Introduction to AI - assignment 3

Oded Yechiel

27/11/18

Contents

Exercise 1 Agents and environment simulators

1.1 An agent that plays Whist

Whist is a zero-sum adversarial game. The environment is static and discrete, non observed and non deterministic. There are many possible goal states, and the agent should maximize its own performance while minimizing the performance of the adversarial agent. Therefore, a **utility-based agent** will best assess the situation and provide the ultimate action.

1.2 An agent that can solve Sokoban problems

Sokoban is a single player puzzle-like game with a single goal state. The environment is static, discrete, fully observable and deterministic. Therefore, a **goal-based agent** will be most suitable than all other agents.

1.3 An autonomous humanoid robot that can win the DARPA robotics challenge

The DRC's task require the robot to perform a series of tasks. Although points are distributed per task, such as, go through the gate, enter the car, drive the car and exit the car, these are merely subgoals and are sequential. Therefore, it would be most logical of using a **goal-based agent**. The environment is static, continuous, discrete and partially observable.

1.4 An internet shopping agent specializing in trip planning

An internet shopping agent has many goals, with various optimizing criteria, such as time, cost and comfort. The environment **may be** considered as a deterministic, fully observable (after query the agent has all of the information) single operating agent (assuming tickets are not running out). Since there is no one perfect goal, the agent should be **utility-based agent.**

1.5 An agent that can play Go

Go is a fully observable, non-deterministic, static adversarial game. Although there is only one winner and a definite goal state, it is unlikely for any agent to calculate so deeply into the game and find the optimal action. Therefore, for a given action it is required to assess the situation and decide what action is sub-optimal for the problem. Therefore, a **utility-based agent** is advised.

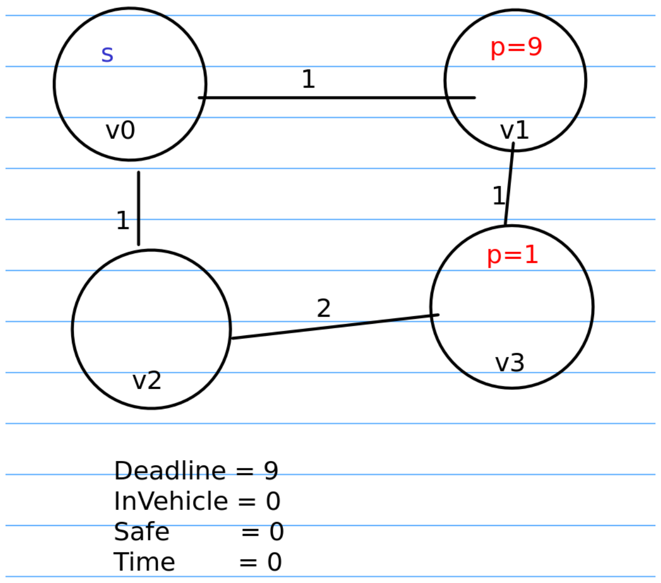


Figure 1: The hurricane environment.

Exercise 2 Search

In the hurricane environment, shown in Fig. 1, there are many different ways in which it is possible to calculate the heuristics of an action. The half-star heuristic, h' calculates the performance by locating the vertex with most un-savable people and multiplying them by a large number. In this exercise we shall multiply each person by 100 for ease of calculation and distinction, although multiplying by the "Deadline value" is sufficient.

We can examine an example of calculating the heuristics for the starting location. The agent is at V_0 and it is possible to go to V_1 or V_2. The heuristic for going to V_1 will be 100, since we have picked up all the 9 people in V_1 and the only town with people is V_3 with 1 person. Going to V_2 will result in a heuristic of 900, since in V_1 there are 9 people which are un-savable.

NOTE - this heuristic is not admissible! In the assignment we have used a different heuristic which is admissible where all of the people are taken into account, not only the people in the town with most people.

2.1 Greedy agent

From the example above, the greedy agent will obviously prefer going to V_1, even-though it is obvious that the pick-up will be his final move. Fig. 2 shows the decision making of the greedy agent: $V_0 \rightarrow V_1 \rightarrow V_3$.

The greedy agent relies on an extremely optimistic heuristic that assumes that whoever in the vehicle will get to safety, and does not take those people (the ones in the vehicle) into account in the calculation. Therefore, the result of the greedy agent is not optimal.

