Redux Update: Building Rules from Examples

Douglas Pearson
ThreePenny Software
douglas.pearson@threepenny.net

John Laird
University of Michigan
laird@umich.edu

Supported by Office of Naval Research

Start Date: Jan 2003

Creating Human-like Behavior is Hard

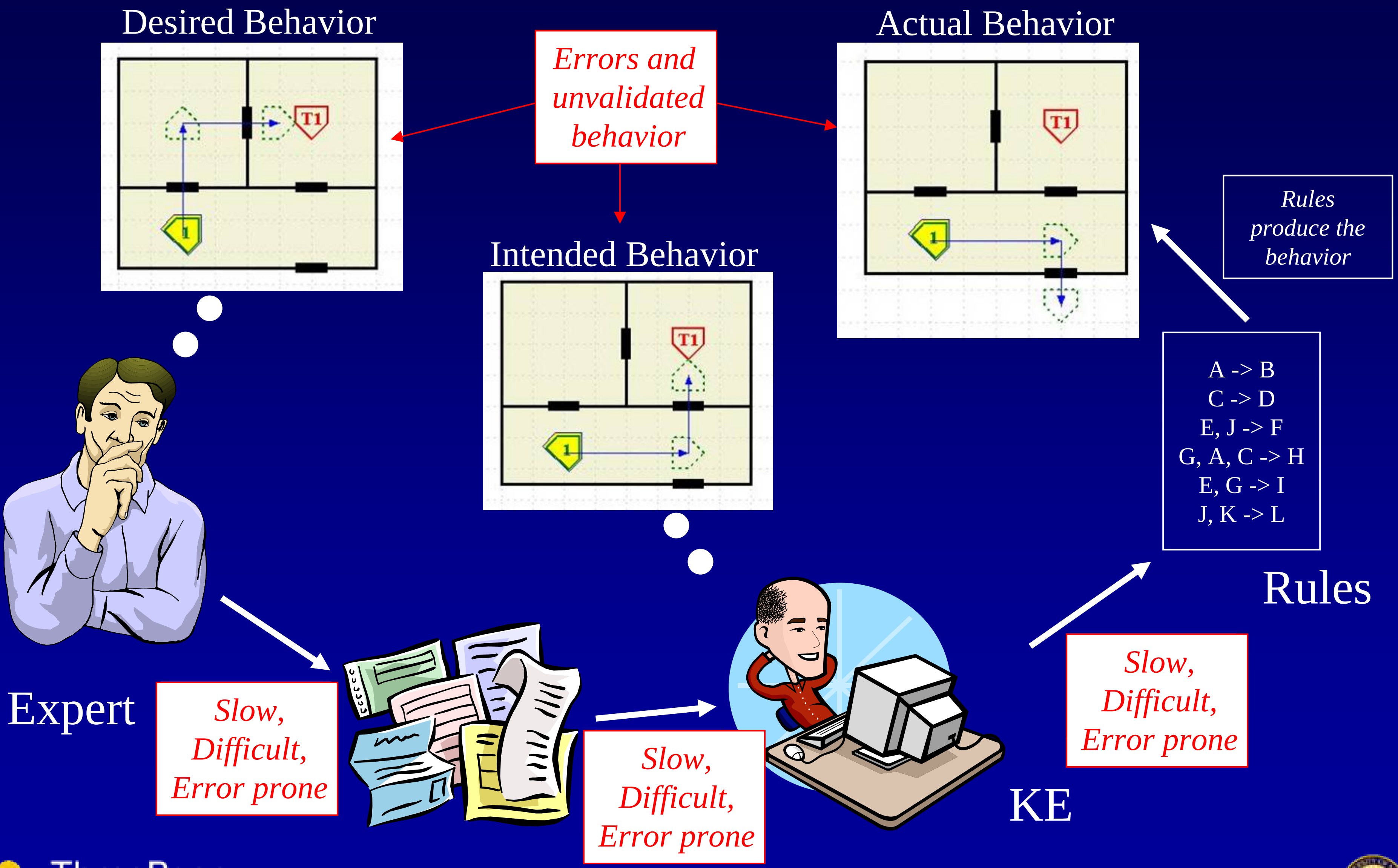
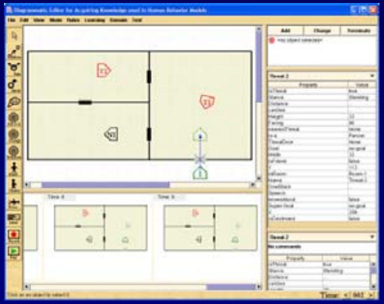
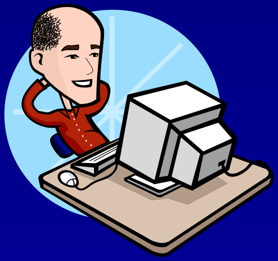

