CS 323, Fall 2000
Problem Set #3 Solutions:
Situation Calculus and the Frame Problem
Handout #17

---

1. [**10 points**] Prove that arboreal situation calculus defined by axioms 1, 2, and 3 has no loops.

   Solution: Here are the axioms we need:

$$\forall a_1, a_2, s_1, s_2.Result(a_1, s_1) = Result(a_2, s_2) \implies (a_1 = a_2 \wedge s_1 = s_2) \tag{1}$$

$$\forall a, s.\neg Result(a, s) = S0 \tag{2}$$

$$\forall \phi.[(\phi(S0) \wedge \forall a, s.\phi(s) \implies \phi(Result(a, s))) \implies \forall s.\phi(s)] \tag{3}$$

Let's assume that there is a loop, and then show a contradiction. The basic idea behind the proof is that you "roll" the loop backwards towards $S0$ using axiom 1. You know you will eventually hit $S0$, because of equation 3. (You're not on a chain (infinite or not) that doesn't attach back to $S0$.) The final blow to the contradiction happens at axiom 2, because the loop will end up rolling right on top of $S0$, which isn't possible, as no situation is allowed to point to it.

Here's picture of what happens:

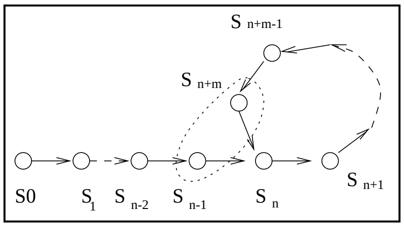

Figure 1: A Loop, at Situation S_n of size m.

Since $Result(a_{n+m}, S_{n+m}) = Result(a_{n-1}, S_{n-1}) = S_n$ (The actions are not labeled for simplicity.), we know then that $S_{n+m} = S_{n-1}$. This is depicted below:

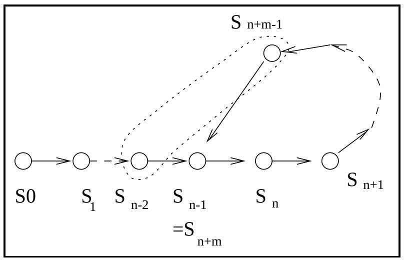

Figure 2: A Loop, at Situation S_n of size $m+1$, Rolled Back One Step.

You can verify that the loop remains of size $m+1$, and that situations in the loop with index i will get mapped to the situation indexed by $i-(m+1)$. Eventually, what is going is happen is that we will approach $S0$, shown below:

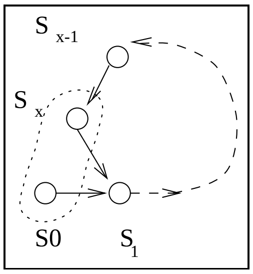

Figure 3: A Loop, Approaching $S0$.

After the next application of axiom 1, we get:

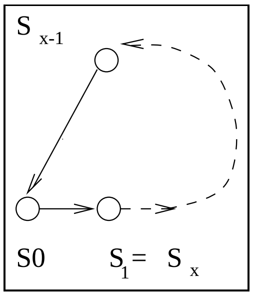

Figure 4: A Loop, with S_{x-1} Pointing to $S0$.

But this is not allowed, by axiom 2. So we've shown that loops can't exist.

We needed the induction axiom to show that there can't be loops in arboreal situation calculus. Otherwise you could have a trivial counterexample with two situations $s1$ and $s2$ pointing to each other, as shown in Figure 5. As long as neither is connected to $S0$, you would have no problem. This should point out how the induction axiom forces all situations to come from $S0$.

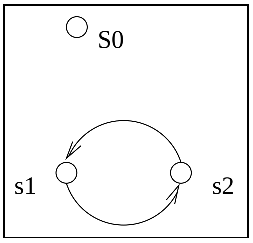

Figure 5: Counterexample that Works if there Is No Induction Axiom.

Common Mistakes:

(a) -1: For a loop, assumes $\exists s, s_a, s_b, a, b. s = Result(a, s_a) = Result(b, s_b)$, but no justification for why $s_a \neq s_b$.

