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.

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.

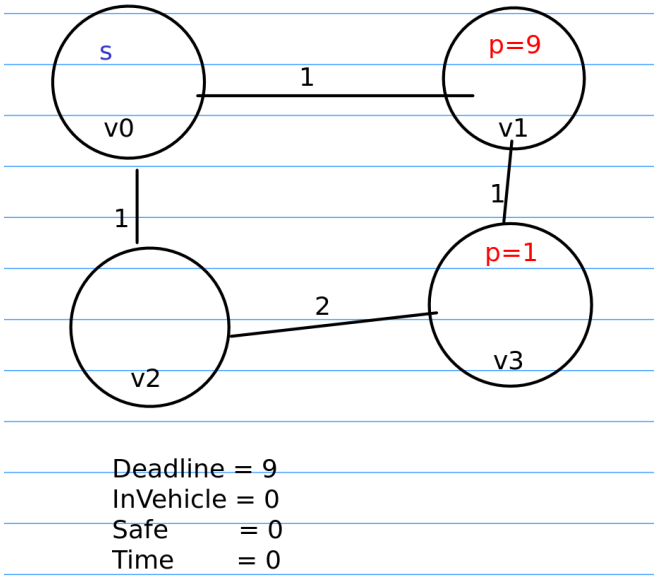


Figure 1: The hurricane environment.

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.

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.

2.2 A^*

Opposed to the greedy agent, the A^* agent will provide an optimal solution since the heuristic is admissible. 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.

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

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

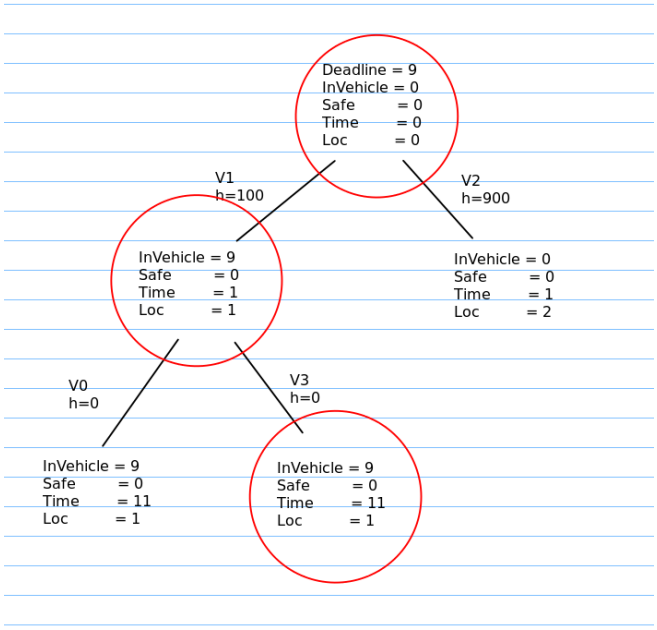


Figure 2: Greedy agent expansions

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 $V_0 \rightarrow V_2 \rightarrow V_3 \rightarrow V_2 \rightarrow V_0$.

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.

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.

[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.

