1 Beekeeper

See the zipped files.

I believe I have programmed correctly, however the provided online solver, and PDDL4 do not solve the problem with a minimal total cost, so I was not able to verify that the written code was indeed correct.

2 Einstein puzzle with a twist

2.1 Constructing sentences

S1 $House(Bob) = Red$

S2 $Drink(Alice) = Snaps$

S3 $Fear(Ted) = Elevators$

S4 $Roommate(Carol, Ted)$

S5 $\forall x\ Fear(x) = Elevators \Rightarrow House(x) = Blue$

S6 $\forall x\ House(x) = Red \Rightarrow Fear(x) = Spiders$

S7 $\forall x\ Drink(x) = Snaps \Rightarrow House(x) = Red$

S8 $\forall x\ Neighbor(Bob, x) \Rightarrow Drink(x) = Beer$

S9 $\forall x, y\ Roommate(x, y) \wedge Fear(y) = Elevators \Rightarrow Fear(x) = Spiders$

S10 $\forall x, y\ Neighbor(x, y) \wedge Drink(x) = Milk \Rightarrow Music(y) = Beatles$

S11 $\forall x\ Roomate(Alice, x) \wedge Fear(x) = Spider \Rightarrow Drink(x) = Milk$

S12 $\forall x, y\ Roommate(x, y) \wedge Drink(x) = Snaps \Rightarrow Music(y) = Beatles$

S13 $\forall x\ Neighbor(Ted, x) \wedge Drink(x) = Snaps \wedge Fear(x) = Spiders \Rightarrow Music(x) = ABBA$

2.2 Inference using generalized modus ponens

Letting $S3$, $S4$ be p_1', p_2' and splitting $S9$ into $p_1 = Roommate(x, y) \wedge Fear(y) = Elevators$ and $q = Fear(x) = Spiders$ we get

$$\frac{Fear(Ted) = Elevators,\ Roommate(Carol, Ted) \quad (Roommate(x, y) \wedge Fear(y) = Elevators \Rightarrow Fear(x) = Spiders)}{Fear(Carol) = Spiders} \tag{1}$$

where $\theta = \{y/Ted,\ x/Carol\}$

2.3 Introducing CNF

C1 $House(Bob) = Red$
C2 $Drink(Alice) = Snaps$
C3 $Fear(Ted) = Elevators$
C4 $Roommates(Carol, Ted)$
C5 $\neg Fear(x) = Elevators \vee House(x) = Blue$
C6 $\neg House(x) = Red \vee Fear(x) = Spiders$
C7 $\neg Drink(x) = Snaps \vee House(x) = Red$
C8 $\neg Neighbor(Bob, x) \vee Drink(x) = Beer$
C9 $\neg Roommates(x, y) \vee \neg Fear(y) = Elevators \vee Fear(x) = Spiders$
C10 $\neg Neighbors(x, y) \vee \neg Drink(x) = Milk \vee Music(y) = Beatles$
C11 $\neg Roommates(Alice, x) \vee \neg Fear(x) = Spider \vee Drink(x) = Milk$
C12 $\neg Roommates(x, y) \vee \neg Drink(x) = Snaps \vee Music(y) = Beatles$
C13 $\neg Neighbors(Ted, x) \vee \neg Drink(x) = Snaps \vee \neg Fear(x) = Spiders \vee Music(x) = ABBA$

Following the $HaveCake/BakedCake$ example in the lecture slides.

2.4 Inference of music

C14 $\neg Roommates(x, y) \vee \neg House(x) = z \vee House(y) = z$

If two persons are roomates, they live in the same house.
(Or more specifially if one lives in a house, the other live in the same house)

C15 $\neg House(x) = z \vee \neg House(y) = z \vee Roommates(x, y)$

If two persons live in the same house, they are roommates.

C16 $\neg Roommates(x, y) \vee Roomates(y, x)$

Roommates are mutually each others roommates.

C17 $\neg House(x) = Red \vee \neg House(y) = Blue \vee Neighbors(x, y)$

If two persons live in different houses (Red and Blue), they are neighbors.

C18 $\neg Neighbors(x, y) \vee Neighbors(y, x)$

Neighbors are mutually each others neighbors.

2.4.1 Inference with the resolution rule

$$\frac{\text{C2: } Drink(Alice) = Snaps, \quad \text{C7: } \neg Drink(x) = Snaps \vee House(x) = Red}{\text{C19: } House(Alice) = Red}, \quad \theta = \{x/Alice\}$$

$$\frac{\text{C19: } House(Alice) = Red, \quad \text{C15: } \neg House(x) = z \vee \neg House(y) = z \vee Roommates(x, y)}{\text{C20: } \neg House(y) = Red \vee Roommates(Alice, y)}, \quad \theta = \{x/Alice, z/Red\}$$

$$\frac{\text{C1: } House(Bob) = Red, \quad \text{C20: } \neg House(y) = Red \vee Roommates(Alice, y)}{\text{C21: } Roommates(Alice, Bob)}, \quad \theta = \{y/Bob\}$$

$$\frac{\text{C21: } Roommates(Alice, Bob), \quad \text{C12: } \neg Roommates(x, y) \vee \neg Drink(x) = Snaps \vee Music(y) = Beatles}{\text{C22: } \neg Drink(Alice) = Snaps \vee Music(Bob) = Beatles},$$