Desired Behavior
T1
1
Errors and unvalidated behavior
Actual Behavior
T1
1
Intended Behavior
T1
1
Rules produce the behavior
A -> B
C -> D
E, J -> F
G, A, C -> H
E, G -> I
J, K -> L
Rules
Expert
Slow, Difficult, Error prone
Slow, Difficult, Error prone
Slow, Difficult, Error prone
KE

Diagram-based Example-driven Development Tool

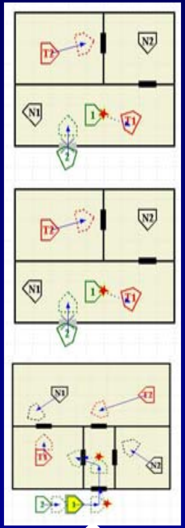
Define behavior with diagram-based examples





Expert KE

Library of validated behavior examples



Analysis & generation tools

- Detect inconsistency
- Generalize
- Generate rules
- Simulate execution
- ⋮

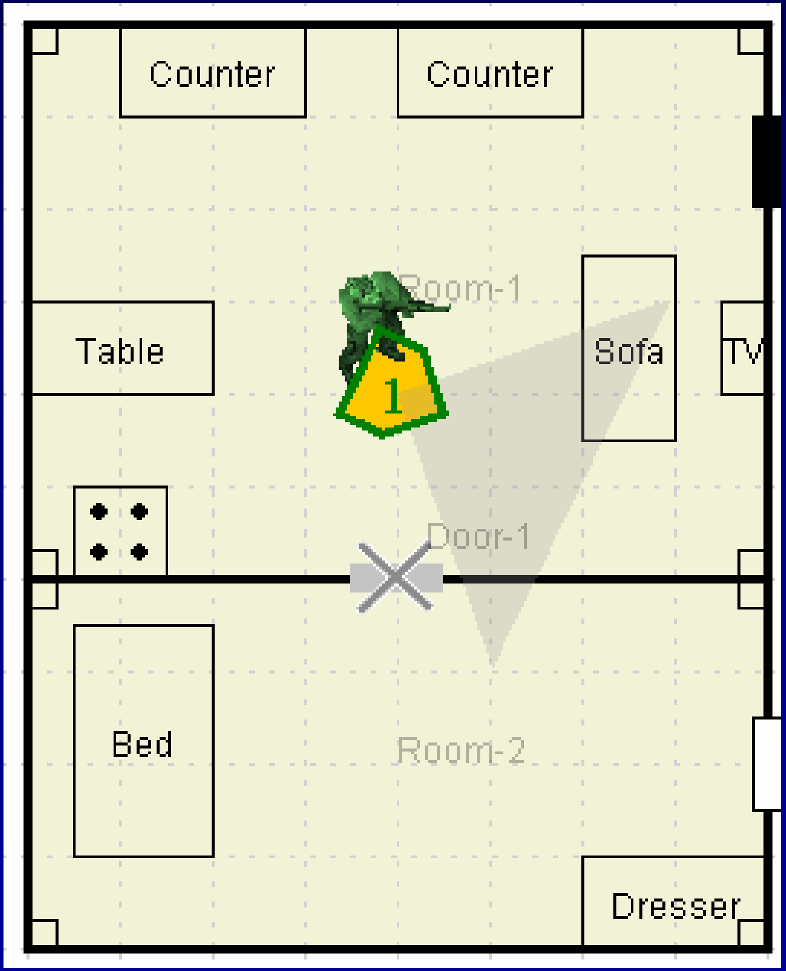
Executable Code

```
A -> B
C -> D
E, J -> F
G, A, C -> H
E, G -> I
J, K -> L
```