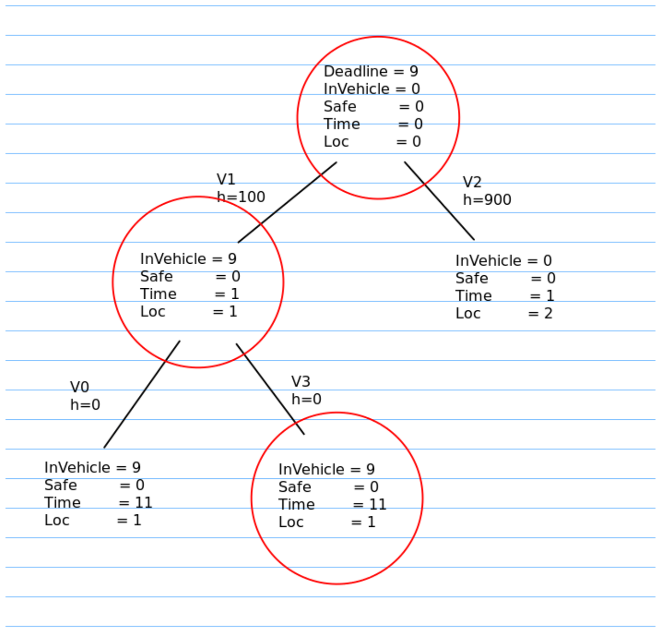


Figure 2: Greedy agent expansions

2.2 A^*

Opposed to the greedy agent, the A^* agent will provide an optimal solution, although the heuristic is not admissible. This is due to the fact that the g value is complementary. Since, the heuristic is extremely optimistic the A^* agent will perform much more expansions than he would have needed if the heuristic was better.

The A^* agent takes into account the time took to get to this state, and if a goal state is reached it adds the number of unsaved people.

$$g_{v_i \to v_j} = \begin{cases} g[v_i] + w_{v_i v_j}, & t + w_{v_i v_j} < T_{deadline} \\ g[v_i] + w_{v_i v_j} + 100 \times UnsavedPeople, & else \end{cases} \tag{1}$$

The agent adds the g value with the h value to decide which node to expand.

Fig. 3 shows the expansion tree of the A^* agent. It can be seen that the agent has 11[1] expansions to find the optimal path, which is $V_0 \to V_2 \to V_3 \to V_2 \to V_0$.

2.3 Real-time A^*

This agent is similar to the A^* agent, however, it stops after 2 expansions and makes a decision. Fig 4a shows the first two expansions of the agent. It can be seen that in this case, the same result state is achieved as with the the A^* agent. After the agent reached V_3 the agent expanded up to two more nodes, as shown in Fig. 4b.

[1]The state V_0 with f=902 was not expanded since it is assumed there is a checking for loops to avoid expanding already visited states.

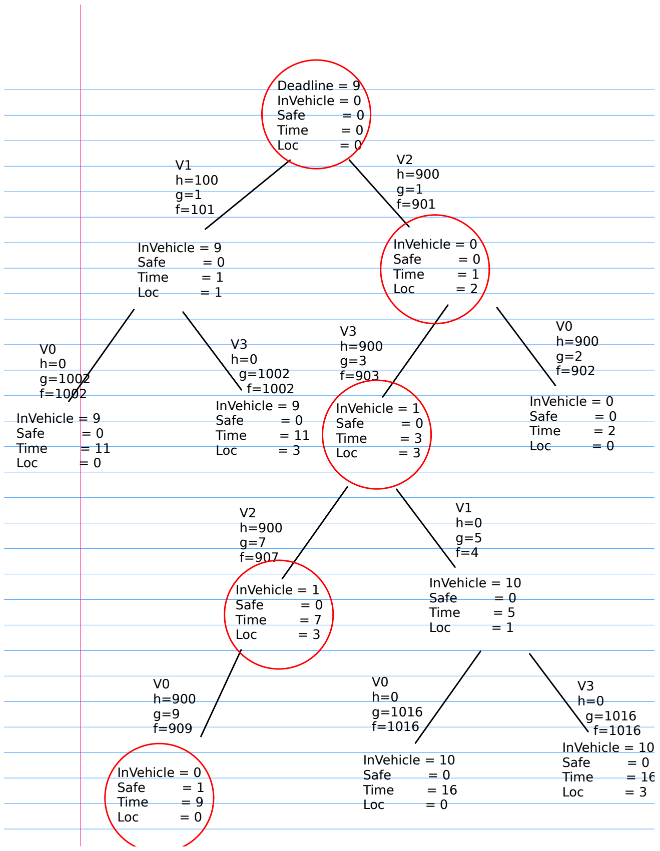
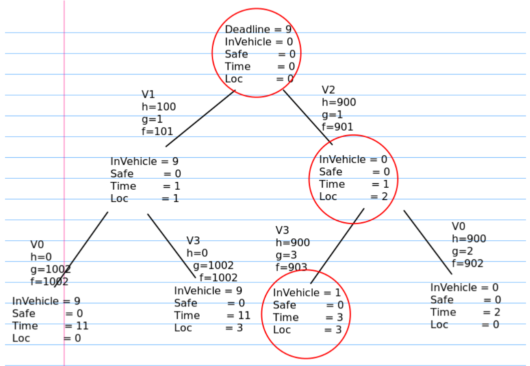
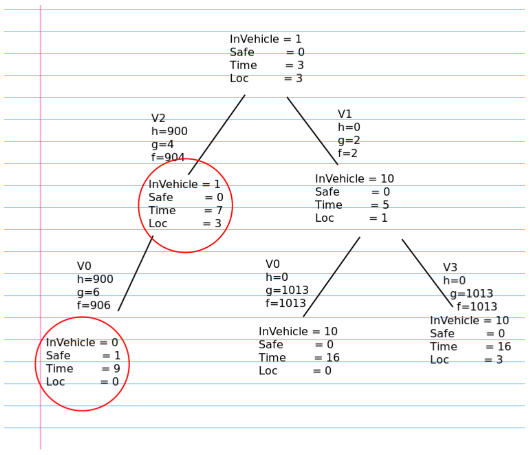


