Soar Agents in Government Applications

Randolph M. Jones and The Crew
(with special thanks to Glenn Taylor, Brian Stensrud, and Mike Quist)
Soar Technology, Inc.
rjones@soartech.com

Soar Workshop, June 2005

Overview

- Beginning with TacAir-Soar, Soar Technology has developed a family of intelligent agent systems for various government applications
- This talks describes a sampling of these agents, together with lessons learned from developing them
  - Does not include all of our agent systems, particularly some of those covered by other talks at this workshop

Soar Technology
Thinking *inside* the box.

TacAir-Soar

- Yes, it's still around
- Used in SAGIS system for training terminal air controllers
  - Integrated with JSAF
  - Expanding and refining behaviors for close-air support missions
- Soar details
  - Soar 7.0.4
  - About 8000 productions
  - "Michigan approach" to goal representation
  - "Floating operators" and "persistent elaborations"
- Other innovations
  - Message parsing modules (see later talk)
- Other notes
  - Design reuse (and some code reuse) in other agent systems

Soar Technology
Thinking *inside* the box.

Helo-Soar

- Used in Automated Wingman system for Army experimentation
  - Integrated with MÄK Technologies' VR-Forces simulator
  - Providing wingman support roles for helicopter groups in air assault and strike missions
- Used in SAGIS system for training terminal air controllers
  - Integrated with JSAF
  - Expanding and refining behaviors for close-air support missions
- Soar details
  - Soar 8.6 (with Soar Technology modifications)
  - About 700 productions before close-air-support
  - "Radical Randy" approach to goal representation
    - I-supported goal DAG on top state
- Other innovations
  - Some use of TCL code-generation templates
  - Voice interface using ANGST semantic parser
  - Serious application of "Behavior design patterns"
    - Iterator, incoming message handler, etc.

Soar Technology
Thinking *inside* the box.

IF-Soar

- Used in SAGIS system for training terminal air controllers
  - Integrated with JSAF
  - Behaviors for Indirect Fire missions as part of coordinated close-air support missions
- Soar details
  - Soar 8.6 (with Soar Technology modifications)
  - About 1400 productions
  - "New Goal System" approach to goal representation
    - Variation of "Radical Randy"
    - O-supported goal DAG on top state
- Other innovations
  - Significant use of TCL code-generation templates
  - Voice interface using ANGST semantic parser
  - Serious application of "Behavior design patterns"
    - Iterator, incoming message handler, etc.
  - Introduction of Soar 8 into JSAF
  - Extensive use of UML-like design language for agent design

Soar Technology
Thinking *inside* the box.

Component technologies and reuse

- “Radical Randy” representation of goals
- “New goal system” representation
- TCL code-generation templates
- ANGST semantic parser
- Behavior design patterns
- UML-like design language

Soar Technology
Thinking *inside* the box.

Top-state goal representation

- Allows multiple goals to be arranged in a tree, forest, or DAG
- Allows simultaneous activation of multiple goals
- Operators stay selected for only one decision; no operator subgoaling
- High match costs are possible
- Need knowledge for interleaving operators that attend to multiple parallel goals
- Tradeoffs between “Radical Randy” and “New Goal System”
  - I-support
    - Automatic clean-up of old goals (and their subgoals)
    - Takes full advantage of Soar’s reason maintenance system
  - O-support
    - Sometimes you want goals to persist
    - Allows reasoning about past achieved and failed goals
    - Can make debugging easier because goals don’t just disappear

Soar Technology
Thinking *inside* the box.

TCL code-generation templates

- "Macros" for common patterns that appear in productions
- Allow representation changes by changing the code-generation rather than the source code
- Templates can be general or domain-specific
- Allows mixing of templates and "primitive" code

```
sp "explain-agent*create-subgoal*achieve-generate-situation-summary
   [sub-goal-creation <glist> <supergoal>]
   [is-most-derived-type <supergoal> explain-agent]
   (<s> ^situation-kb.vista-situation <vs>)
   (<vs> ^timestamp <time>)
-->
   [create-sub-goal <glist> achieve-generate-situation-summary <supergoal>]
   (<new-goal> ^vista-situation <vs>)
   [create-object <new-goal> document-sections class_Collection <ds> <dstags>]"
```


Soar Technology
Thinking inside the box.

Expanded template

```
sp {explain-agent*create-subgoal*achieve-generate-situation-summary
   (state <s> ^superstate nil
              ^situation-kb.vista-situation <vs>
              ^goals <goals>)
   (<goals> ^active.goal <supergoal>
            ^all <glist>)
   (<vs> ^timestamp <time>)
   (<supergoal> ^type-info.most-derived-type explain-agent)
-->
   (<glist> ^goal <new-goal>)
   (<new-goal> ^tags <new-tags>
               ^type-info <type-info-109>
               ^supergoal <supergoal>
               ^vista-situation <vs>
               ^document-sections <ds>)
   (<type-info-109> ^most-derived-type achieve-generate-situation-summary
                    ^all-types <types-110>)
   (<types-110> ^type achieve-generate-situation-summary
                ^type achievement-goal)
   (<ds> ^tags <dstags>
         ^type-info <type-info-113>)
   (<type-info-113> ^most-derived-type |class_Collection|
                    ^all-types.type |class_Collection| +)
}
```


Soar Technology
Thinking inside the box.