$\theta = \{x/Alice, y/Bob\}$

$$\frac{\text{C2: } Drink(Alice) = Snaps, \quad \text{C22: } \neg Drink(Alice) = Snaps \vee Music(Bob) = Beatles}{\text{C23: } Music(Bob) = Beatles}, \quad \theta = \emptyset$$

Therefore Bob likes the Beatles. Furthermore

$$\frac{\text{C19: } House(Alice) = Red, \quad \text{C17: } \neg House(x) = Red \vee \neg House(y) = Blue \vee Neighbors(x, y)}{\text{C24: } \neg House(y) = Blue \vee Neighbors(Alice, y)}, \quad \theta = \{x/Alice\}$$

$$\frac{\text{C4: } Roommates(Carol, Ted), \quad \text{C9: } \neg Roommates(x, y) \vee \neg Fear(y) = Elevators \vee Fear(x) = Spiders}{\text{C25: } \neg Fear(Ted) = Elevators \vee Fear(Carol) = Spiders},$$

$\theta = \{x/Carol, y/Ted\}$

$$\frac{\text{C3: } Fear(Ted) = Elevators, \quad \text{C25: } \neg Fear(Ted) = Elevators \vee Fear(Carol) = Spiders}{\text{C26: } Fear(Carol) = Spiders}, \quad \theta = \emptyset$$

$$\frac{\text{C26: } Fear(Ted) = Elevators, \quad \text{C5: } \neg Fear(x) = Elevators \vee House(x) = Blue}{\text{C27: } House(Ted) = Blue}, \quad \theta = \{x/Ted\}$$

$$\frac{\text{C4: } Roommates(Carol, Ted), \quad \text{C16: } \neg Roommates(x, y) \vee Roomates(y, x)}{\text{C28: } Roommates(Ted, Carol)}, \quad \theta = \{x/Carol, y/Ted\}$$

$$\frac{\text{C28: } Roommates(Ted, Carol), \quad \text{C14: } Roommates(x, y) \vee \neg House(x) = z \vee House(y) = z}{\text{C29: } \neg House(Ted) = z \vee House(Carol) = z},$$

$\theta = \{x/Ted, y/Carol\}$

$$\frac{\text{C27: } House(Ted) = Blue, \quad \text{C29: } \neg House(Ted) = z \vee House(Carol) = z}{\text{C30: } House(Carol) = Blue}, \quad \theta = \{z/Blue\}$$

$$\frac{\text{C30: } House(Ted) = Blue, \quad \text{C24: } \neg House(y) = Blue \vee Neighbors(Alice, y)}{\text{C31: } Neighbors(Alice, Ted)}, \quad \theta = \{y/Ted\}$$

$$\frac{\text{C19: } House(Alice) = Red, \quad \text{C9: } \neg House(x) = Red \vee Fear(x) = Spiders}{\text{C32: } Fear(Alice) = Spiders}, \quad \theta = \{x/Alice\}$$

$$\frac{\text{C31: } Neighbors(Alice, Ted), \quad \text{C18: } \neg Neighbors(x, y) \vee Neighbors(y, x)}{\text{C33: } Neighbors(Ted, Alice)}, \quad \theta = \{x/Alice, y/Ted\}$$

$$\frac{\text{C31: } Neighbors(Alice,Ted), \quad \text{C33: } \neg Neighbor(Ted,x) \vee \neg Drink(x) = Snaps \vee \neg Fear(x) = Spiders \vee Music(x) = ABBA}{\text{C34: } \vee \neg Drink(Alice) = Snaps \vee \neg Fear(Alice) = Spiders \vee Music(Alice) = ABBA},$$

$\theta = \{x/Alice\}$

$$\frac{\text{C2: } Drink(Alice) = Snaps, \quad \text{C34: } \vee \neg Drink(Alice) = Snaps \vee \neg Fear(Alice) = Spiders \vee Music(Alice) = ABBA}{\text{C35: } \neg Fear(Alice) = Spiders \vee Music(Alice) = ABBA},$$

$\theta = \emptyset$

$$\frac{\text{C32: } Fear(Alice) = Spiders, \quad \text{C35: } \neg Fear(Alice) = Spiders \vee Music(Alice) = ABBA}{\text{C36: } Music(Alice) = ABBA},$$

$\theta = \emptyset$

So Alice likes ABBA. Additionally:

$$\frac{\text{C1: } House(Bob) = Red, \quad \text{C6: } \neg House(x) = Red \vee Fear(x) = Spiders}{\text{C37: } Fear(Bob) = Spiders}, \quad \theta = \{x/Bob\}$$

$$\frac{\text{C1: } House(Bob) = Red \quad \text{C17: } \neg House(x) = Red \vee \neg House(y) = Blue \vee Neighbors(x, y)}{\text{C38: } \neg House(y) = Blue \vee Neighbors(Bob, y)}, \quad \theta = \{x/Bob\}$$

$$\frac{\text{C29: } House(Ted) = Blue, \quad \text{C38: } \neg House(y) = Blue \vee Neighbors(Bob, y)}{\text{C39: } Neighbors(Bob, Ted)}, \quad \theta = \{y/Ted\}$$

$$\frac{\text{C21: } Roomates(Alice, Bob), \quad \text{C11: } \neg Roommates(Alice, x) \vee \neg Fear(x) = Spider \vee Drink(x) = Milk}{\text{C40: } \neg Fear(Bob) = Spider \vee Drink(Bob) = Milk},$$

