
Example Theory: The Yale Shooting Problem

- Primitive actions: *load, shoot, wait*
- Fluents: *alive, loaded*
- Action Precondition Axioms \mathcal{D}_{ap} :

$$Poss(load, s) \equiv \neg loaded(s) \quad (P1)$$

$$Poss(shoot, s) \equiv loaded(s) \quad (P2)$$

$$Poss(wait, s) \equiv True \quad (P3)$$

- Effect Axioms:

$$loaded(do(load, s))$$

$$\neg alive(do(shoot, s))$$

$$\neg loaded(do(shoot, s))$$

- Normal Form Effect Axioms:

$$a = load \supset loaded(do(a, s))$$

$$a = shoot \supset \neg loaded(do(a, s))$$

$$false \supset alive(do(a, s))$$

$$a = shoot \supset \neg alive(do(a, s))$$

- Successor State Axioms \mathcal{D}_{ss} :

$$loaded(do(a, s)) \equiv a = load \vee loaded(s) \wedge a \neq shoot \quad (SS1)$$

$$alive(do(a, s)) \equiv alive(s) \wedge a \neq shoot \quad (SS2)$$

- Initial State Axioms \mathcal{D}_{S_0} :

$$\neg loaded(S_0) \quad (I1)$$

$$alive(S_0) \quad (I2)$$

- Unique Name Axioms for Actions \mathcal{D}_{una} :

$$load \neq shoot$$

$$shoot \neq wait$$

$$load \neq wait$$

Example: Executability Testing

- In Yale shooting Problem domain, show that

$$\mathcal{D} \models \text{executable}(\text{do}([\text{load}, \text{wait}, \text{shoot}], S_0))$$

- By the corrolary, this holds iff

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[\text{Poss}(\text{load}, S_0) \wedge \text{Poss}(\text{wait}, \text{do}(\text{load}, S_0)) \wedge \text{Poss}(\text{shoot}, \text{do}(\text{wait}, \text{do}(\text{load}, S_0)))]$$

$$\begin{aligned} & \mathcal{R}[\text{Poss}(\text{load}, S_0) \wedge \text{Poss}(\text{wait}, \text{do}(\text{load}, S_0)) \wedge \\ & \quad \text{Poss}(\text{shoot}, \text{do}(\text{wait}, \text{do}(\text{load}, S_0)))] = \\ & \mathcal{R}[\text{Poss}(\text{load}, S_0)] \wedge \mathcal{R}[\text{Poss}(\text{wait}, \text{do}(\text{load}, S_0))] \wedge \\ & \mathcal{R}[\text{Poss}(\text{shoot}, \text{do}(\text{wait}, \text{do}(\text{load}, S_0)))] \end{aligned}$$

$$\mathcal{R}[\text{Poss}(\text{load}, S_0)] = \neg \text{loaded}(S_0)$$

$$\mathcal{R}[\text{Poss}(\text{wait}, \text{do}(\text{load}, S_0))] = \text{True}$$

$$\begin{aligned} & \mathcal{R}[\text{Poss}(\text{shoot}, \text{do}(\text{wait}, \text{do}(\text{load}, S_0)))] = \\ & \mathcal{R}[\text{loaded}(\text{do}(\text{wait}, \text{do}(\text{load}, S_0)))] = \\ & \quad \text{wait} = \text{load} \vee \mathcal{R}[\text{loaded}(\text{do}(\text{load}, S_0))] \wedge \text{wait} \neq \text{shoot} \end{aligned}$$

$$\begin{aligned} & \mathcal{R}[\text{loaded}(\text{do}(\text{load}, S_0))] = \\ & \quad \text{load} = \text{load} \vee \text{loaded}(S_0) \wedge \text{load} \neq \text{shoot} \end{aligned}$$

- Clearly,

$$\begin{aligned} & \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \\ & \quad \neg \text{loaded}(S_0) \wedge \text{True} \wedge \\ & \quad (\text{wait} = \text{load} \vee \\ & \quad (\text{load} = \text{load} \vee \text{loaded}(S_0) \wedge \text{load} \neq \text{shoot}) \wedge \\ & \quad \text{wait} \neq \text{shoot}) \end{aligned}$$