(b) -1, -2,-3: Implicitly assumes axiom 3.

(c) -1,-2: Unclear.

(d) -2: Unsound steps in proof.

(e) -3: Incomplete definition of loop.

The motivation of this problem is not only to get yourself familiar with situation calculus and proofs, but also to understand how hard it is to axiomatize domains. It is not obvious that axioms 1 and 2 aren't enough to get a tree rooted in $S0$.

2. [**10 points**] Say we have A actions and F fluents in our language. How many unique situations are possible in the arboreal interpretation? What if we treated situations as states of fluents? Don't forget the cases where $A = 0$ and $F = 0$.

Solution: If $A = 0$, then there is one unique situation possible in the arboreal interpretation; otherwise there is an infinite number. (You can create a bijection from each situation to the sequence of actions that created it from $S0$ – these action sequences can be thought of as strings, of which there are countably many.)

In the situation-state interpretation, if every situation were completely specified, there would be 2^F different situations. If $F = 0$ then we only get one distinct situation.

There are a lot more situations in the arboreal version!

Common Mistakes:

(a) -0: There was an observation by many of you about the fact that when $A = 0$, there are still 2^F situations, you just can't "get" to them. The point is that situations are distinguished/identified differently in arboreal vs. situation-state versions, one by the actions which comprise it, and the other by its fluents. Accessibility has nothing to do with how you identify situations.

(b) -0: Others pointed out that when $F = 0$, there might be no situation. This is a little confusing, something along the lines of the question "Is the empty set a set?" When $F = 0$ there should be precisely one situation-state S_ϕ, where nothing holds. Any other situation would be identified as equal to this one, since there are no fluents to distinguish it from S_ϕ.

(c) -1: Confusions regarding upper/lower bounds.

(d) -2: Not recognizing $n = infinity$ when $A > 0$.

(e) -3: Left out $A = 0$ case.

3. **[10 points]** Pick one of the solutions in [GLR91] and explain it in your own words. This paper is available in class or outside Gates 218.

Solution: I guess I could just refer you to [GLR91] as the solution, but a nice summary here is a good exercise.

(a) **8.2.3 Actions which restore the world to its original state/involve no activity.** It seems as if actions that restore the world to its original state, such as $Move(A, Table, Table)$, or do not change the world, like $Wait$, should result in the same situation as before. But one thing definitely different before and after the action is time. So you really do not want the before and after situations to be the same.

Then actions which have no effects should be handled either by frame axioms or some other solution to the frame problem.

(b) **8.2.1 Partially Ordered Events.** We can specify at most a partial order of events using nondeterminism. In the example, we can have a partial order of events e_1 and e_2 by saying: $\exists a_1, a_2.(F(a_1, a_2) \wedge s = Result(a_1, Result(a_2, S_0)))$, where $F(a_1, a_2) \iff (a_1 = e_1 \wedge a_2 = e_2) \vee (a_1 = e_2 \wedge a_2 = e_1)$

(c) **8.2.1 Incomplete Information about a Situation.** (to get more information about the past) Here we can use the same trick as in specifying partially ordered events. You may have incomplete information about an event. The trick here is to existentially quantify out the information which is missing. If you know a block was moved but not which one, say $\exists b.s = Result(Move(b, Table), S0$.

Since fluents and actions are reified in situation calculus, almost incomplete information is "existentially quantifiable".

(d) **8.2.2 Duration of Actions.** (Slow vs. fast actions) For durations we obviously need to formalize time, with such axioms as $\forall a.Duration(a) > 0$ and $\forall a, s.Time(Result(a, s)) = Time(s) + Duration(a)$. Then we can talk about the duration of actions, which in some respect talks about whether they happen quickly or not.[1]

Note that this fix is purely additive.

(e) **8.2.5 Actions with Delayed Effects, Natural Death.** We can use our handy $Duration$ function to talk about both actions with delayed effects, and those with a natural death. For delayed effects, we write axioms such as: $\forall a, s.Duration(a) = 30 \implies Holds(Green, Result(Press; a, s))$. To do natural death, we can write axioms such as $\forall a, s.Duration(a) \leq 30 \implies Holds(EmitsAir, Result(Press; a, s))$.