$\theta = \{x/Bob\}$

$$\frac{\text{C37: } Fear(Bob) = Spiders, \quad \text{C38: } \neg Fear(Bob) = Spider \vee Drink(Bob) = Milk}{\text{C41: } Drink(Bob) = Milk}, \quad \theta = \{x/Bob\}$$

$$\frac{\text{C39: } Neighbors(Bob, Ted), \quad \text{C10: } \neg Neighbors(x, y) \vee \neg Drink(x) = Milk \vee Music(y) = Beatles}{\text{C42: } \neg Drink(Bob) = Milk \vee Music(Ted) = Beatles},$$

$\theta = \{x/Bob, y/Ted\}$

$$\frac{\text{C41: } Drink(Bob) = Milk, \quad \text{C42: } \neg Drink(Bob) = Milk \vee Music(Ted) = Beatles}{\text{C43: } Music(Ted) = Beatles}, \quad \theta = \emptyset$$

So Ted listens to the Beatles. Lastly:

$$\frac{\text{C1: } House(Bob) = Red, \quad \text{C17: } \neg House(x) = Red \vee \neg House(y) = Blue \vee Neighbors(x, y)}{\text{C44: } \neg House(Carol) = Blue \vee Neighbors(Bob, Carol)},$$

$\theta = \{x/Bob, y/Carol\}$

$$\frac{\text{C26: } House(Carol) = Blue, \quad \text{C44: } \neg House(Carol) = Blue \vee Neighbors(Bob, Carol)}{\text{C45: } Neighbors(Bob, Carol)},$$

$\theta = \{x/Bob, y/Carol\}$

$$\frac{\text{C45: } Neighbors(Bob, Carol), \quad \text{C10: } \neg Neighbors(x, y) \vee \neg Drink(x) = Milk \vee Music(y) = Beatles}{\text{C46: } \neg Drink(Bob) = Milk \vee Music(Carol) = Beatles},$$

$\theta = \{x/Bob, y/Carol\}$

$$\frac{\text{C38: } Drink(Bob) = Milk, \quad \text{C46: } \vee \neg Drink(x) = Milk \vee Music(y) = Beatles}{\text{C47: } Music(Carol) = Beatles}, \quad \theta = \emptyset$$

So Carol is also a fan of the Beatles.
Summed up:

$$\begin{cases} Music(Ted) = Beatles \\ Music(Carol) = Beatles \\ Music(Bob) = Beatles \\ Music(Alice) = ABBA \end{cases}$$

Some resolutions above may have been unnecessary.

2.5 Resolving two clauses in two different ways

$$\frac{House(Ted) = Red \vee Fear(Alice) = Spiders, \quad \neg House(x) = Red \vee \neg Fear(x) = Spiders}{???} \tag{2}$$

Can be resolved in two different ways:

$$\frac{House(Ted) = Red \vee Fear(Alice) = Spiders, \quad \neg House(x) = Red \vee \neg Fear(x) = Spiders}{\neg Fear(Ted) = Spiders}, \theta = \{x/Ted\} \tag{3}$$

or

$$\frac{House(Ted) = Red \vee Fear(Alice) = Spiders, \quad \neg House(x) = Red \vee \neg Fear(x) = Spiders}{\neg House(Alice) = Red}, \theta = \{x/Alice\} \tag{4}$$

with toally different meanings.