Simulation Environment

Define behavior for Breach-Door

- Stage 1:
  - Define correct example of behavior
  - Quick and easy
  - Concrete

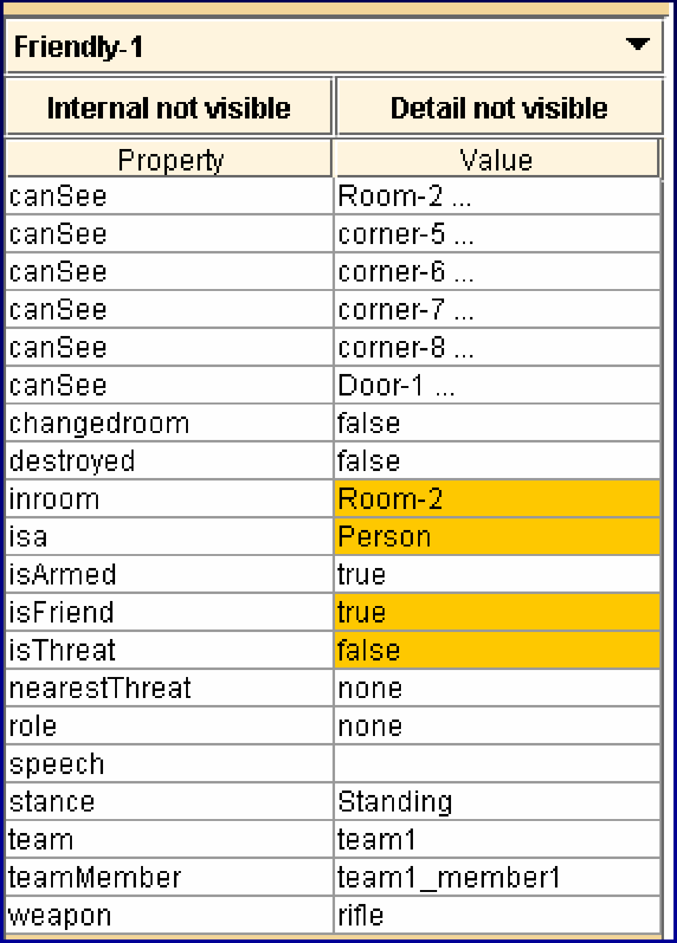


Building rules for Breach-Door

- Stage 2:
  - Create a rule to produce the target behavior
  - If we had a correct rule, all conditions would match objects in the current example.
  - So to create the rule we select features from those objects and generalize them
  - Can add more complex conditions too, but lots are as simple as a click

```
sp { redux*propose*move*proposalrule-5
   (state <s> ^redux-state true)
   (<s> ^sel <friendly-1>)
   (<friendly-1> ^isa person)
   (<friendly-1> ^isthreat false)
   (<friendly-1> ^isfriend true)
   (<friendly-1> ^goal <goal>)
   (<goal> ^door <door-1>)
   (<goal> ^name breach-the-door)
   (<friendly-1> ^inroom <room-2>)
   (<door-1> ^isa door)
   (<friendly-1> ^cansee <cansee>)
   (<cansee> ^name <door-1>)
   (<cansee> ^distance <arg*1>)
   (<arg*1> ^range0to32 false)
   (<door-1> ^destroyed false)
-->
   (<s> ^operator <o> +,=)
   (<o> ^name move)
   (<o> ^relative <door-1> ^coords …)
}
```