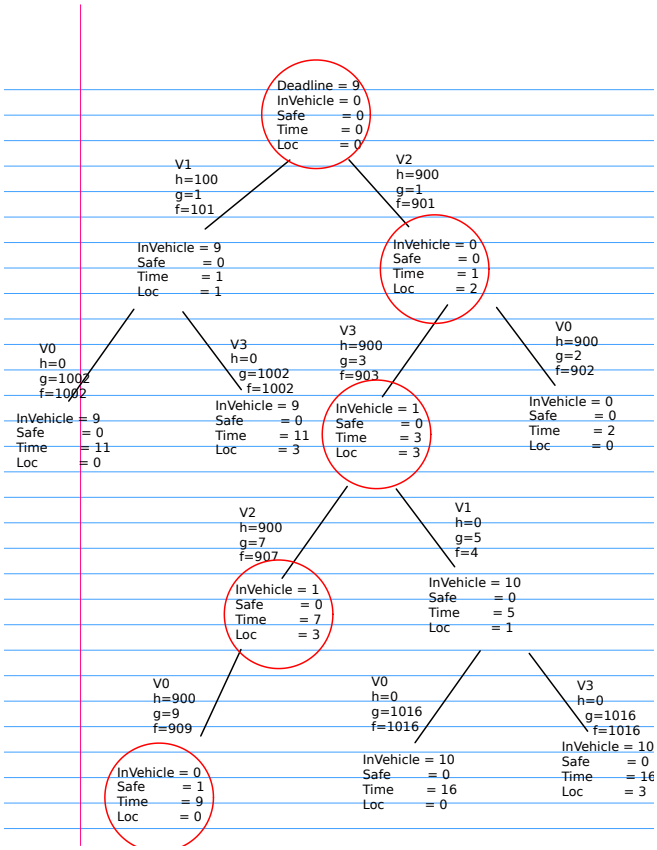
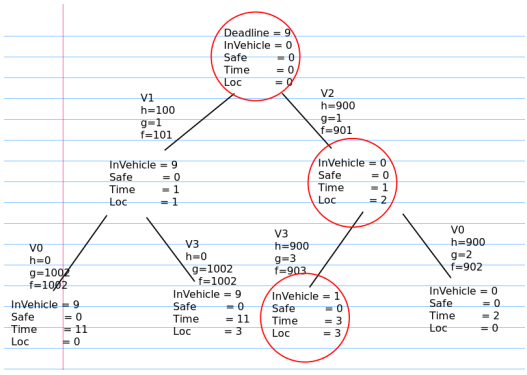
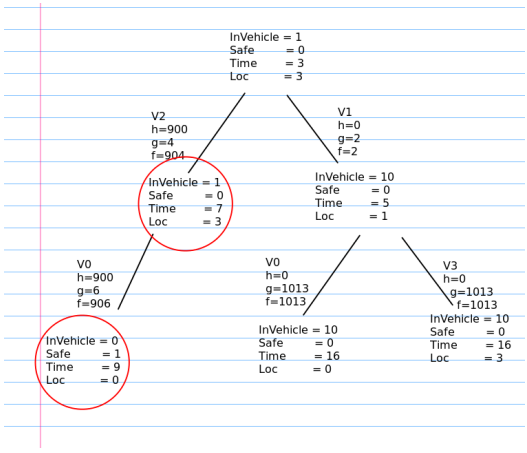


Figure 3: A^* agent expansions



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



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

Exercise 3 Game trees

Exercise 4 Game-tree search - alpha-beta pruning

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, 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.

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 models is 64.

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 64.

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 8.

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

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

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
  $\forall s, t \exists d \quad TimeUp(s) \Rightarrow Time(t, s) > Deadline(d)$
- Bidirectional edge
  $$\forall e, v1, v2 \quad Edge(e, v1, v2) \Leftrightarrow Edge(e, v2, v1)$$

6.1.2 Effect axioms

- Update time fluent
  $$\forall a, s, t \quad (\exists e, w \quad a = Traverse(e) \wedge Weight(e, w)) \wedge Time(t, s) \Rightarrow Time(t + w, Result(a, s))$$
  or,
  $$\forall at, a, s, t \quad a = NoOp() \wedge Time(t, s) \Rightarrow Time(t + 1, Result(a, s))$$

- Loc update

$$\forall s, v1, a, e \quad (\exists v \quad a = Traverse(e) \wedge Edge(e, v, v1) \wedge Loc(v, s)) \Rightarrow Loc(v1, Result(a, s))$$

- Pickup people

$$\forall c, v, a, s \quad Carrying(c, s) \wedge Loc(v, Result(a, s)) \Rightarrow \exists n At(v, n, s) \wedge Carrying(c + n, Result(a, s))$$