(f) **8.2.6 Simultaneous/Overlapping/Concurrent Actions.** We can add another operator "+", which creates the simultaneous action $a_1 + a_2$ out of two actions a_1 and a_2. We can use this operators along with ";" and $Wait$ to get arbitrary combinations of actions. We would have to add a lot of extra machinery though, to decide the effects of such concurrent actions, and this is the subject of section 8.3.

(g) **8.2.4 Continuous Processes.** We can add more to our ontology about situations by introducing the operator ";" which refers to a sequence of actions, and $Head$ and $Tail$ which talk about performing the beginning or end of an action for a certain

[1] As mentioned in the paper, we can't say anything about the uniformity of the motion.

amount of time. The axioms formalizing these notions using the concept of *Duration* are listed in the paper.

One can formalize continuous processes by stating their values are a mathematical function of time, i.e. $x = v \times t + x_0$.

Common Mistakes: 1-3 points off if your summary was too brief, or had some omissions.

4. [**25 points**] Here are the logical equations again for the Yale Shooting Problem:

$$\begin{array}{l}
Holds(Alive, S0) \\
\neg Holds(Loaded, S0) \\
\\
\forall s. Holds(Loaded, Result(Load, s)) \\
\forall s. Holds(Loaded, s) \implies \neg Holds(Alive, Result(Shoot, s)) \\
\forall s. \neg Holds(Loaded, Result(Shoot, s)) \\
\\
\forall a, f, s. \neg Ab(a, f, s) \implies (Holds(f, s) \equiv Holds(f, Result(a, s)))
\end{array} \tag{4}$$

(a) [**10 points**] Enumerate the remaining eleven unintended models of $Circ[\Sigma; Ab; Holds]$, by trying the other fifteen possibilities for the unspecified intervals. (Four of the fifteen won't be legal because they won't obey the fourth axiom in 4.) Draw them all out in the same form as shown. (Yes it is a pain but it is good for you!)

Solution: Here they are:

These first four are models where the gun is unloaded during the *Shoot* action, as intended. Model 1 is the intended model.

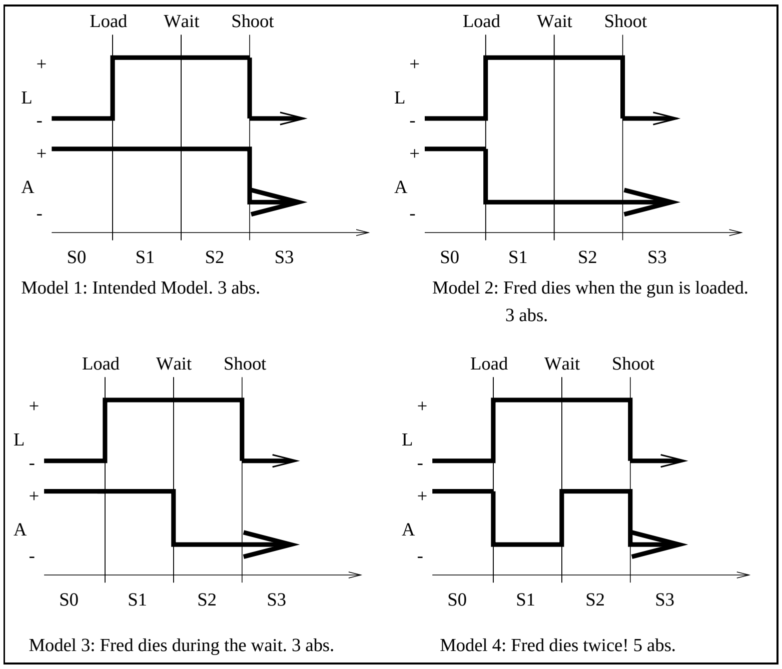

Figure 6: Four Models of the YSP, where the gun unloads at *Shoot*.

The next eight are the models where the gun unloads during the $Wait$ action, which is unintended.