3 Treasure hunt

3.1 State space

3.1.1 Graph

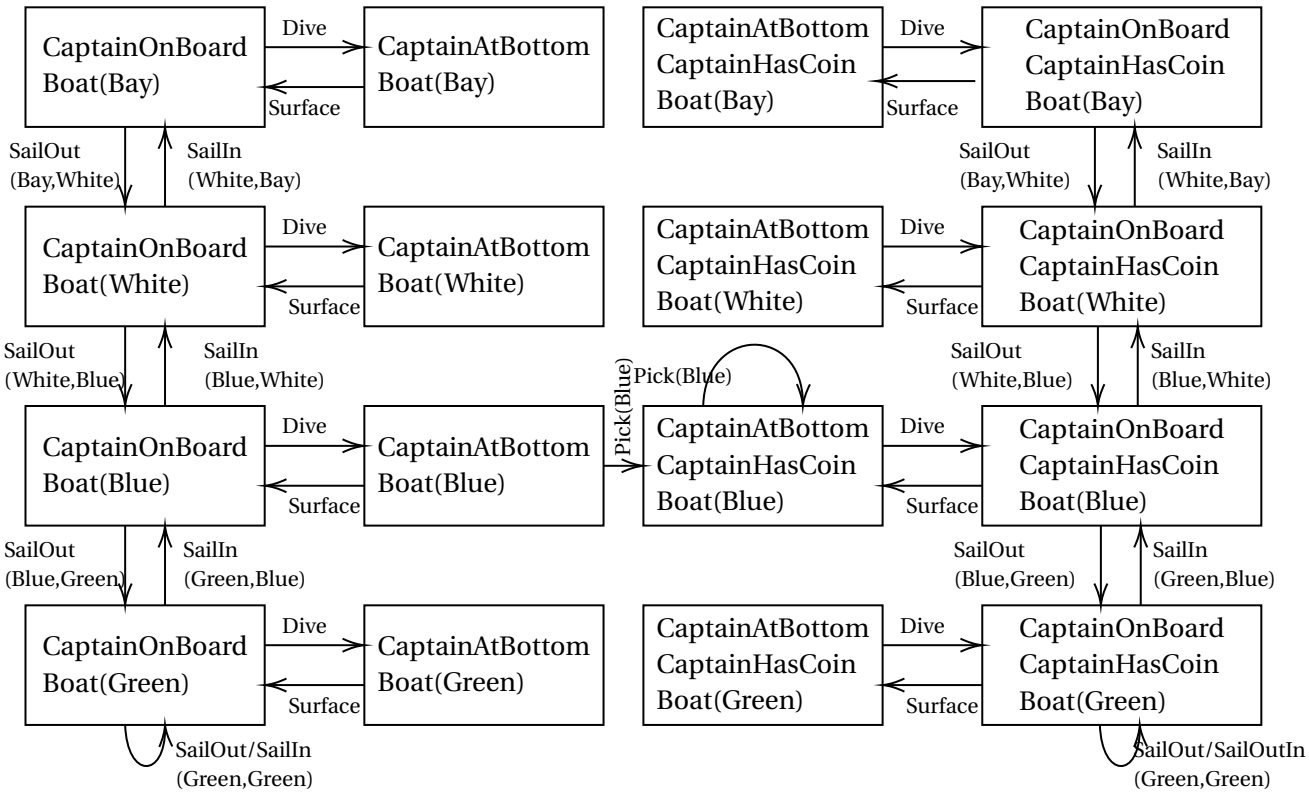


Figure 1: Connected graph representing the state space of the treasure hunt problem

3.2 Answers to questions

- There are an infinite number of plans that lead to the satisfaction of the goal, if we take into the consideration that the ship can keep going back and forth between two seas indefinitely, and thus there can be an infinite number of plans that lead to satisfaction of the goal.
- There would still be an infinite number of plans leading to the satisfaction of the goal, if one considers that instead of sailing from Green to Green, the ship could just sail back to Blue and then to Green again, so the infinite looping between seas is still possible.
- The most optimal plan is of course for the captain to immediately head for the Blue sea, get a coin, and return to port. This is done with:
  $SailOut(Bay, White)$ $SailOut(White, Blue)$ $Dive$ $Pick(Blue)$
  $Surface$ $SailIn(Blue, White)$ $SailIn(White, Bay)$
  So in 7 steps.

3.3 Belief State Space

3.3.1 Graph

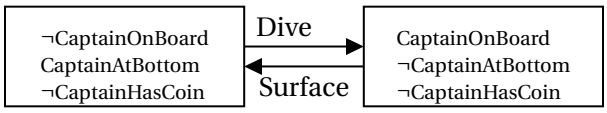


Figure 2: Illustration of the reachable states in the belief state space.

The captain does not know the current location, thus the ship cannot sail anywhere since the *from* value in the precondition is not fulfilled. Therefore the captain can only *Dive* and *Surface* in an infinite loop. See Figure 2 for an illustration. I have assumed that we still use the closed-world assumptions that $\neg CaptainAtBottom$ and $\neg CaptainHasCoin$.

3.3.2 Answer to questions

- As given by the PDDL, fluents that are always true include
  $Next(Bay, White)$, $Next(White, Blue)$, $Next(Blue, Green)$,
  while fluents that are always false include
  $Next(Bay, Blue)$, $Next(White, Green)$, $Next(Bay, Bay)$.
- The initial physical state contains 4 states, one for each sea, as we do not know which sea we start in.
- There are no plans that lead to satisfaction of the goal, since there is no way of actually sailing between seas (and Picking up the coin from *Blue*).
- As there are no plans that lead to satisfaction of the goal, there is no optimal path.

3.4 Belief state, with a twist

3.4.1 Graph

As soon as the captain sails to either se from the initial state, we will be certain of where we are, and can use the same method as the fully observable states, ie sailing wherever we want. Please see Figure 3. Just as in the previous problem I assume that we still use the closed-world assumptions that $\neg CaptainAtBottom$ and $\neg CaptainHasCoin$. Should that not be the case, then the drawn graph would be much larger.