- Remove picked up people

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

$$\forall v, a, s \quad Loc(v, Result(a, s)) \Rightarrow At(v, 0, Result(a, s))$$

- Save people

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

$$\forall s, a, v, n \quad Safe(n, s) \wedge Loc(v, s) \wedge Shelter(v) \wedge \neg TimeUp(s) \Rightarrow \exists c \quad Safe(n+c, Result(a, s)) \wedge Carrying(c, s)$$

- Remove carried people

$$\forall s, a, v \quad Loc(v, s) \wedge Shelter(v) \wedge \neg TimeUp(s) \Rightarrow Carrying(0, Result(a, s))$$

6.1.3 Successor state axioms

- Loc unchanged

$$\forall a, s, v \quad Loc(v, s) \Leftrightarrow Loc(v, Result(a, s)) \wedge (a = NoOp())$$

- Number of people in towns unchanged

$$\forall a, v, s \quad At(v, n, s) \wedge \neg Loc(v, s) \Rightarrow At(v, n, Result(a, s))$$

- Number of safe people unchanged

$$\forall a, s \quad Safe(n, s) + (\exists v \quad (Loc(v, s) \wedge \neg Shelter(v)) \Rightarrow Safe(n, Result(a, s))$$

- Number of people carrying unchanged

$$\forall a, s \quad Carrying(n, s) \wedge (\exists v \quad (Loc(v, s) \wedge \neg Shelter(v)) \Rightarrow Carrying(n, Result(a, (cs))$$

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

$$\forall a, s, t \quad (\forall e, w \quad \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s))$$

$$\forall s, v1, a, e \quad (\neg \exists v \quad a = Traverse(e) \wedge Edge(e, v, v1) \wedge Loc(v, s)) \vee Loc(v1, Result(a, s))$$

$$\forall s, v1, a, e \quad (\forall v \quad \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)) \tag{2}$$

$$\forall v, a, s, c \quad \neg Carrying(c, s) \vee \neg Loc(v, Result(a, s)) \vee (\exists n Carrying(n + c, Result(a, s)) \tag{3}$$

$$\forall s, t \exists d \quad \neg TimeUp(s) \vee (Time(t, s) > Deadline(d)) \tag{4}$$

$$\forall e, v1, v2 (\neg Edge(e, v1, v2) \vee Edge(e, v2, v1)) \wedge (Edge(e, v1, v2) \vee \neg Edge(e, v2, v1)) \tag{5}$$

$$\forall a, v, s \quad At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \tag{6}$$

$$\forall e, v1, v2 \quad (\neg Edge(e, v1, v2) \vee Edge(e, v2, v1)) \wedge (Edge(e, v1, v2) \vee \neg Edge(e, v2, v1)) \tag{7}$$

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

$$\forall v, a, s, c \quad \neg Carrying(c, s) \neg Loc(v, Result(a, s)) \vee (\exists n \quad Carrying(n + c, Result(a, s)) \wedge At(v, n, s)) \tag{9}$$

$$\neg Safe(n, s) \vee \neg Loc(v, s) \vee \neg Shelter(v) \vee TimeUp(s) \vee \exists (c \quad Safe(n + c, Result(a, s)) \wedge Carrying(c, s)) \tag{10}$$

6.2.2 What we need to prove

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

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

$$S1 = Result(Traverse(E2), S0)$$
$$S2 = Result(Traverse(E3), S1)$$
$$S3 = Result(Traverse(E3), S2)$$
$$S4 = Result(Traverse(E2), S3)$$
$$TimeOut(S4)$$

$$Loc(V0, S0)$$
$$Carrying(0, S0)$$
$$At(V3, 1, S0)$$
$$At(V1, 9, S0)$$
$$Safe(0, S0)$$
$$Time(0, S0)$$