(b) [**10 points**] By inspection, you should be able to draw a Hasse diagram based on the ordering $\leq_{Ab;Holds}$ on all twelve models. [Hint: Since $Holds$ is varied, you don't need to worry about the values of *Loaded* and $Alive$. All you have to do is order the models based on subset inclusion of the *Ab*s, which exactly correspond to transitions in the time functions.]

[Hint #2: You can split the twelve models into two groups which are mutually incomparable if you look at the *Ab*s from the behavior of *Loaded* – four will have one *Loaded* function while eight will have another, and the *Ab*s in each are not inclusive of the other.]

[Hint #3: One set will look like a tree and the other a cube.]

Solution: We automatically know that the four models where the gun unloads during the shoot is incomparable under $le_{Ab;Holds}$ with the eight models where the gun unloads during the $Wait$. This is because the first set has $Ab(Shoot, Loaded, S2)$, while the second set has $Ab(Wait, Loaded, S1)$, and neither set subsumes the other.

This insight should make the comparison a lot easier. Here is the Hasse diagram of the models, with numbers from the figures above.

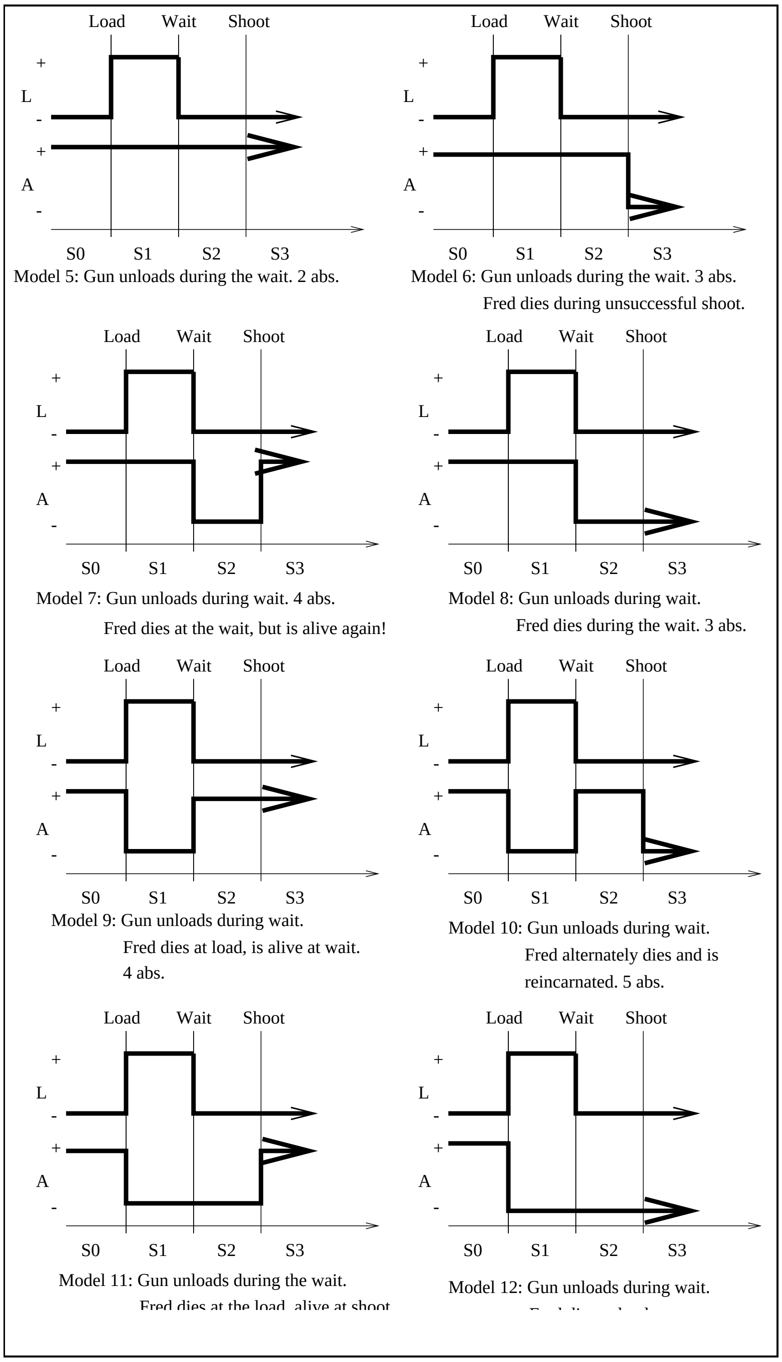

Figure 7: Eight Models of the YSP, where the gun unloads at *Wait*.