Figure 3: A^* agent expansions



(a) Real-Time A^* agent first 2 expansions.



(b) Real-Time A^* agent second 2 expansions.

2.4 $h = 2 * h'$

Since h' multiplied the number of people by a large number, it does not matter if this large number is X or $2 * X$. Therefore, all of the results will remain the same, and the heuristic is still admissible.

Exercise 3 Game trees

3.1 Each agent out for itself, and they cannot communicate.

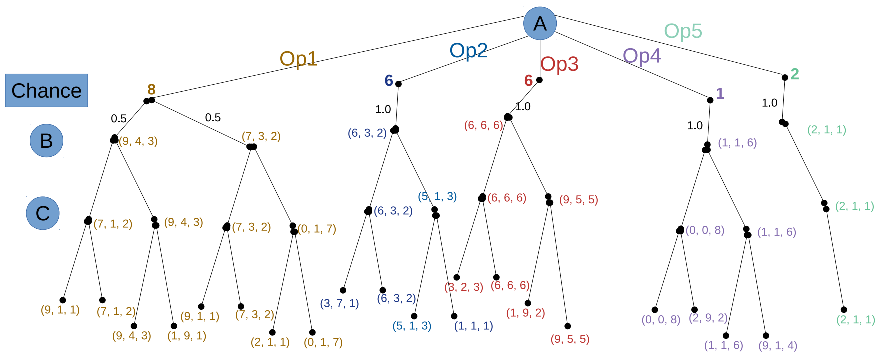


Figure 5: Option 1 is the best

3.2 Paranoid assumption: B and C are against A

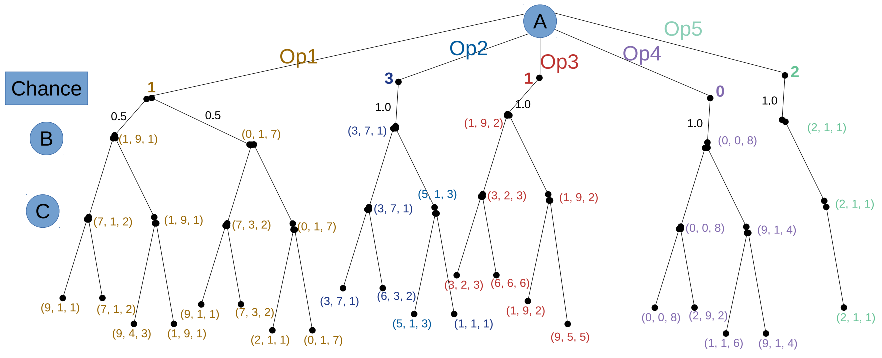


Figure 6: Option 2 is the best

3.3 B and C may agree on a deal if it is beneficial to both of them

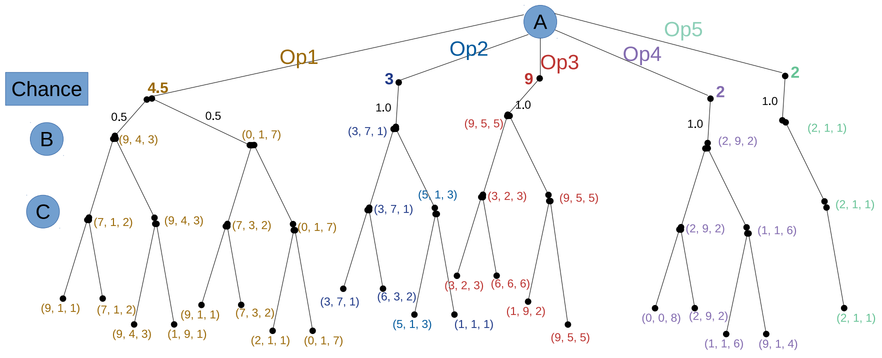


Figure 7: Option 3 is the best

3.4 A and C are partners aiming to maximize the sum of their scores

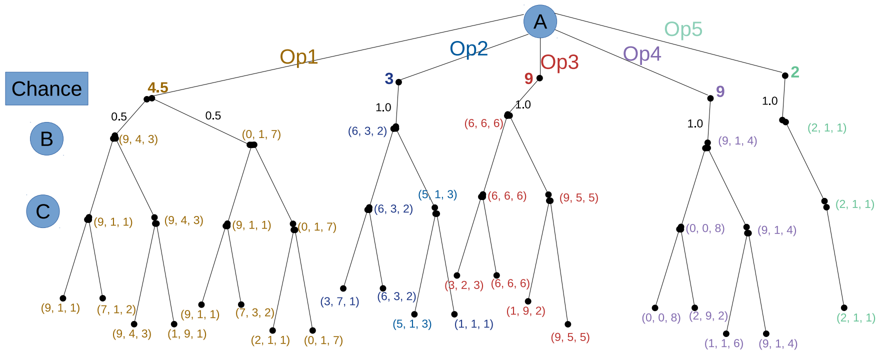


Figure 8: Option 4 is the best

3.5 A is a firm believer in Murphy's laws: "mother nature is a b*!=h", and wants toplay absolutely safe.

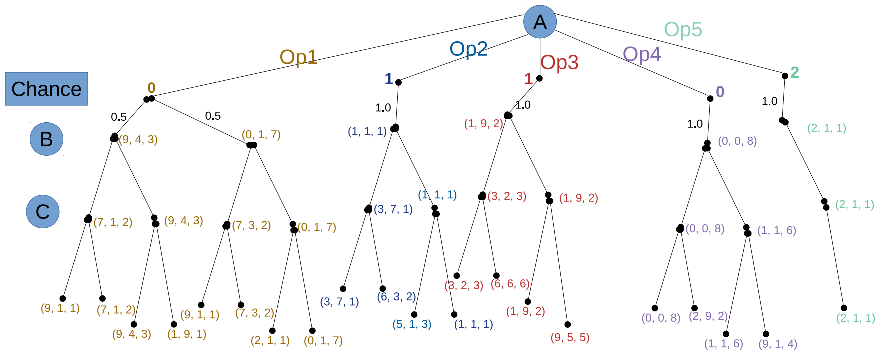


Figure 9: Option 5 is the best

Exercise 4 Game-tree search - alpha-beta pruning

4.1 Construct an example where the optimal first move is no-op

Assuming the world shown in Fig. 10. K=0 and Deadline=5. In this world B is a vandal agent that works as follows: He moves one turn, destroying the traversed edge, and then does NoOp() one turn. The agent A is at V4 and already carrying people (number does not matter). The agent wants to get to the shelter, however, he must see first what B will do. If B goes to V1, A have to be at V1 already, because next move B might go to the shelter and A won't be able to save the carrying people. However, if B goes to V2 or V3 first, the optimal solution will be to save an additional person.

