**General SOAR terminology**

Production

-An If-Then Pair.

Operator

-*Cause* thing to happen in imagination, or real world.

-Doesn’t actually *do* anything. Just a token in memory that might causes ‘productions’ to fire.

**A Case Study in Integrating Probabilistic Decision Making and Learning in Symbolic Cognitive Architecture**

Soar’s (cognitive architecture) primary memory systems encode knowledge in symbolic representations, and (until now) it has not incorporated probabilities.

Humans use an “expected number” approach when evaluating bids that doesn’t actually involve calculating probabilities.

-i.e., we tend to sum the count of dice with the value in question (that we know about), then we sum in the number of unknown dice divided by a constant. Difference between this *expected number* and the bid as the basis of comparison to other bids.

There are four classes of knowledge that could prove useful to an agent:

1. Probability or expected-number calculations
2. Opponent Models
   1. There is usually a correlation between a player’s bid and the dice under their cup. What dice would my opponent have to have in order that they should make that bid?
3. Expert Heuristics
   1. There are some strategies that don’t really depend on the probability of the rolls, which can be encoded formally/non-probabilistically. Paper gives specific examples for different types of rules that we’re not currently using in our implementation.
4. Knowledge learned by experience
   1. Particularly useful if we want to figure out regularities in other player’s actions.

In SOAR *preferences* are used to select between competing operators (actions):

1. Symbolic Preferences
   1. Create a partial ordering, stating which operator is *better* than which other.
   2. Which operators are equally as good.
   3. What operators are preferred over all others?
2. Numeric Preferences
   1. Specify the expected value of an operator.

Decision procedure uses symbolic preferences to pick out the best subset of operators. Boltzmann distribution is used to select among this subset, assuming they all now have *numeric* preferences.

If an impasse is reaches Soar generates a *subgoal* in which other operators can be selected and applied to resolve the impasse.

-Subgoals allow agents to reason about which (superstate) operator can be selected.

-This subgoal reasoning is where the *four different types of useful knowledge* can be used.

1. Probability or Expected-Number Operations:

Expected-Number (Human-like) strategy:

-Previous Soar systems that reason in subgoals used search. But due to uncertainty, look-ahead search won’t work here.

-Alternative: Select operators that evaluate the success of a given dice-game operator (action).

-“Success” refers to a challenge that is won, or a bid that is not challenged.

-This definition is narrow, and we can improve it.

-Here, agents use classification systems for bids. Based on deviations from the *expected number* it calculated, and the *known value* it has.

-Equivalence classes are created: e.g., which bids are “risky” and which are “safe”.

-Random decision is made from among the actions in top equivalence class.

-Randomness makes the agent’s strategy harder for other player’s to learn.

-Note: Agent will not prefer bluffing (making risky bids) under this model.

Probability Strategy:

-Queries are placed with a probability calculator.

-Computed probabilities are assigned to *numeric preference* for dice-game operators.

-Final selection is made based on Boltzmann Distribution, which make them semi-random.

2. Opponent Model

-Weakness: These Soar agents are quick to challenge unlikely bids. If a player has a high number of some face-value, and bids accordingly, the agent usually calls them and forfeits a die.

-Lets add an abstract operator embodying a model of previous player’s biddng. Attempts to induce the dice under the previous player’s cup, given previous bid.

-Operator is only proposed if there wasn’t just a reroll. If dice were just rerolled, there’s nothing to gain from the model.

-To use model: Agent recreates situation that existed when the opponent made their bid (with the dice under Soar agent’s cup being unknown).

-Agent then incrementally hypothesizes different numbers of dice with the face that was bid, evaluating likelihood of opponent actually having that. Also evaluates if this number of dice would support the bid the opponent made.

-Note: Rules for evaluating hypothetical opponent situations are distinct from dice-game-action task knowledge.

3. Expert Heuristics

-Once probs/expected numbers are computed, more knowledge can be incorporated to select between dice-game-action operators in the top equivalence class.

-Prunes the top dice-game-action operators.

Result

-Expected number agents appear to do better than probability agents.

-However, using probabilistic agents means we can tune performance using reinforcement learning.

Summary: Soar agents use knowledge sources to deliberately reason about alternative dice-game-action operators in a subgoal.

-Result of these calculations are converted to preferences for the dice-game-action operators.

Reinforcement learning in Soar uses Q-learning to adjust numeric preferences of rules (created by chunking) when there is a reward signal.

Discussion:

-Agents initially use deliberate reasoning to compute probability/expected numbers.

-These computations can be combined with heuristic knowledge and opponent models.

-Through chunking, agents combine these types of knowledge and create new rules.

-Reinforcement learning used to tune chunked rules achieve improvements in performance.