ANGST semantic parser

- Maps multiple message forms to an architecture-neutral ontological form
- Transmits neutral representation through ATE onto agent input-link

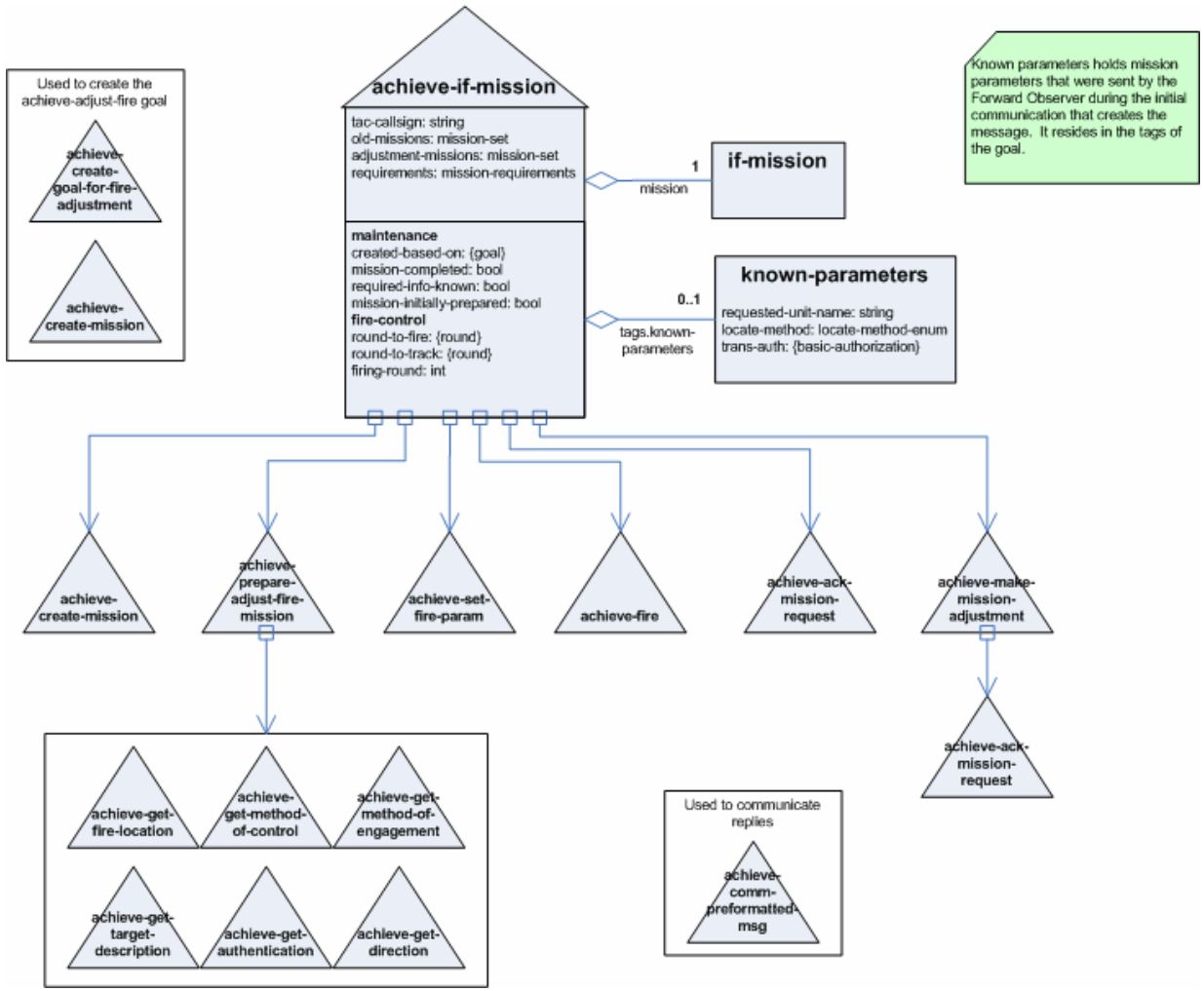
```
<utt> = my name is <name>, I'm <age> and my mom's name is <mom-name>
      | I'm <age>, my name is <name> and my mom's name is <mom-name>
      | My mom's name is <mom-name>, I'm <age> and my name is <name>
```

```
<message>
   <content>
      <name>Brian</name>
      <age>26</age>
      <mom-name>Lynn</mom-name>
   </content>
<message>
```

```
^input-link
  ^message
    ^content
      ^name |Brian|
      ^age 26
      ^mom-name |Lynn|
```


Soar Technology
Thinking *inside* the box.

UML-like design language



Soar Technology
Thinking *inside* the box.

Gold

- We are still building knowledge-intensive agents
- We are getting better at it
- We have developed new technologies for improving and streamlining the design of agents
- We are starting to see significant reuse across knowledge-intensive agents

Soar Technology
Thinking *inside* the box.

Coal

- Building these agents is still hard to do
- Need to refine and improve technologies and reuse
- Lots of room still for improvement

Soar Technology
Thinking *inside* the box.