3.4.2 Answer to questions

- The initial physical state still contains 4 states, since there are still 4 possible seas that the ship may be located at.
- There are still an infinite number of plans leading to satisfaction of the goal, as it is still possible to get caught in infinite loops of jumping back and forth between seas.
- The most optimal plan is now only 5 steps, because can proceed directly from sea to sea without going through unneccessary seas. The most optimal path is

$$SailToBlue \rightarrow Dive \rightarrow Pick(Blue) \rightarrow Surface \rightarrow SailToBay \tag{5}$$

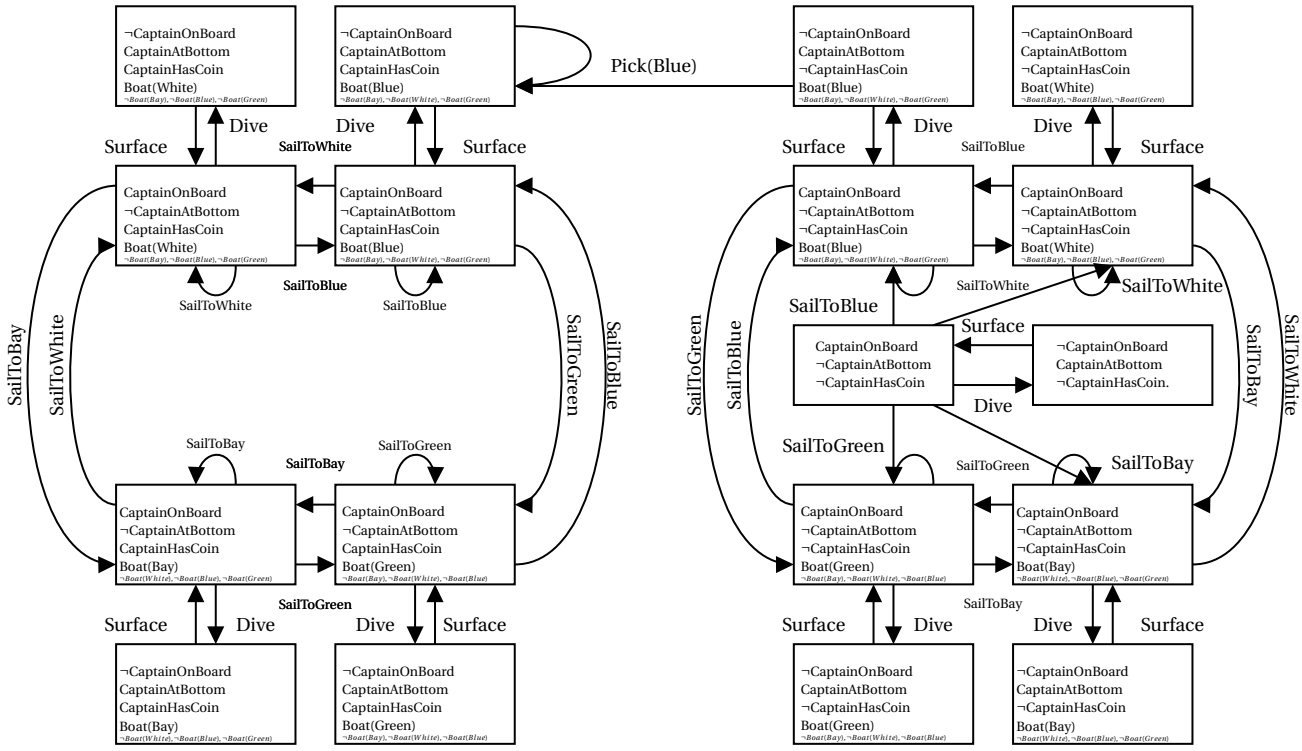
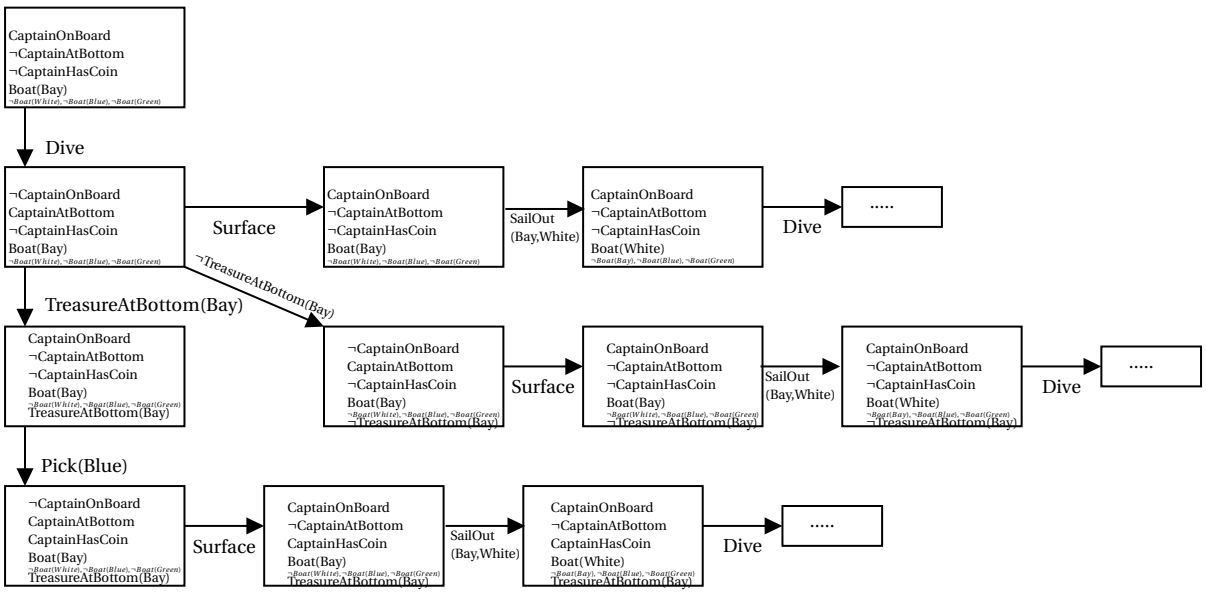


Figure 3: Belief state space with modified SailTo actions

3.5 State space with Percept

3.5.1 Graph



In essence, the captain can still dive at every sea, but when the captain dives at a sea he has not dived at before, the captain can choose to either *Surface* immediately, or perceive whether there is a treasure there or not. This effectively splits into 3 subtrees:

- One with the same information as before, the captain surfaces, and can either dive immediately again, or keep sailing. Rinse and repeat.