And more...
1. Proof of Timeup

$$TimeUp(S4), Deadline(9), \neg TimeUp(s) \vee (Time(t, s) > Deadline(d)) \rightarrow Time(t, S4) > Deadline(9) \tag{11}$$

$$Time(0, S0), S1 = Result(Traverse(E2), S0), Weight(E2, 2), \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \tag{12}$$
$$Time(0 + 2, S1) \rightarrow Time(2, S1)$$

$$Time(1, S1), S2 = Result(Traverse(E3), S1), Weight(E3, 1), \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \tag{13}$$
$$Time(2 + 1, S2) \rightarrow Time(3, S2)$$

$$Time(3, S3), S3 = Result(Traverse(E3), S2), Weight(E3, 1), \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \tag{14}$$
$$Time(3 + 1, S3) \rightarrow Time(4, S3)$$

$$Time(4, S3), S2 = Result(Traverse(E2), S3), Weight(E2, 2), \neg a = Traverse(e) \vee \neg Weight(e, w)) \vee \neg Time(t, s) \vee Time(t + w, Result(a, s)) \tag{15}$$
$$Time(4 + 2, S4) \rightarrow Time(6, S4)$$

From CNF 11 and the Time expansion above we get a contradiction, i.e.,

$$Time(6, S4), Time(t, S4) > Deadline(9) \Rightarrow False \tag{16}$$

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,

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

$$\begin{gathered} S1 = Result(Traverse(E2), S0), Loc(V0, S0), Edge(E2, V0, V2), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)) \end{gathered} \tag{17}$$

$$Loc(V2, S1)$$

$$\begin{gathered} S2 = Result(Traverse(E3), S1), Loc(V2, S1), Edge(E3, V2, V3), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)) \end{gathered} \tag{18}$$

$$Loc(V3, S2)$$

$$\begin{gathered} S3 = Result(Traverse(E3), S2), Loc(V3, S2), Edge(E3, V2, V3), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)), and(7) \end{gathered} \tag{19}$$

$$Loc(V2, S3)$$

$$\begin{gathered} S4 = Result(Traverse(E2), S3), Loc(V2, S3), Edge(E2, V0, V2), \\ \neg(a = Traverse(e)) \vee \neg Edge(e, v, v1) \vee \neg Loc(v, s)) \vee Loc(v1, Result(a, s)), and(7) \end{gathered} \tag{20}$$

$$Loc(V0, S4)$$

In addition, we need to move the At() fluents forward for all cases, but we will show only for V3 using 6

$$\begin{gathered} S1 = Result(Traverse(E2), S0), At(V3, 1, S0), Loc(V0, S0), \\ At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \rightarrow \end{gathered} \tag{21}$$

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

$$\begin{gathered} S2 = Result(Traverse(E3), S1), At(V3, 1, S1), Loc(V2, S1), \\ At(v, n, Result(a, s)) \vee \neg At(v, n, s) \vee \neg Loc(v, s) \rightarrow \end{gathered} \tag{22}$$

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

At next state (6) will not hold, however, (8) does hold, and thus,

$$\begin{gathered} S3 = Result(Traverse(E3), S2), At(V3, 1, S2), Loc(V3, S2), \\ \neg Loc(v, Result(a, s)) \vee At(v, 0, Result(a, s)) \rightarrow \end{gathered} \tag{23}$$

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

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 (10). Once again, we skipped propagating the safe fluent through states.

$$\begin{gathered} Safe(0, S3), S4 = Result(Tranverse(E2), S3), Loc(v0, S4), Shelter(V0), \neg TimeUp(S4), Carrying(1, S4) \\ \neg Safe(n, s) \vee \neg Loc(v, s) \vee \neg Shelter(v) \vee TimeUp(s) \vee \exists(c \quad Safe(n + c, Result(a, s)) \wedge Carrying(c, s)) \end{gathered} \tag{24}$$

$$Safe(1, S4)$$

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. (10)) where more than one of the predacits is positive.