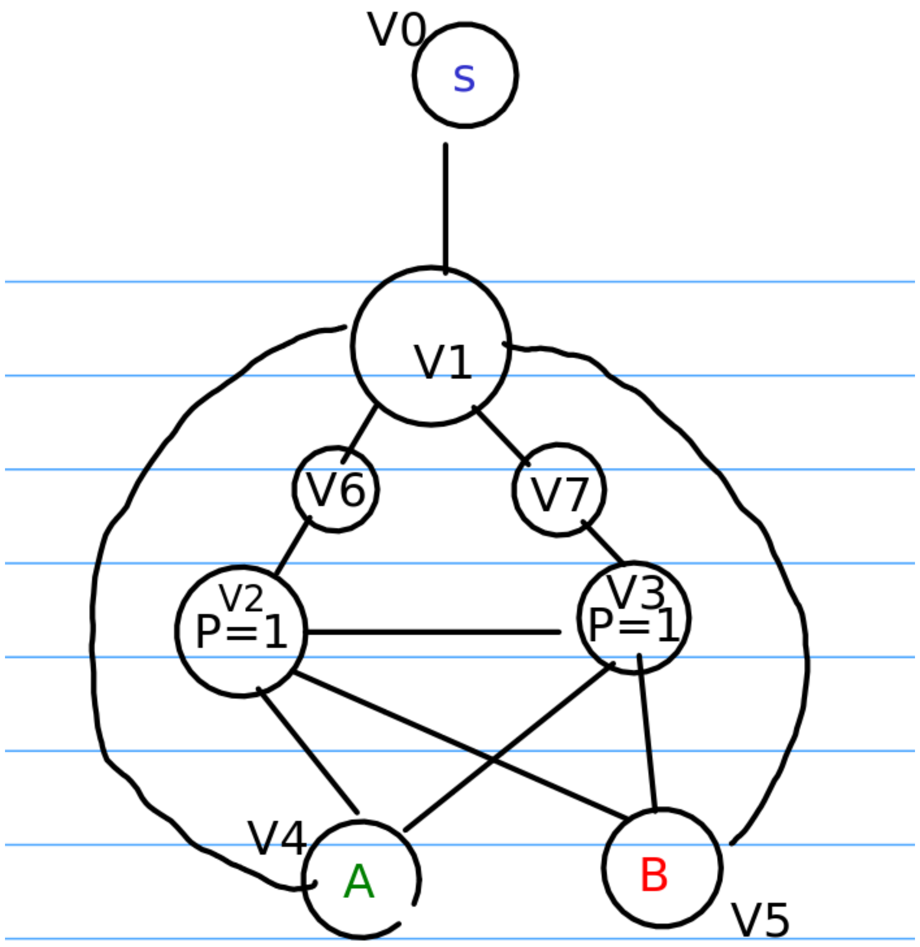


Figure 10: World where all edges are 1. B is a vandal agent that waits 1 turn in place, and one turn moves randomly and destroys the traversed edge. A is carrying 1 person and should wait to see what B does first.

4.2 Construct and show an example

Using the heuristic of calculating how many people will not be saved, the expansion tree is given for the choices of agent A. At first it can go to V2, V3 or do NoOp. For each option, the B agent will randomly go to either V2, V3 or V1 with probability of 1/3.

As seen by Fig. 11, If agent A will go to V2 or V3 there is a possibility that the agent will finish the game with a very high heuristic value, therefore, the h value is 466.

However, if agent A will do No-Op then it has enough time to react to agent's B decision.

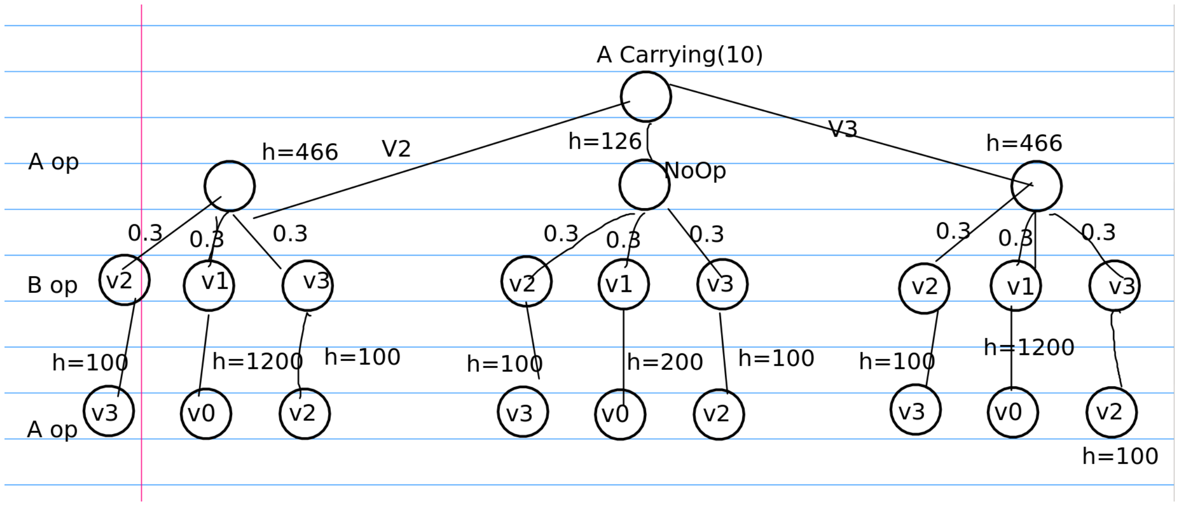


Figure 11: Expansion tree.

4.3 Show where alpha-beta pruning can decrease search effort in the tree from b

Assume that we start expanding V2. After the choice of agent B, several of the remaining choices can be discarded. In addition (and mostly), due to symmetry, expanding V3 will be futile and will finish this branch much faster.

Exercise 5 Propositional logic

For validity it will be shown that the sentence is always true.

An example for satisfiability a true model and a false model will be given.

For unsatisfiability it will be shown that the setence is always false.

Number of models equals to: 2^n, if the sentence is valid, where n is the number of symbols.

5.1 $(\neg A \wedge \neg B \wedge \neg C \wedge \neg D \wedge \neg E \wedge F) \vee (A \wedge B \wedge C \wedge D \wedge E)$

True model - A, B, C, D, F are true, therefore,

$True \vee False = True$

False model - A is true, B is false (the rest whatever), therefore,

$False \vee False = False$

Satisfiable. Number of satisfiable models is 3.

5.2 $(\neg A \vee \neg B \vee \neg C \vee \neg D \vee \neg E \vee \neg F) \wedge (A \vee B \vee C \vee D \vee E \vee F)$

False model - A, B, C, D, F are true, therefore,
$True \wedge False = False$
True model - A is true, B is false (the rest whatever), therefore,
$True \wedge True = True$
Satisfiable. Number of satisfiable models is 62.