Building rules for Breach-Door



Friendly-1

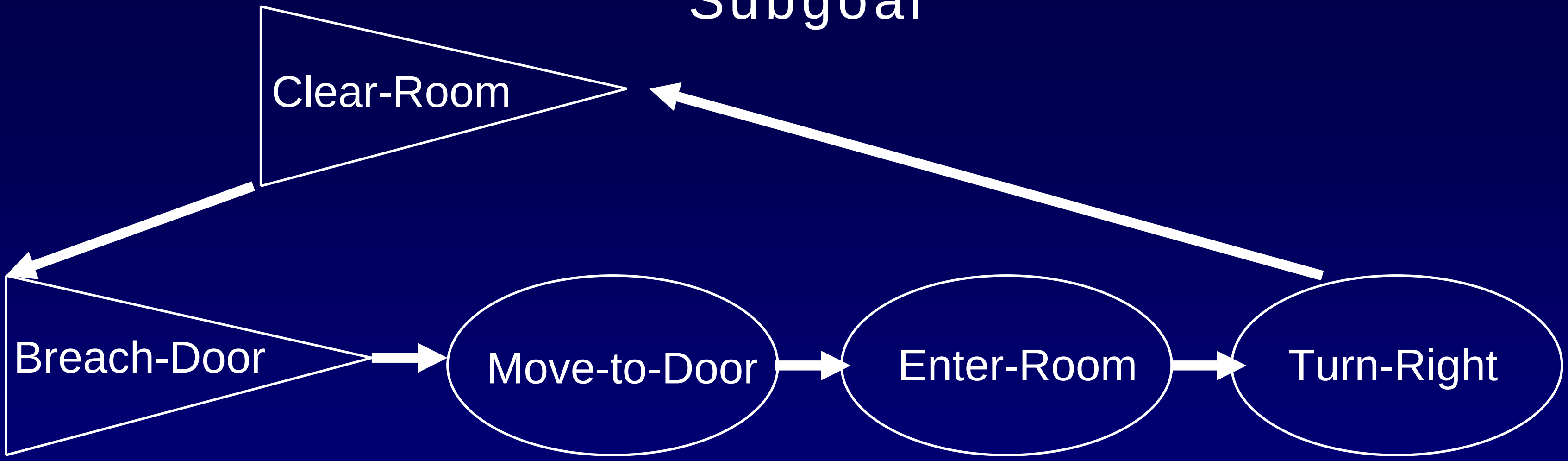
Internal not visible | Detail not visible

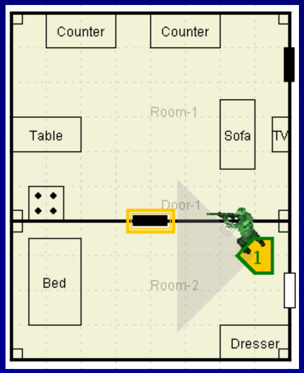
Property	Value
canSee	Room-2 ...
canSee	corner-5 ...
canSee	corner-6 ...
canSee	corner-7 ...
canSee	corner-8 ...
canSee	Door-1 ...
changedroom	false
destroyed	false
inroom	Room-2
isa	Person
isArmed	true
isFriend	true
isThreat	false
nearestThreat	none
role	none
speech	
stance	Standing
team	team1
teamMember	team1_member1
weapon	rifle

```
sp { redux*propose*move*proposalrule-5
   (state <s> ^redux-state true)
   (<s> ^sel <friendly-1>)
   (<friendly-1> ^isa person)
   (<friendly-1> ^isthreat false)
   (<friendly-1> ^isfriend true)
   (<friendly-1> ^goal <goal>)
   (<goal> ^door <door-1>)
   (<goal> ^name breach-the-door)
   (<friendly-1> ^inroom <room-2>)
   (<door-1> ^isa door)
   (<friendly-1> ^cansee <cansee>)
   (<cansee> ^name <door-1>)
   (<cansee> ^distance <arg*1>)
   (<arg*1> ^range0to32 false)
   (<door-1> ^destroyed false)
-->
   (<s> ^operator <o> +,=)
   (<o> ^name move)
   (<o> ^relative <door-1> ^coords …)
}
```