- If *Perceive* is executed, we will end up with more information, either *TreasureAtBottom(at)* or ¬*TreasureAtBottom(at)*, and the now known fluent will result in 2 new subtrees where this information is known. This process is repeated at each diving point, leading to a lot of subtrees. Of course subtrees can be connected to each other if for instance if *TreasureAtBottom(Blue)* and *TreasureAtBottom(White)* are Perceived in one subtree (in that order), and another Perceives them in opposite order; the information gained is still the same.

3.5.2 Contingent plan

```
1  //Assume initial sea is Bay
2  Dive();
3  if (TreasureAtBottom(Bay)) {
4      Pick(Bay);
5      Surface();
6      return;
7  }
8  else {
9      Surface();
10     SailOut(Bay,White);
11     Dive();
12     if (TreasureAtBottom(White)) {
13         Pick(White);
14         Surface();
15     }
16     else {
17         Surface();
18         SailOut(White,Blue);
19         Dive();
20         if (TreasureAtBottom(Blue)) {
21             Pick(Blue);
22             Surface();
23         }
24         else {
25             Surface();
26             SailOut(Blue,Green);
27             Dive();
28             if (TreasureAtBottom(Green)) {        // Unnecessary
29                 Pick(Green);
30                 Surface();
31             }                                     // Unnecessary
32             else {                                // Unnecessary
33                 Surface();
34                 NoOp();                           // Unnecessary
35             }                                     // Unnecessary
36             SailIn(Green,Blue);
37         }
38         SailIn(Blue,White);
39     }
40     SailIn(White,Bay);
41     return;
```

42 }

The lines marked as Unnecessary do not apply if the algorithm works under the assumption that there is a treasure, and that there is only one treasure in our world (because the algorithm *will* find a treasure somewhere).

3.6 B-level questions

- If we compare the size of the state space graph in Figure 1 with the belief state space graph in Figure 3, I we see that in these specific domains the belief state space graph will always be larger than the state space graph. This is because the belief state graph will always have an initial state where the location is unknown, and one action is required to determine the current location/sea, thus one extra node will always exist in the belief space graph. However if we instead consider the first sensorless case which has only 2 reachable states from the initial state (initial state included) as seen in Figure 2, we see that the belief state space can indeed be smaller than the state space state. Its all dependent on how the domain is defined.
- Consider the initial state in the sensorless case in Figure 3, where there are 3 fluents, and the initial state in the fully observable case in Figure 1 where there are only 2 explicitly listed fluents. This would suggest that the size of the representation of the fully observable case is larger than the sensorless case equivalent. However imagine that we are in a PDDL domain where there is no coin, and the captain cannot dive, ie the fluents *CaptainAtBottom* and *CaptainHasCoin* do not exist. Then we have the opposite space size relation when the size of the representation of the fully observable case is *smaller* than the sensorless equivalent. I thus must conclude that the size relation between the fully observable and sensorless cases is not a oneway street. It is dependent on how many negative fluents are known in the initial state (and also on how the domain is defined). (This assumes that the fully observable case is a closed world, and the sensorless case an open world)

4 Exercise 4

The following environment was given:

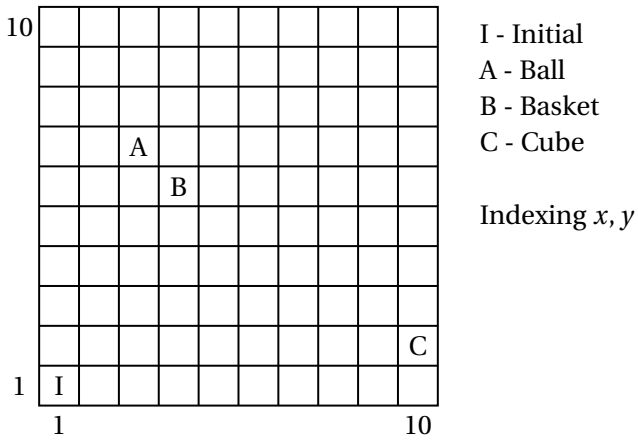


Figure 4: Illustration of the robot world

4.1 C-level

See Figure 5 for graphical representation.