5.3 $(A \vee B \wedge (D \vee \neg A) \wedge (E \vee A) \Rightarrow (B \vee C \wedge (\neg D \vee E))) \wedge (A \wedge \neg A)$

$\alpha \equiv (A \vee B \wedge (D \vee \neg A) \wedge (E \vee A) \Rightarrow (B \vee C \wedge (\neg D \vee E)))$
$\alpha \wedge (A \wedge \neg A) \rightarrow \alpha \wedge False \rightarrow False$
Unsatisfiable Number of models is 0.

5.4 $(A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C$

implication elimination $\neg(A \wedge (\neg A \vee B) \wedge (\neg B \vee C)) \vee C$

Distributivity $\neg((A \wedge \neg A) \vee (A \wedge B)) \wedge (\neg B \vee C)) \vee C$

$\neg(False \vee (A \wedge B)) \wedge (\neg B \vee C)) \vee C$

Associativity $\neg(A \wedge (B \wedge (\neg B \vee C))) \vee C$

Distributivity $\neg(A \wedge ((B \wedge \neg B) \vee (B \wedge C))) \vee C$

$\neg(A \wedge ((False) \vee (B \wedge C))) \vee C$

$\neg(A \wedge (B \wedge C)) \vee C$

De Morgan $\neg A \vee \neg B \vee \neg C \vee C \rightarrow True$

Valid. Number of models is 8.

5.5 $\neg((A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C)$

implication elimination $\neg(\neg(A \wedge (\neg A \vee B) \wedge (\neg B \wedge C)) \vee C)$

Distributivity $\neg(\neg((A \wedge \neg A) \vee (A \wedge B) \wedge (\neg B \wedge C)) \vee C)$

$\neg(\neg((A \wedge B) \wedge (\neg B \wedge C)) \vee C)$

De Morgan $\neg(\neg A \vee \neg B \vee \neg\neg B \vee \neg C \vee C)$

Double negation elimination $\neg(\neg A \vee \neg B \vee B \vee \neg C \vee C) \rightarrow$

$\neg(\neg A \vee True \vee True) \rightarrow$

$\neg(True) \rightarrow False$

Unsatisfiable. Number of models is 0.

5.6 $(A \Rightarrow \neg A) \lor (\neg A \Rightarrow A)$

implication elimination $(\neg A \lor \neg A) \lor (A \lor A) \rightarrow$

$\neg A \lor A \rightarrow True$

Valid. Number of models is 2.

Exercise 6 FOL and situation calculus

6.1 Axioms of world behavior

6.1.1 Auxiliary predicats

- TimeUp(s) - check if time passed the deadline

$$\forall s,t \exists d \quad TimeUp(s) \Rightarrow Time(t,s) > Deadline(d) \tag{2}$$

- Bidirectional edge

$$\forall e, v1, v2 \quad Edge(e, v1, v2) \Leftrightarrow Edge(e, v2, v1) \tag{3}$$

6.1.2 Effect axioms

- Update time fluent

$$\forall a,s,t \quad (\exists e,w \quad a = Traverse(e) \land Weight(e,w)) \land Time(t,s) \Rightarrow Time(t+w, Result(a,s)) \tag{4}$$

or,

$$\forall at,a,s,t \quad a = NoOp() \land Time(t,s) \Rightarrow Time(t+1, Result(a,s)) \tag{5}$$

- Loc update

$$\forall s, v1, a, e \quad (\exists v \quad a = Traverse(e) \land Edge(e,v,v1) \land Loc(v,s)) \Rightarrow Loc(v1, Result(a,s)) \tag{6}$$

- Pickup people

$$\forall c,v,a,s \quad Carrying(c,s) \land Loc(v, Result(a,s)) \Rightarrow \exists n At(v,n,s) \land Carrying(c+n, Result(a,s)) \tag{7}$$

- Remove picked up people
  For all states and vertices, 0 people are at vertex, v, if loc is v.

$$\forall v,a,s \quad Loc(v, Result(a,s)) \Rightarrow At(v, 0, Result(a,s)) \tag{8}$$

- Save people
  People in vehicle are safe if vehicle is at a shelter and time is not up

$$\forall s,a,v,n \quad Safe(n,s) \land Loc(v,s) \land Shelter(v) \land \neg TimeUp(s) \Rightarrow \exists c \quad Safe(n+c, Result(a,s)) \land Carrying(c,s) \tag{9}$$

- Remove carried people

$$\forall s,a,v \quad Loc(v,s) \land Shelter(v) \land \neg TimeUp(s) \Rightarrow Carrying(0, Result(a,s)) \tag{10}$$

6.1.3 Successor state axioms

- Loc unchanged

$$\forall a, s, v \quad Loc(v, s) \Leftrightarrow Loc(v, Result(a, s)) \land (a = NoOp()) \tag{11}$$

