell as it should be.

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

   The maximal local minimum exit distance, $mlmed(S,T)$, of a state space (S,T) is the maximum over the exit distances of all states on local minima.

   The exit distance $ed(s)$ of a search state s is the distance to the nearest exit.

   Thus, considering that exit is the goal, the exit distance of the (2,1) would be 5 whereas for the (1,3) cell would be 4. Then, the $mlmed$ in that case is the value for the (2,1) local minima cell, $mlmed = 5$.

4.4 A - Level: Answer the questions for the modified problem

The only way of having a dead end in this case, would be if the robot enters the forest without a flashlight. This is not an option since the robot always starts from the first cell, picks the flashlight, and never drops it. So I will assume that there is a possibility of dropping the flashlight. For the case of H1, this could be done since it doesn't check the preconditions. However, for H2, this is not possible since it will always be in C1,1, thus having the flashlight. Therefore,

1. Does there exist a reachable recognized dead end for H1?
   No, because ignoring the preconditions of the actions doesn't affect the result. We still can apply any action at any state. Since the robot carries with it the flashlight due to the initial condition, it won't be affected when entering the forest.

2. Does there exist a reachable unrecognized dead end for H1?
   Yes because we don't take into account the precondition of the "Have(Flashlight)", thus the robot may stuck in the forest.

3. Does there exist a reachable recognized dead end for H2?
   No because even if we reach the dead end (forest without a flashlight), the robot will still thinks that has the flashlight.

4. Does there exist a reachable unrecognized dead end for H2?
   Yes because the robot may try to pick more than two objects at a time, and because may try to drop the object when it doesn't hold one. This is because of ignoring the Hold negation of the "Drop" action.

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?

In that case it really depends. In order to reach a reachable unrecognized dead end, that means that the robot reached a state in which it stucked, so no solution can be found. However, as we saw in the forest example, there is a high probability that the dead ends are anyway unreachable (for example the robot cannot enter the forest if it doesn't have a flashlight. So it cannot stuck inside there). But if for any reason, the dead end is reachable (this is the keyword of the question) then there is no way to escape it, thus find a solution, since the brain of our algorithm thinks it found a solution, even though it didn't really.