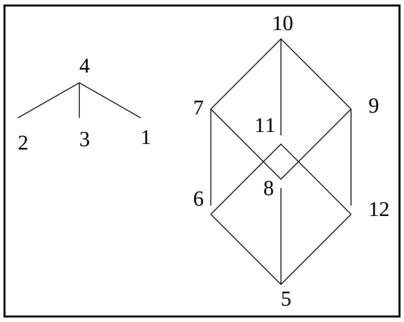

Figure 8: Hasse diagram of all twelve models of the YSP.

(c) [**5 points**] We can read off the minimal models by inspection of the Hasse diagram. It turns out there are four Ab-minimal models. For each, note how many Abs there are, and give a brief English description of each.

Solution: We we draw out the models as such, we see that there are indeed four ab-minimal models. They are indexed: 5, 1, 2, and 3.

i. Model 1 has 3 abs. This is the intended model where the gun unloads at the $Shoot$ action, and Fred dies as a result of being shot.

ii. Model 2 has 3 abs. Here, the gun unloads when it is supposed to, but Fred dies when the gun is loaded.

iii. Model 3 has 3 abs. Once again, the gun unloads when it is supposed to, but this time Fred dies during the wait.

iv. Model 5 has only 2 abs. The gun gets unloaded during the wait, so that Fred is saved! This unintended model prevents us from concluding that Fred will die as a result of Loading, Waiting and then Shooting.

Common Mistakes:

i. -0: Some of you pointed out some of the 11 unintended models were "odd." This is the point! Our axiomatization is not strong enough to disallow these unintended models. Isn't that depressing?

ii. : -1: Small errors.

5. [**10 points**]

Let's enumerate the $Abextent(s, M)$ for each model:

	$S0$	$S1$	$S2$
***1**	$\{\langle Load, Loaded\rangle\}$		$\{\langle Shoot, Loaded\rangle, \langle Shoot, Alive\rangle\}$
2	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$		$\{\langle Shoot, Loaded\rangle\}$
3	$\{\langle Load, Loaded\rangle\}$	$\{\langle Wait, Alive\rangle\}$	$\{\langle Shoot, Loaded\rangle\}$
4	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$	$\{\langle Wait, Alive\rangle\}$	$\{\langle Shoot, Loaded\rangle, \langle Shoot, Alive\rangle\}$
5	$\{\langle Load, Loaded\rangle\}$	$\{\langle Wait, Loaded\rangle\}$	
6	$\{\langle Load, Loaded\rangle\}$	$\{\langle Wait, Loaded\rangle\}$	$\{\langle Shoot, Alive\rangle\}$
7	$\{\langle Load, Loaded\rangle\}$	$\{\langle Wait, Loaded\rangle, \langle Wait, Alive\rangle\}$	$\{\langle Shoot, Alive\rangle\}$
8	$\{\langle Load, Loaded\rangle\}$	$\{\langle Wait, Loaded\rangle, \langle Wait, Alive\rangle\}$	
9	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$	$\{\langle Wait, Loaded\rangle, \langle Wait, Alive\rangle\}$	
10	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$	$\{\langle Wait, Loaded\rangle, \langle Wait, Alive\rangle\}$	$\{\langle Shoot, Alive\rangle\}$
11	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$	$\{\langle Wait, Loaded\rangle\}$	$\{\langle Shoot, Alive\rangle\}$
12	$\{\langle Load, Loaded\rangle, \langle Load, Alive\rangle\}$	$\{\langle Wait, Loaded\rangle\}$	

The intended model is 1. We can read right off the bat, that:

$$\forall i \in \{2,4,9,10,11,12\}.Abextent(S0, M_1) \subset Abextent(S0, M_i).$$

Thus, $S0$ satisfies (b)i, while every other situation (which is after $S0$) is covered by (b)ii. Hence $M_1 \leq_{chron} M_i$, for $i \in \{2,4,9,10,11,12\}$.