- Number of people in towns unchanged

$$\forall a, v, s \quad At(v, n, s) \land \neg Loc(v, s) \Rightarrow At(v, n, Result(a, s)) \tag{12}$$

- Number of safe people unchanged

$$\forall a, s \quad Safe(n, s) + (\exists v \quad (Loc(v, s) \land \neg Shelter(v)) \Rightarrow Safe(n, Result(a, s)) \tag{13}$$

- Number of people carrying unchanged

$$\forall a, s \quad Carrying(n, s) \land (\exists v \quad (Loc(v, s) \land \neg Shelter(v)) \Rightarrow Carrying(n, Result(a, (cs)) \tag{14}$$

6.2 Prove that the people in V3 are safe

Please note - This is an extremely long process. Therefore, all the NoOp are not written since they are not part of the plan. In addition, simple arithmetic, which should be part of the axioms are excluded.

6.2.1 CNF

$$\forall s, t \exists d \quad \neg TimeUp(s) \lor (Time(t, s) > Deadline(d)) \tag{15}$$

$$\forall e, v1, v2 (\neg Edge(e, v1, v2) \lor Edge(e, v2, v1)) \land (Edge(e, v1, v2) \lor \neg Edge(e, v2, v1)) \tag{16}$$

$$\forall a, s, t \quad (\forall e, w \quad \neg a = Traverse(e) \lor \neg Weight(e, w)) \lor \neg Time(t, s) \lor Time(t + w, Result(a, s)) \tag{17}$$

$$\forall s, v1, a, e \quad (\neg \exists v \quad a = Traverse(e) \land Edge(e, v, v1) \land Loc(v, s)) \lor Loc(v1, Result(a, s))$$
$$\forall s, v1, a, e \quad (\forall v \quad \neg(a = Traverse(e)) \lor \neg Edge(e, v, v1) \lor \neg Loc(v, s)) \lor Loc(v1, Result(a, s)) \tag{18}$$

$$\forall v, a, s, c \quad \neg Carrying(c, s) \neg Loc(v, Result(a, s)) \lor (\exists n \quad Carrying(n + c, Result(a, s)) \land At(v, n, s)) \tag{19}$$

$$\forall v, a, s \quad \neg Loc(v, Result(a, s)) \lor At(v, 0, Result(a, s)) \tag{20}$$

$$\neg Safe(n, s) \lor \neg Loc(v, s) \lor \neg Shelter(v) \lor TimeUp(s) \lor \exists (c \quad Safe(n + c, Result(a, s)) \land Carrying(c, s)) \tag{21}$$

$$\forall s, a, v \quad \neg Loc(v, s) \lor \neg Shelter(v) \lor TimeUp(s) \lor Carrying(0, Result(a, s)) \tag{22}$$

$$\forall a, v, s \quad At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \tag{23}$$

$$\forall a, v, n, s \quad \neg Safe(n, s) \vee \neg Loc(v, s) \vee Shelter(v) \vee Safe(n, Result(a, s)) \tag{24}$$

$$\forall v, a, s \quad \neg Carrying(c, s) \vee \neg Loc(v, Result(a, s)) \vee Shelter(v) \vee Carrying(n, Result(a, s))) \tag{25}$$

6.2.2 What we need to prove

From Q2 we have discovered that the optimal plan is $E2 \to E3 \to E3 \to E2$, and the location of the agent will be $V0 \to V2 \to V3 \to V2 \to V0$.

In order to show this plan, $p = [Traverse(E2), Traverse(E3), Traverse(E3), Traverse(E2)]$ will actually save the people in $V3$, we shall prove by contradiction that time will not elapse, i.e. $KB \wedge p \wedge TimeOut(S4)$ is false. In addition, we shall prove that the people are actually safe, i.e., $KB \wedge p \wedge (\exists n \quad \neg(Safe(n, S4) \wedge At(V3, n, s0)))$

6.2.3 Proof with resolution

$$S1 = Result(Traverse(E2), S0) \tag{26}$$
$$S2 = Result(Traverse(E3), S1) \tag{27}$$
$$S3 = Result(Traverse(E3), S2) \tag{28}$$
$$S4 = Result(Traverse(E2), S3) \tag{29}$$
$$TimeOut(S4) \tag{30}$$

$$Loc(V0, S0) \tag{31}$$
$$Carrying(0, S0) \tag{32}$$
$$At(V3, 1, S0) \tag{33}$$
$$At(V1, 9, S0) \tag{34}$$
$$Safe(0, S0) \tag{35}$$
$$Time(0, S0) \tag{36}$$
$$Deadline(9) \tag{37}$$

And more...