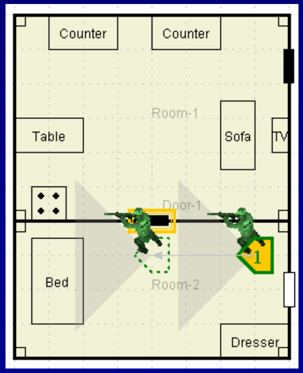
“Human level chunking”

Incremental Approach: Use Breach-Door as a Subgoal

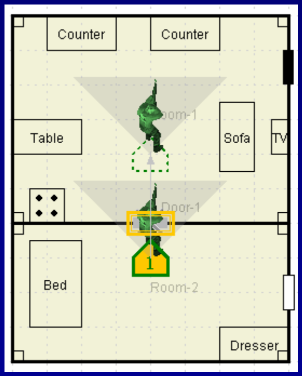




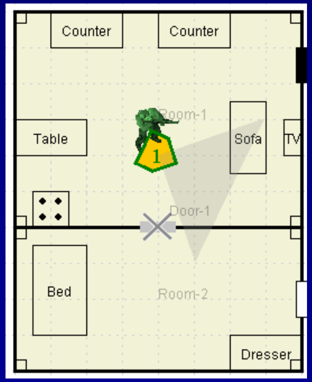
Clear-Room
Door[d, closed]
→
Breach-Door[d]



Breach-Door[d]
Door[d, closed]
→
Move-to-Door[d] &
Shoot[d]