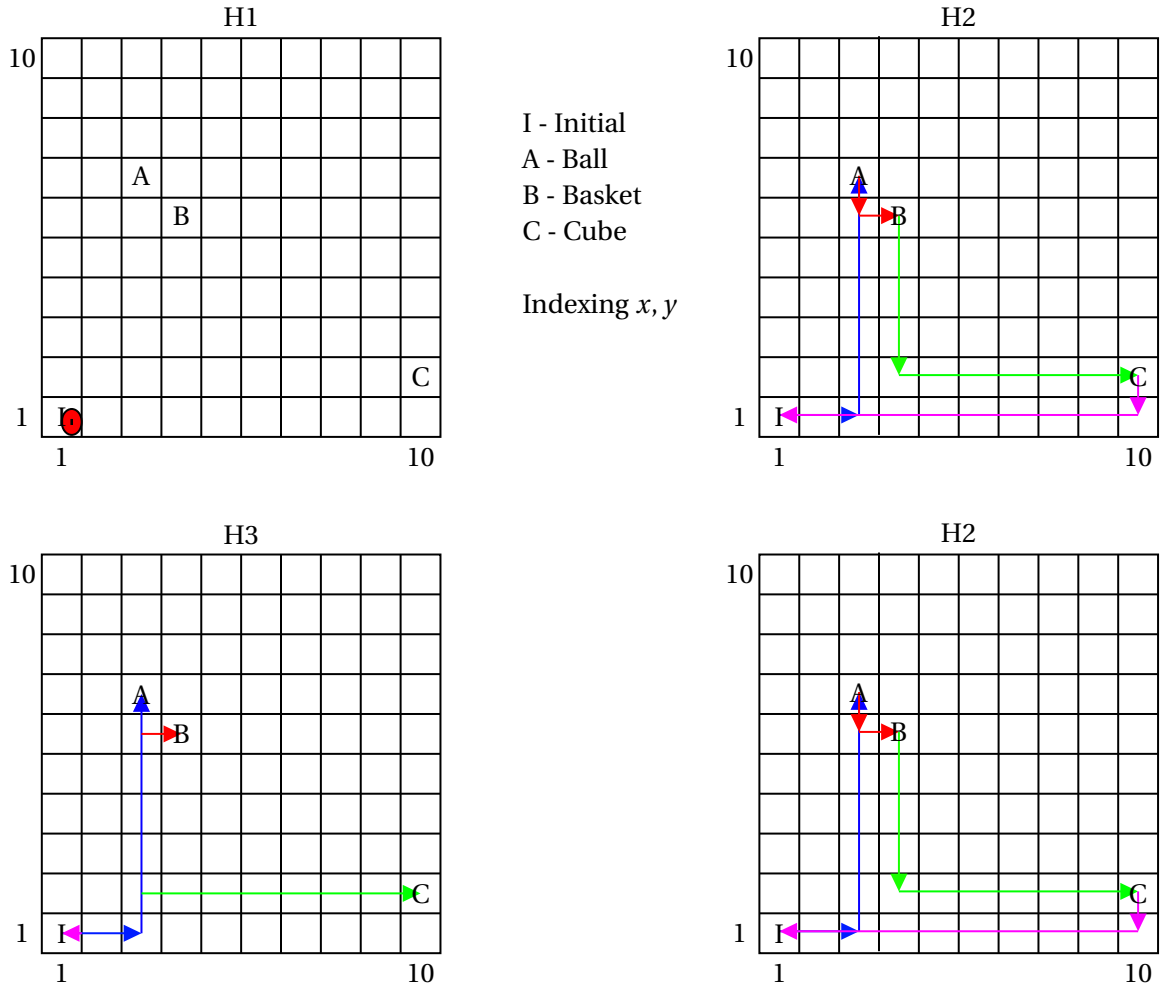


Figure 5: Illustration of the robot world with paths according to heuristics H1-H4.

- *What is the value of h_1 for the initial state?*
  3. By ignoring all the preconditions the robot can simply drop all items one-by-one at $1,1$ immediately.

- *What is the value of h_2 for the initial state?*
  36. By ignoring the precondition $Free(Robot)$ in $Pick$, we can effectively walk directly between the objects and pick them all up and head back to the goal.

- *What is the value of h_3 for the initial state?*
  22. By ignoring the delete lists in the effects, whenever the robot moves, the robot exists at one more location, ie the robot does not cease to exist at a location when moved. This makes it possible to "teleport" to locations, or in this case for example move from A to B in one action, if the heuristics assumes the robot is still in $4,5$ and can move immediately to $5,5$.

- *What is the value of h_4 for the initial state?*
  36. This heuristic is effectively the same as $h2$ logically speaking. By ignoring the delete lists in $Pick$, the robot arm will always be free, which is equivalent to ignoring the $Free(Robot)$ precondition in h_2.

- *Compare heuristics h_1 and h_2. Does one of them dominate?*
  h_1 is smaller than h_2, so h_2 dominates h_1.

- *Compare heuristics h_3 and h_4. Does one of them dominate?*
  h_3 is smaller than h_4, so h_4 dominates h_3.

- *Compare heuristics h_2 and h_4. Does one of them dominate?*
  h_2 gives the same value as h_4, so none of them dominate.