1. Proof of Timeup Resolving 15 and 30 and 37

$$TimeUp(S4), Deadline(9), \neg TimeUp(s) \vee (Time(t, s) > Deadline(d)) \to \tag{38}$$

$$Time(t, S4) > Deadline(9)$$

Resolving 17 and 33 and Weight(E2,2) and 26

$$\begin{gathered} Time(0, S0), S1 = Result(Traverse(E2), S0), Weight(E2, 2), \\ \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \end{gathered} \tag{39}$$

$$Time(0+2, S1) \rightarrow Time(2, S1)$$

Resolving 17 and 39 and Weight(E3,2) and 27

$$\begin{gathered} Time(1, S1), S2 = Result(Traverse(E3), S1), Weight(E3, 1), \\ \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \end{gathered} \tag{40}$$

$$Time(2+1, S2) \rightarrow Time(3, S2)$$

$$\begin{gathered} Time(3, S3), S3 = Result(Traverse(E3), S2), Weight(E3, 1), \\ \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \end{gathered} \tag{41}$$

$$Time(3+1, S3) \rightarrow Time(4, S3)$$

$$\begin{gathered} Time(4, S3), S2 = Result(Traverse(E2), S3), Weight(E2, 2), \\ \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \end{gathered} \tag{42}$$

$$Time(4+2, S4) \rightarrow Time(6, S4)$$

From CNF 38 and the Time expansion above we get a contradiction, i.e.,

$$Time(6, S4), Time(t, S4) > Deadline(9) \Rightarrow False \tag{43}$$

This concludes that time does not elapse. We will continue with proving the people are safe.
2. Proof of safety

$$\exists n \quad \neg(Safe(n, S4) \wedge At(V3, n, s0))$$

$$\exists n \quad \neg Safe(n, S4) \vee \neg At(V3, n, s0)$$

Along with At(V3, 1, S0) we need to disprove that,

$$\neg Safe(1, S4)$$

$$\begin{gathered} S1 = Result(Traverse(E2), S0), Loc(V0, S0), Edge(E2, V0, V2), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)) \end{gathered} \tag{44}$$

$$Loc(V2, S1)$$

$$\begin{gathered} S2 = Result(Traverse(E3), S1), Loc(V2, S1), Edge(E3, V2, V3), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)) \end{gathered} \tag{45}$$

$$Loc(V3, S2)$$

$$\begin{aligned} S3 = Result(Traverse(E3), S2), Loc(V3, S2), Edge(E3, V2, V3), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)), and (16) \end{aligned} \tag{46}$$

$$Loc(V2, S3)$$

$$\begin{aligned} S4 = Result(Traverse(E2), S3), Loc(V2, S3), Edge(E2, V0, V2), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)), and (16) \end{aligned} \tag{47}$$

$$Loc(V0, S4)$$

In addition, we need to move the At() fluents forward for all cases, but we will show only for V3 using 23

$$\begin{aligned} S1 = Result(Traverse(E2), S0), At(V3, 1, S0), Loc(V0, S0), \\ At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \rightarrow \end{aligned} \tag{48}$$

$$At(V3, 1, S1)$$

$$\begin{aligned} S2 = Result(Traverse(E3), S1), At(V3, 1, S1), Loc(V2, S1), \\ At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \rightarrow \end{aligned} \tag{49}$$

$$At(V3, 1, S2)$$

At next state (23) will not hold, however, (20) does hold, and thus,

$$\begin{aligned} S3 = Result(Traverse(E3), S2), At(V3, 1, S2), Loc(V3, S2), \\ \neg Loc(v, Result(a, s)) \vee At(v, 0, Result(a, s)) \rightarrow \end{aligned} \tag{50}$$

$$At(V3, 0, S3)$$

Now we need to handle the carrying fluent. The same procedure is applied as with the At fluent. To conserve time and space, we will skip this process and jump to the fluent at state S4, i.e.,

$$Carrying(1, S4)$$

The final step for proving the people are safe is performed using (21). Once again, we skipped propagating the safe fluent through states.

$$\begin{aligned} Safe(0, S3), S4 = Result(Tranverse(E2), S3), Loc(v0, S4), Shelter(V0), \neg TimeUp(S4), Carrying(1, S4) \\ \neg Safe(n, s) \vee \neg Loc(v, s) \vee \neg Shelter(v) \vee TimeUp(s) \vee \exists (c \quad Safe(n + c, Result(a, s)) \wedge Carrying(c, s)) \end{aligned} \tag{51}$$

$$Safe(1, S4)$$

We got a contradiction, therefore, the people are safe and sound.

6.3 Remove frame axioms - bad idea

Of course the proof will fail without the frame (Successor state) axioms since we won't have any knowledge of whether there are people at V_3 any more, or will not know what if people are safe, or are we carrying people in the vehicle.

6.4 Proving with forward chaining

Unfortunately, Forward chaining is not an option since there are some non Horn clauses (e.g. (21)) where more than one of the predacits is positive.