Soar Technology
Thinking *inside* the box.

Cost-Effective Conversion of Large Soar 7 Systems to Soar 8

Randolph M. Jones
Soar Technology

Top-Level Problem

- Someday we would really like to convert TacAir-Soar to use Soar 8.
- Notable issues:
  - TacAir-Soar is implemented in Soar 7
  - TacAir-Soar uses the “Michigan Approach” to implement (some of) its goals
  - TacAir-Soar includes a number of programming patterns that Soar 8 was specifically designed to prevent from working

Challenges For The Conversion

- In Soar 7, it was easy to create operators that contained arbitrarily long sequences of application rules
  - …and so we did
  - Soar 7 operators are guaranteed not to be deselected until:
    - They are deliberately terminated, AND
    - The Soar decision cycle reaches quiescence
  - Soar 8 operators are deselected as soon as their proposal conditions are no longer matched
    - For many TacAir-Soar operators, this means that the operator would get deselected before the chain of applications gets a chance to finish

Challenges For The Conversion

- Using the Michigan Approach in Soar 7 makes it easy to create operators that stay selected for very long times (hours, even)
  - …and so we did
  - But operators in Soar 8 are much “fussier”, in order to prevent the knowledge in subgoals from becoming inconsistent with the knowledge in supergoals
  - In a Soar 8 implementation of the Michigan Approach, you need proposal conditions to start off a long-lasting operator/goal, but you also need proposal conditions that will re-propose the operator/goal so it can “pick up where it left off” if and when it is interrupted
  - Re-engineering all of the TacAir-Soar operators in this way is prohibitively expensive

An Engineering-Oriented Solution

- If it makes the conversion cheaper, we are willing to use “non-kosher” solutions
  - …and hopefully fix and refactor later
- One of the strategies for implementing the Michigan Approach in Soar 8 is to move subgoal-related information to the top state (so the operators can take up where they left off when interrupted)
  - The alternative “Forest of Goals” approach also maintains all subgoal information on the top state
- Key insight:
  - We can use a couple of tricks to make goals in the “Forest of Goals” approach behave (at least mostly) like Michigan-Approach operators behaved in Soar 7

Basics of the Approach

- Any operator that really only does one quick task can remain an operator
- Any operator that assumes it is going to persist for more than one application must be converted to a "persistent goal"
  - A persistent goal gets installed into the "goal forest" by an operator, using the original operator's proposal conditions
  - A persistent goal gets removed from the "goal forest" by an operator, using the original operator's termination conditions
  - Operator preferences are handled on a case by case basis (but mostly ignored)
- Application productions of converted operators get converted into "goal applications"
  - Goal applications test for the existence of a goal in the "goal forest" instead of for the existence of an operator on the state
  - Goal applications use :o-support to have persistent effects

Examples

- Old code

```
sp {top-ps*propose*init-agent
   (state <s> ^problem-space.name top-ps
             -^initialized *yes*)
-->
   (<s> ^operator <o> + >, =)

   (<o> ^name init-agent ^type output) }
```

- New code

```
sp "top-ps*propose*init-agent
   [match-root-goal <g> <s>]
   (state <s> -^initialized *yes*)
-->
   [create-persistent-subgoal <sg> init-agent <g>]

   (<g> ^type output) "
```

Examples

- Old code

```
sp {init-agent*apply*intialized
   (state <s> ^operator.name init-agent
              ^io.output-link.command flight-command)
-->
   (<s> ^initialized *yes*) }
```

- New code

```
sp "init-agent*apply*intialized
:o-support
   [match-active-goal <g> init-agent <s>]
   (state <s> ^io.output-link.command.value flight-command)
-->
   (<s> ^initialized *yes*) "
```

Soar Technology
Thinking *inside* the box.

Examples

- Old code

```
sp {init-agent*terminate
   (state <s> ^initialized *yes*
              ^operator <o>)
   (<o> ^name init-agent)
-->
   (<s> ^operator <o> @) }
```

- New code

```
sp "init-agent*terminate
   [match-active-goal <g> init-agent <s>]
   (state <s> ^initialized *yes*)
-->
   (<g> ^remove-persistent-goal <g>) "
```

Conclusion

- Nuggets
  - Conversion of “air-route” mission in TacAir-Soar (including all mission-planning and route-flying code)
  - Conversion of Tambe’s STEAM code
  - …”on time and under budget”
  - Makes many rules more explicit about whether they are going to have persistent effects
  - Eliminates some of the problems associated with using “persistent goal stack” (Michigan Approach in Soar 7)
- Lumps
  - Still not clearly cost effective to complete the conversion of TacAir-Soar
  - Some parts of conversion still need to be handled by someone who is intimate with the code
    - Some use of attribute preferences
    - Returning of results from subgoals to supergoals
    - Some preferences between operators