Example: Projection

- In Yale Shooting Problem domain, show that

$$\mathcal{D} \models \neg \text{alive}(\text{do}([\text{load}, \text{wait}, \text{shoot}], S_0))$$

- By the regression theorem, this holds iff

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[\neg \text{alive}(\text{do}([\text{load}, \text{wait}, \text{shoot}], S_0))]$$

$$\begin{aligned} \mathcal{R}[\neg \text{alive}(\text{do}([\text{load}, \text{wait}, \text{shoot}], S_0))] &= \\ \neg(\mathcal{R}[\text{alive}(\text{do}([\text{load}, \text{wait}], S_0))] \wedge \text{shoot} \neq \text{shoot}) & \end{aligned}$$

$$\begin{aligned} \mathcal{R}[\text{alive}(\text{do}([\text{load}, \text{wait}], S_0))] &= \\ \mathcal{R}[\text{alive}(\text{do}(\text{load}, S_0))] \wedge \text{wait} \neq \text{shoot} & \end{aligned}$$

$$\begin{aligned} \mathcal{R}[\text{alive}(\text{do}(\text{load}, S_0))] &= \\ \text{alive}(S_0) \wedge \text{load} \neq \text{shoot} & \end{aligned}$$

- Clearly,

$$\begin{aligned} \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \neg[& ((\text{alive}(S_0) \wedge \text{load} \neq \text{shoot}) \\ & \wedge \text{wait} \neq \text{shoot}) \\ & \wedge \text{shoot} \neq \text{shoot}] \end{aligned}$$

Implementing the Yale Shooting E.g. in Prolog

```
/* load definition of not in Quintus prolog */

:- ensure_loaded(library(not)).

/* Precondition Axioms */

poss(load,S) :- not loaded(S).
poss(shoot,S) :- loaded(S).
poss(wait,S).

/* Successor State Axioms */

loaded(do(A,S)) :- A = load ;
                  loaded(S), not A = shoot.
alive(do(A,S)) :- alive(S), not A = shoot.

/* Initial State Axioms */

alive(s0).
/* can leave out not loaded(s0). */

/* Definition of executable */

executable(s0).
executable(do(A,S)) :- poss(A,S), executable(S).
```

```
tiger 56 % prolog
Quintus Prolog Release 3.2 (Sun 4, SunOS 5.5.1)
Copyright (C) 1994, Quintus Corporation. All rights reserved.
301 East Evelyn Ave, Mountain View, California U.S.A. (415) 254-2800
Licensed to York Univerity, Canada
```

```
| ?- [yisp].
% compiling file /cs/home/fac1/lesperan/yisp.pl
% loading file /cs/local/packages/quintus/generic/qplib3.2/library/not.
% loading file /cs/local/packages/quintus/generic/qplib3.2/library/fre
% freevars.qof loaded in module negation, 0.010 sec 2,868 bytes
% not.qof loaded, 0.020 sec 6,256 bytes
% module negation imported into user
* Singleton variables, clause 3 of poss/2: S
* Approximate line: 7, file: '/cs/home/fac1/lesperan/yisp.pl'
% yisp.pl compiled in module user, 0.040 sec 7,524 bytes
```

```
yes
| ?- not alive(do(shoot,do(wait,do(load,s0))))).
```

```
yes
| ?- alive(s0).
```

```
yes
| ?- alive(do(load,s0)).
```

```
yes
| ?- alive(do(wait,do(load,s0))).
```

```
yes
| ?- loaded(s0).
```

```
no
| ?- loaded(do(load,s0)).
```

```
yes  
| ?- loaded(do(wait,do(load,s0))).
```

```
yes  
| ?- loaded(do(shoot,do(wait,do(load,s0)))).
```

```
no  
| ?- executable(do(shoot,do(wait,do(load,s0)))).
```

```
yes  
| ?-
```