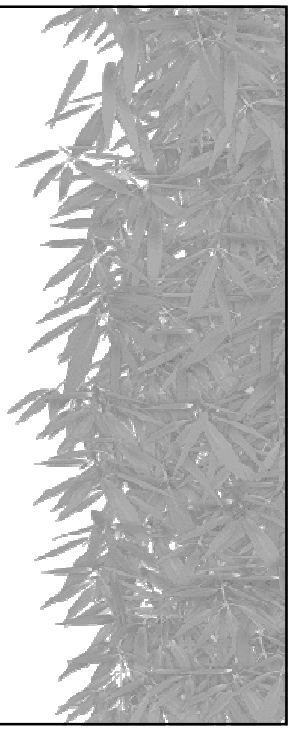
# Learning Action Models from Soar Experience Traces

John Hawkins
Soar Research Group
University of Michigan



# Action Models

- Basic Idea: agents should be able to make plans about taking actions to achieve goals.
- Problem: Action effects are complicated and have bizarre corner cases, so they are hard and time consuming to explicate.
- Solution: We want agents that can learn about actions from experience to better understand their environment.

# Shields in Tanksoar

- Action: move-left
- Battery: 35
- Radar at full power
- Shields on
- Incoming missile from behind
- Will the shields hold out? Why?


# Why is this hard?

- Lots of factors influence how even the simplest aspects of the environment (such as the shields) change over time
- Action effects have a temporal aspect that is hard to observe. (i.e. destroying the enemy with a missile requires repetition and happens after a delay.)

# Inductive Logic Programming

- ILP can handle structured, first-order descriptions of the environment.
- Takes a collection of positive and negative examples and tries to learn rules that cover as many examples as possible without covering more negative examples than necessary.

# Background Knowledge

- The background is essentially just all the sensor information from any state in the range along with a record of which actions were taken
- This information is pulled directly from a soar trace and converted into a first-order logic representation.

- shields_now(1, off ).
- battery(1, 200 ).
- radar(1, 0 ).
- action(1,shields_on).

# Positive&Negative Examples

- The state *before* a predicate becomes true (or false) is a positive (or negative, respectively) example for that predicate.
- This information is also extracted from a soar trace.

**If shields are on at state 2:**

- State 1 is a positive example for shields_next(on).
- State 1 is a negative example for shields_next(off).

# Learning

- Learned Clause: shields_next(state, status)
- Most specific clause is

```
IF not(action(A,shields_on))      and
   not(action(A,shields_off))     and
   shields_now(A,on)              and
   battery_above(A,0)             and
   battery_above(A,5)             and
   battery_below(A,15)
THEN shields_next(A,off)
```

- All learned rules are generalizations of this clause

# Trial Run

- In order to prototype the methods and work out kinks in the various tools, we first created a simplified example by hand, removing soar temporarily from the picture.

# Induced Rules

**EFFECT RULES**

**IF:** action(State,shields_on) **and** battery_above(State,0).
**THEN:** shields_next(State,on)

**IF:** action(State,shields_off).
**THEN:** shields_next(State,off)

# Induced Rules

**FRAME AXIOMS:**

| | | |
|---|---|---|
| **IF:** | not(action(State,shields_on)) | **and** |
| | shields_now(State,off). | |
| **THEN:** | shields_next(State,off) | |
| **IF:** | not(action(State,shields_off)) | **and** |
| | shields_now(State,on) | **and** |
| | battery_above(State,10). | |
| **THEN:** | shields_next(State,on) | |
| **IF:** | battery_below(State,15). | |
| **THEN:** | shields_next(State,off) | |

# Current/Future Work

- Our experiment had good distribution of positive and negative examples. In order to use more complete datasets effectively, we may need to be more selective when choosing positive and negative examples.
- Most background knowledge is irrelevant to any given predicate. May need to provide additional bias to help ILP decide what is important.
- Have tested on a single, simple concept. Long range goal requires learning over numerous concepts.

## Nuggets

- Successfully learned informative rules in hand constructed test domain.

## Coal

- We don't know whether using complete background knowledge and pulling training examples automatically from long soar traces will overwhelm the system.