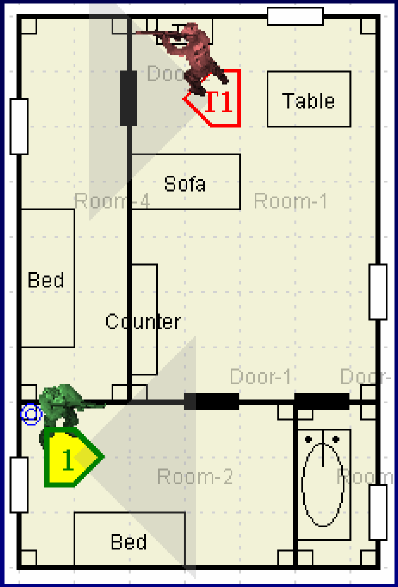


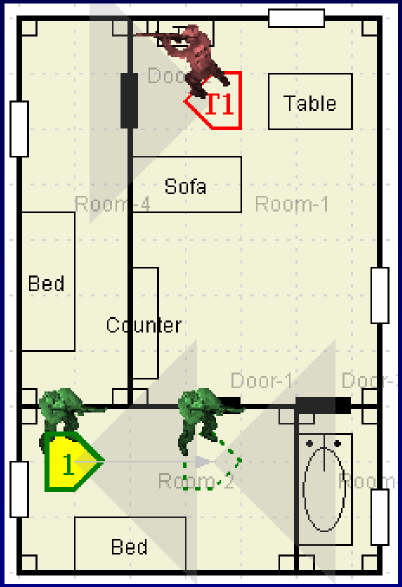
Clear-Room
Door[d, open]
→
Enter-Room

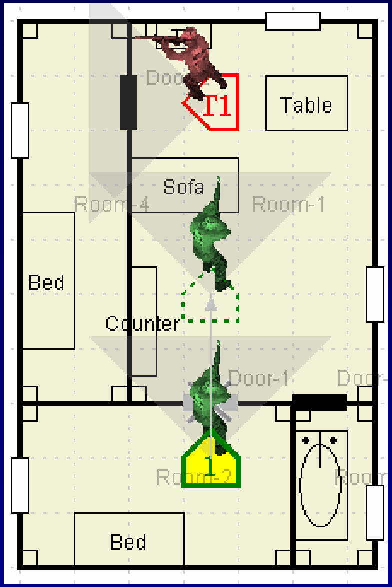


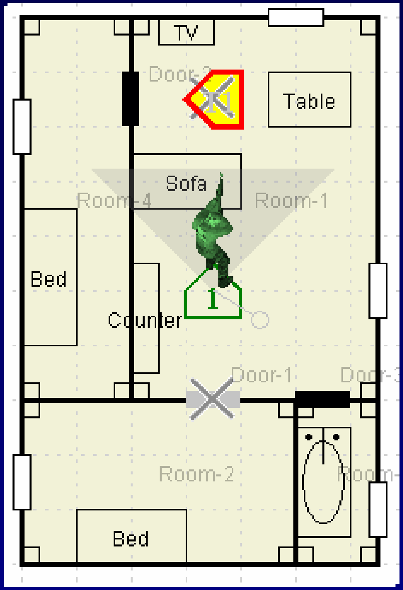
Clear-Room
In-Room
→
Turn-Right

Apply Rules to Implement Subgoal in a New Situation









Clear-Room
Door[d, closed]
→
Breach-Door[d]

Breach-Door[d]
Door[d, closed]
→
Move-to-Door[d] &
Shoot[d]

Clear-Room
Door[d, open]
→
Enter-Room

Clear-Room
Threat[x]
→
Shoot[x]