For $i \in \{3,5,6,7,8\}$, we notice that $Abextent(S0, M_1) = Abextent(S0, M_i)$, and $Abextent(S1, M_1) = \phi \subset Abextent(S1, M_i)$. So, $S0$ and $S1$ are definitely covered by (b)i, while any situation following $S1$ is covered by (b)ii and $S1$. Hence $M_1 \leq_{chron} M_i$, for $i \in \{3,5,6,7,8\}$.

Common Mistakes:

(a) -0: Many people associated the s in $Abextent(s)$ with the wrong situation. For example, they said $Abextent(S0, M_1) = \{\}$, while $Abextent(S1, M_1) = \{\langle Load, Loaded\rangle\}$, $Abextent(S2, M_1) = \{\}$, $Abextent(S3, M_1) = \{\langle Shoot, Loaded\rangle, \langle Shoot, Alive\rangle\}$ – basically shifting the $Abextent$s over by one.

(b) -5: Very unclear/confused.

6. [**5 points**] Write the four axioms needed for causal minimization in first-order logic, using only the predicates $Affects$, $Causes$, $Holds$, $Success$, $Precond$, and $=$.

Solution: You could probably just read this off from the English description (or the book).

$$\forall a, s, f, v.(Success(a, s) \wedge Causes(a, f, v)) \implies (Holds(f, Result(a, s)) \iff v = True)$$

$$\forall a, f, s.\neg Affects(a, f, s) \implies (Holds(f, Result(a, s)) \iff Holds(f, s))$$

$$\forall a, s.Success(a, s) \equiv \forall f.(Precond(f, v, a) \implies (Holds(f, s) \iff v = True))$$

$$\forall a, f, s.Affects(a, f, s) \equiv Success(a, s) \wedge \exists v.Causes(a, f, v) \tag{5}$$

Common Mistakes

(a) -0: In response to my email a lot of you changed your answers from the form of $Holds(f = v, s)$ to $Holds(f, s) \iff v = True$. I hope yall understand why the first is wrong, and didn't simply change it because I said so. Once again, $f = v$ is a proposition, which stands by itself, and is *not* a fluent. f is a fluent.

(b) -1: Assorted errors, including the wrong axiomatization, or leaving out quantifiers.

(c) -2: Circular definitions.

(d) -3: Treating the variable v, which can take on values "true" or "false," as a logical symbol. You should be able to avoid this error because v shows up in the logic as a term (we have $Causes(a, f, v)$ so you know that it can't also be a proposition.

(e) -5: You wrote out some other axioms.

7. [**10 points**] "Everybody loves my baby, but my baby only loves me." Formalize this in first-order logic, and then use Otter to show that $Me = Baby$.

Solution: Here is my file:

```
% Everybody loves my baby, but my baby only loves me.
% Tue Jun 22 18:06:39 1999 This proves that if the above
% statement is true, then I am my baby.

set(auto).
formula_list(usable).

x = x.
all x (loves(x, baby)).
loves(baby, me).
all x (loves(baby, x) -> x = me).
-(baby = me).
end_of_list.
```

The proof is:

```
1 [] -loves(baby,x)|x=me.
2 [] baby!=me.
3 [copy,2,flip.1] me!=baby.
5 [] loves(x,baby).
7 [hyper,5,1,flip.1] me=baby.
9 [binary,7.1,3.1] $F.
```

We can read off the proof as so:

$\neg loves(baby, x) \vee x = me$	(1) Given.
$\neg baby = me$	(2) Negation of Conclusion.
$\neg me = baby$	(3) Symmetry on (2)
$loves(y, baby)$	(5) Given.
$me = baby$	(7) Resolve (1) & (5):, flip $y \leftarrow baby, x \leftarrow baby$
$\perp$	(9) Resolve (3) & (7)

Intuitively, if everyone loves *Baby*, then of course *Baby* loves *Baby*. But if *Baby* loves anyone, it can only be Me. So $Me = Baby$.

