Introduction to Intelligent, Knowledge-Based, and Cognitive Systems
89-674

Mor Vered
http://u.cs.biu.ac.il/~veredm/89-674/
mvered89230@gmail.com
Sunday 10-11, Building 202, Room 119
Learning

• Soar goal is to build a system capable of general intelligent behavior.
• One necessary component of such an architecture is a general learning mechanism.
  • Task Generality – Can improve performance on all tasks in the domain.
  • Knowledge Generality – Can base improvements on any knowledge available about the domain (examples, instructions, hints, its own experience).
  • Aspect Generality – It can improve all aspects of the system.
  • Transfer of Learning – What is learned in one situation will be used in other situations to improve performance.
Search Control

• Performance in soar is based on the problem-state-hypothesis.
• Consists of a set of states representing possible situations
• And set of operators that transform states into other states.
• Problem solving consists of starting at some given initial state and applying operators until a desired state is reached.
• Throughout the process Soar makes selection decisions between available states, operators.
• The knowledge that controls these decisions is called search control
• Problem solving without search control is possible in Soar but leads to an exhaustive search of the problem space.
Elaboration Phase

• During the elaboration phase all directly available knowledge relevant to the current situation is brought to bear.

• Represented as productions in long term memory.
  • 1) Existing objects may have their descriptions elaborated via augmentations
  • 2) Preferences can be created that specify the desirability of an object.
• Quiescence – when no more productions are eligible to fire.

• Decision Procedure – gathers and interprets the preferences provided by the elaboration phase to produce a specific decision.

• If sufficient knowledge exists to make a decision – proceed.

• Otherwise: **impasse**
  • 1) no-change
  • 2) tie
  • 3) conflict
  • 4) rejection

• *Universal subgoaling* – generate **subgoals** in response to impasses

*Figure 1: The Soar decision cycle.*
Resolution

• The goals are structured as a stack, referred to as the context stack.
• A subgoal terminates when its impasse is resolved.
• Soar pops the context stack removing from working memory all augmentations and preferences created in that subgoal that are not connected to a priori context.

• There is some default knowledge to cope with subgoals when no knowledge exists (usually for reject impasses – involves backing up to a prior choice in the context).
Eight Puzzle

Download from: http://soar.eecs.umich.edu/articles/downloads/agents/149-8-puzzle

• The states are configurations of the numbers 1-8 in a 3x3 grid.
• Single move operator to move adjacent tiles into empty cells.
Eight Puzzle Impasses

• Impasses arise each time an operator has to be selected.

• Default tie impasse – exhaustive DFS search

• Whenever another tie impasse arises during the investigation of one alternative an additional subgoal is generated and the search deepens.

• DFS as a means of problem solving does not have to be selected but emerges from the structure of the task and the available search control knowledge.
• Another way to control the search in Eight Puzzle is to break it into a set of subgoals to get the individual tiles into position.

• In this task the subgoals are *non-serializable* – tasks for which there exists no ordering of the subgoals such that successive subgoals can be achieved without undoing what was accomplished by earlier subgoals.

• Must move 2 in order to achieve 3.
8 Puzzle SubGoals

1) Have the blank in its correct position
2) Have the blank and the first tile in their correct positions
3) Have the blank and the first two tiles in their correct positions.
4) So on through the eighth tile...
Chunking

• If impasses solution is remembered can be used for future impasses.
• Chunking is Soar creates rules that summarize the processing of a subgoal.
• When a subgoal begins a learning episode begins that could lead to the creation of a chunk.
• Each chunk is a rule (or set of rules) that gets added to the production memory.
• Chunked knowledge may than be brought to bear in the elaboration phase of later decisions.
Chunk Creation

1) Collection of conditions and actions. Maintains a goal-referenced-list which contains working memory elements that existed before the creation of the goal and are relevant to the satisfaction of the goal.

2) Variabilization of identifiers – label of WME elements, does not mention specific identifiers but only compares value equality.

3) Chunk optimization – Remove irrelevant conditions, eliminate redundant copying, apply a static reordering algorithm to increase matchability.
Evaluation – State Positive

In Eight_Puzzle_Agent\Eight_Puzzle_Agent\eight-puzzle\Impasse__Operator_Tie\evaluate-operator\elaborations.soar

• sp {eight*elaborate*state*evaluation*positive*one
•  (state <s> ^desired.binding <b2>
•    ^tried-tied-operator
•    ^tile-cell <c1>
•    ^binding <b1>)
•  (<b1> ^cell <c1> ^tile <v1>)
•  (<b2> ^cell <c1> ^tile <v1>)
•  -->
•  (<s> ^numeric-value 1))}
Evaluation – State Neutral

• ## 0 points for not moving tile in or out of its desired cell
• #
• sp {eight*elaborate*state*evaluation*neutral*zero
• (state <s> ^desired <d>
• ^tried-tied-operator
• ^tile-cell <c1>
• ^blank-cell <c0>
• ^binding <b1> { <> <b1> <b2> } { <> <b1> <> <b2> <b3> }
• (<d> ^binding <b4> { <> <b4> <b5> })
• (<b1> ^cell <c1> ^tile <v1>)
• (<b2> ^tile <v2>)
• (<b3> ^tile <v3>)
• (<b4> ^cell <c1> ^tile { <> <v1> <v2> })
• (<b5> ^cell <c0> ^tile { <> <v1> <v3> })
• -->
• (</s> ^numeric-value 0)}
Evaluation – state Negative

- ## -1 points for moving tile out of its desired cell
- #
- sp {eight*elaborate*state*evaluation*negative*one
-   (state <s> ^desired.binding <b2>
-     ^tried-tied-operator
-     ^tile-cell <c1>
-     ^binding <b1>
-     ^blank-cell <c0>}
- (b1 ^cell <c1> ^tile <v1>)
- (b2 ^cell <c0> ^tile <v1>)
- -->
- (s ^numeric-value -1)}