Clear-Room
In-Room
→
Turn-Right

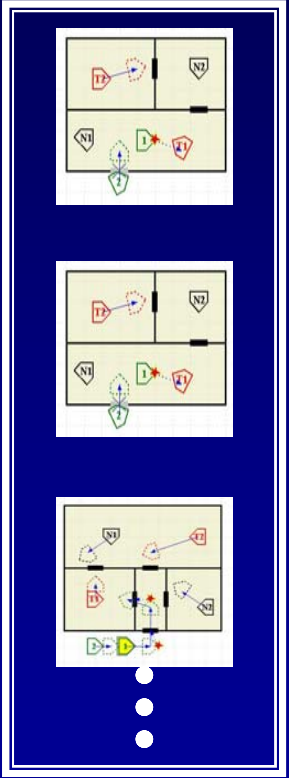
Clear-Room
Threat[x]
→
NOT Turn-Right

Then revise behavior if it's not correct in the new situation

ThreePenny
SOFTWARE

Flexible Ways to Acquire Knowledge

Library of validated behavior examples



Executable Code

A -> B
C -> D
E, J -> F
G, A, C -> H
E, G -> I
J, K -> L

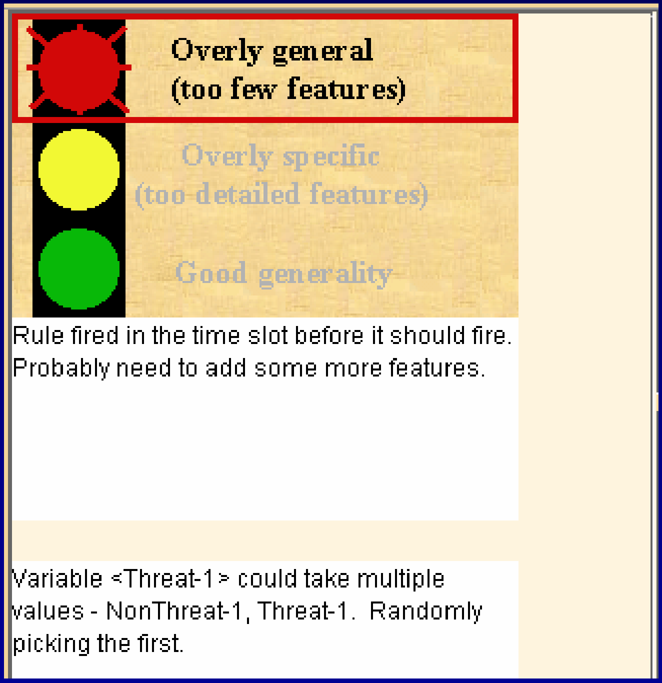
- Compose behavior from building blocks

Simulation Environment

ThreePenny SOFTWARE

Behavior is Created Quickly and Accurately

- Expert's guidance directly available
  - Example walkthroughs
  - Puts the instructor "in the car"
- Differences detected immediately
  - Incorrect action taken
  - Action taken at wrong time
  - Unclear what action to take
- Example guides initial behavior creation
  - Redux can guess at approximate rule
- Result
  - Faster and more accurate rule creation
  - Less skill required by user

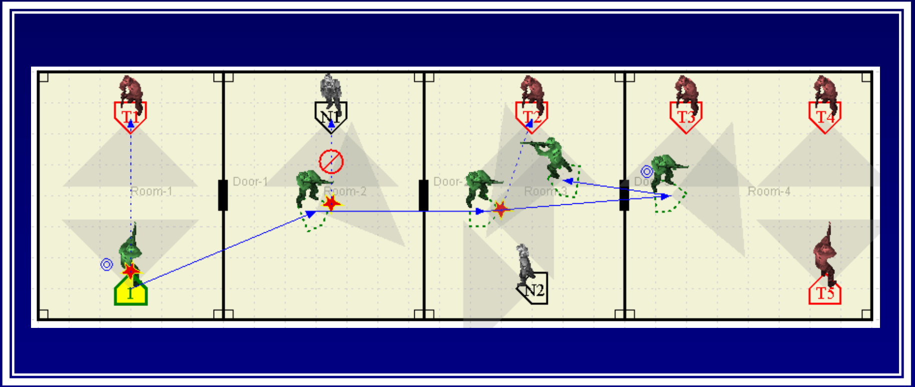


Rule fired in the time slot before it should fire. Probably need to add some more features.

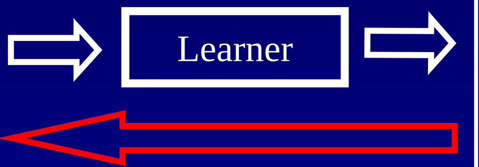
Variable <Threat-1> could take multiple values - NonThreat-1, Threat-1. Randomly picking the first.

Automatically Generate Approximate Behavior Model

Library of validated behavior examples



Generalized Rule



```
Eliminate-Threats
and
Nearest-Threat[x]
→
Shoot[x]
```

- Each scenario provides many examples of “(goal, state) -> action”
- Automatically determines most important features
- Rules can be returned to Redux and refined by the user

- Reduces skill level required to build a behavior model
- Reduces total cost of building model

ThreePenny SOFTWARE

Nuggets and Coal

- Nuggets
  - Lots of power from examples; less from diagrams
  - Even one example is highly constraining
  - Incremental approach now support building blocks
  - Integration with Tolga's learning system
  - Ability to read in external rules and revise them
  - Fast at building rules
    - Standard: ~10 rules/day (equates to ~100 LOC/day)
      - TacAirSoar 8,000 rules => ~2.5 man years
    - Redux: ~10 rules/20 minutes (~25x faster).
  - Simpler
    - Less skill required of developer
    - Easier to capture desired (expert) behavior

- Coal
  - Coverage
    - Can't extend the representation dynamically yet
    - Can we create all rules that we need quickly?

ThreePenny SOFTWARE