8. [**10 points**] Write the explanation closure axioms for the Yale Shooting Problem in otter style. Combine with the initial observations and effect axioms (everything but the last axiom in 4). Use otter to prove that $\neg Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0))))$.

Solution:

```
% Yale Shooting Problem
% Using Explanation Closure Style Axioms
% Tue Nov  2 05:22:48 1999

set(auto).
formula_list(usable).

all _x (_x = _x).

% UNA
-(Load = Shoot).
-(Load = Wait).
-(Shoot = Wait).
-(Alive = Loaded).

% Initial sentences
Holds(Alive, S0).
-Holds(Loaded, S0).

% Effect axioms
all _s (Holds(Loaded, Result(Load, _s))).
all _s (Holds(Loaded, _s) -> -Holds(Alive, Result(Shoot, _s))).
all _s (-Holds(Loaded, Result(Shoot, _s))).

% Explanation Closure Axioms
all _a _s ((Holds(Loaded, _s) & -Holds(Loaded, Result(_a, _s))) -> _a
= Shoot).
all _a _s ((-Holds(Loaded, _s) & Holds(Loaded, Result(_a, _s))) -> _a
= Load).
all _a _s ((Holds(Alive, _s) & -Holds(Alive, Result(_a, _s))) ->
        (_a = Shoot & Holds(Loaded, _s))).
all _a _s (Holds(Alive, Result(_a, _s)) -> Holds(Alive, _s)).

% Negation of conclusion
Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0)))).

end_of_list.
```

The proof:

```
5 [] Shoot!=Wait.
```

```
6 [copy,5,flip.1] Wait!=Shoot.
10 [] -Holds(Loaded,x)| -Holds(Alive,Result(Shoot,x)).
12 [] -Holds(Loaded,x)|Holds(Loaded,Result(y,x))|y=Shoot.
19 [] Holds(Loaded,Result(Load,x)).
20 [] Holds(Alive,Result(Shoot,Result(Wait,Result(Load,S0)))).
27 [hyper,19,12] Holds(Loaded,Result(x,Result(Load,y)))|x=Shoot.
81 [hyper,27,10,20] Wait=Shoot.
83 [binary,81.1,6.1] $F.
```

$\neg Shoot = Wait$	(5) Given.
$\neg Wait = Shoot$	(6) Symmetry on (5).
$\neg Holds(Loaded, x) \vee \neg Holds(Alive, Result(Shoot, x))$	(10) Given.
$\neg Holds(Loaded, x) \vee Holds(Loaded, Result(y, x)) \vee y = Shoot$	(12) Given.
$Holds(Loaded, Result(Load, z))$	(19) Given.
$Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0))))$	(20) Negation of Conclusion.
$Holds(Loaded, Result(y, Result(Load, z))) \vee y = Shoot$	(27) Resolve (12) & (19): $x \leftarrow Result(Load, z)$.
$Wait = Shoot$	(81) Multiple resolutions: (10), (20), & (27): $x \leftarrow Result(Wait, Result(Load, S0))$, $y \leftarrow Wait, z \leftarrow S0$
$\perp$	(83) Resolve (6) & (81)

Here is a description of this proof. We are given that (12): If a gun is loaded, and the action is not *Shoot*, then the gun will stay loaded. We take that fact along with (19): After you load a gun it is loaded. From these two axioms we infer (27): As long as an action y isn't *Shoot*, loading the gun, then doing y will ensure that the gun stays loaded. Then we resolve (10), (20) and (27). When we take (10) and (20) together we get the fact that if someone is alive after *Load*, $Wait$, *Shoot*, then the gun wasn't loaded in $Result(Wait, Result(Load, S0))$. But the only way this can happen, according to (27) is if the action after the *Load*, $Wait = Shoot$. But this isn't possible by the UNA (5,6).