4.2 B-level - dead ends

4.2.1 Defining terminology

From Jörg Hoffnam: "Where 'ignoring delete lists' works: Local search topology in planning benchmarks"

> One phenomenon is clearly relevant for the performance of heuristic state space search is that of dead end states s, $gd(s) = \infty$. A heuristic function h can return $h(s) = \infty$. Taking this as an indication that s is a dead end, the obvious idea is to remove s from the search space. This technique is only adequate if h is *completeness preserving* in the sense that $h(s) = \infty \Rightarrow gd(s) = \infty$. With a completeness-preserving heuristic, a dead end state s is called *recognized* if $h(s) = \infty$ and unrecognized otherwise. If a task can not be solved even when ignoring the delete lists, then the task is unsolvable. With respect to dead ends, any planning state falls into one of the four classes. The state space is called:
>
> 1. *Undirected*, if, got all (s, s'), $(s', s) \in T$.
> 2. *Harmless*, if there exists $(s, s') \in T$ such that $(s', s) \notin T$, and, for all $s \in S$, $gd(s) < \infty$.
> 3. *Recognized*, if there exists $s \in S$ such that $gd(s) = \infty$, and, for all $s \in S$, $gd(s) = \infty$ then $h(s) = \infty$.
> 4. *Unrecognized*, if there exists $s \in S$ such that $gd(s) = \infty$ and $h(s) < \infty$

So from that quote, and the paragraph previous to that, I extract:

- **Goal distance:** the shortest route to the goal.

  > The goal distance $gd(s)$ for a state $s \in S$ is the length of a shortest path in (S, T) from s to a goal state, or $gd(s) = \infty$ if there is no such path.

- **Dead end:** when the goal distance is ∞, ie no route exists.

- **Recognized dead end:** when the heuristics reports that there is a dead end following the searched path.

- **Unrecognized dead end:** when the heuristic does not return a value that indicates a dead end (∞) in the searched path, ie a dead end exists, but the heuristic does not "see" it.

4.2.2 Does there exist a recognized / unrecognized dead end for H3?

Since the goal distance for any state in the unmodified version of the robot world (as depicted in Figure 4) is finite, no dead end exist at all. Thus there can be no recognized / unrecognized dead end using H3 (recall a prerequisite for recognized / unrecognized state space is that first $gd(s) = \infty$).

- *Does there exist a reachable recognized dead end for H3?*
  No.

- *Does there exist a reachable unrecognized dead end for H3?*
  No.

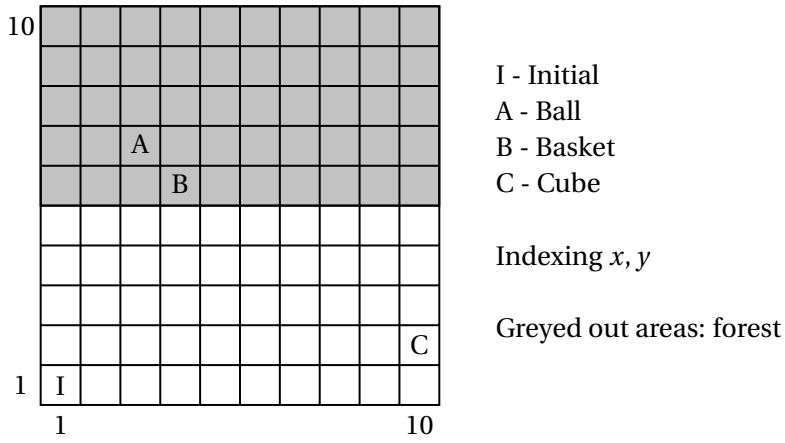


Figure 6: Illustration of the modified robot world.

4.3 B-level - new variant of robot world

To recap: The modified robot world shows that if the robot enters the forest areas using $MoveInCity$, without a flashlight, it will be left unable to leave the forest with $MoveInForest$, and will then reach a dead end. As illustrated in Figure 6.

Suppose that the robot is in the City, then there exist no dead end, since the robot will always have the possibility to grab the flashlight before entering the forest. In other words, there is a finite goal distance even if the robot is at the border to the Forest.

Suppose now that a state s where the robot has just entered the Forest without a flashlight is evaluated. The goal distance is now ∞ since the robot cannot leave the Forest without a flashlight. H3 does naively "save" the previously visited locations, but if the state s is the root of the search, H3 will not have a location in the City saved. Thus H3 will also yield ∞, and we have a recognized dead end.

Following the quote from the article as shown in the previous subsection, I also deduce that H3 is completeness preserving, as well as defines the state space (S', T') which is Recognized.

Since these were the two possible scenarios for dead ends in this domain and I cannot find an unrecognized dead end, I conclude that there is none for H3.

- *Does there exist a reachable recognized end for H3?*
  Yes.
- *Does there exist a reachable unrecognized end for H3?*
  No.

4.4

Suppose that we are in a domain as described in Section 4.3. Introduce another heuristics, H5, which ignores the $HasFlashlight(Robot)$ precondition in the action $MoveInForest$. The H5 state space will naturally be called Unrecognized, since the value of h_5 will always be finite (while the goal distance is infinite if robot has entered Forest without flashlight), regardless of if the robot has gone into the forest without a flashlight or not. Now, suppose that the robot is at the border to the Forest, H5 will suggest that the robot enter the Forest because it is faster than going to fetch the flashlight. If the search algorithm then follows H5 and enters the forest, it will be left without a way out, and it

will then be impossible to find a goal. Even if a state where the robot has entered the forest without a flashlight is evaluated with H5, the search algorithm will keep searching with H5 since there is no recognized end.