Common Mistakes:

(a) -1: Random errors, like writing an explanation closure axiom wrong.

9. [**10 points**] Write the successor state axioms for the Yale Shooting Problem in otter style. Combine with the initial observations. Use otter to prove that $\neg Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0))))$.

Solution:

```
% Yale Shooting Problem
% Using Successor State Style Axioms
% Tue Nov  2 06:16:45 1999

set(auto).
formula_list(usable).
```

```
all _x (_x = _x).

% UNA
-(Load = Shoot).
-(Load = Wait).
-(Shoot = Wait).
-(Alive = Loaded).

% Initial sentences
Holds(Alive, S0).
-Holds(Loaded, S0).

% Successor State:
all _a _s (Holds(Alive, Result(_a, _s)) <->
        (Holds(Alive, _s) & -(_a = Shoot & Holds(Loaded, _s)))).

all _a _s (Holds(Loaded, Result(_a, _s)) <->
        ((_a = Load) | (Holds(Loaded, _s) & -(_a = Shoot &
Holds(Loaded, _s))))).

% Negation of conclusion
Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0)))).

end_of_list.
```

Here is the proof:

```
5 [] Shoot!=Wait.
6 [copy,5,flip.1] Wait!=Shoot.
11 [] -Holds(Alive,Result(x,y))|x!=Shoot| -Holds(Loaded,y).
16 [] Holds(Loaded,Result(x,y))|x!=Load.
17 [] Holds(Loaded,Result(x,y))| -Holds(Loaded,y)|x=Shoot.
18 [] x=x.
20 [] Holds(Alive,Result(Shoot,Result(Wait,Result(Load,S0)))).
21 [hyper,18,16] Holds(Loaded,Result(Load,x)).
31 [hyper,21,17] Holds(Loaded,Result(x,Result(Load,y)))|x=Shoot.
126 [hyper,31,11,20,18] Wait=Shoot.
128 [binary,126.1,6.1] $F.
```

$\neg Shoot = Wait$	(5) Given.
$\neg Wait = Shoot$	(6) Symmetry on (5).
$\neg Holds(Alive, Result(x, y)) \lor \neg x = Shoot \lor \neg Holds(Loaded, y)$	(11) Given.
$Holds(Loaded, Result(w, u)) \lor \neg w = Load$	(16) Given.
$Holds(Loaded, Result(v, s)) \lor \neg Holds(Loaded, s) \lor v = Shoot$	(17) Given.
$z = z$	(18) Reflexivity.
$Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S0))))$	(20) Negation of conclusion.
$Holds(Loaded, Result(Load, u))$	(21) Resolve (16) & (18): $z \leftarrow Load, w \leftarrow Load.$
$Holds(Loaded, Result(v, Result(Load, u))) \lor v = Shoot$	(31) Resolve (17) & (21): $s \leftarrow Result(Load, u)$
$Wait = Shoot$	(126) Multiple resolutions (31), (11), (20), & (18) $x \leftarrow Shoot, z \leftarrow Shoot,$ $y \leftarrow Result(Wait, Result(Load, S0)),$ $v \leftarrow Wait, u \leftarrow S0.$
$\perp$	(128) Resolve (6) & (126)

So in English, what happens is we take the axiom (16) and (18) to get the fact that a gun is loaded after *Load*. Axiom (17) says if a gun is loaded and we do an action other than *Shoot*, it remains loaded. We combine this fact with (21) to derive that (31): after loading, and doing action v, the gun stays loaded, provided $v \neq Shoot$. Then for (126) we perform multiple inferences. If Fred is alive after $Load, Wait, Shoot$ (20), then the gun wasn't loaded at $Result(Wait, Result(Load, S0))$ (11). But then by (31) (if the gun is loaded, Fred's dead after a *Shoot*), this means $Wait = Shoot$, which of course is a contradiction.

Common Mistakes

(a) -0: Left in effect axioms along with successor state. The successor state axioms imply the effect ones. This is the whole point behind the successor state approach!

(b) -1: Small errors.

References

[GLR91] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus? In Robert Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 167–179. Kluwer Academic, Dordrecht, 1991.