What’s offered:
- A Soar-like execution cycle

Meaning:
1. $\text{^io.input-link}$
2. elaboration cycle
3. numeric preferences (and implicit operator proposal)
4. decide
   - impasses
5. act
What’s present:
- Soar-RL-like reinforcement learning support
- Architectural support for efficiently creating more specific RL-rules over time – a generative model for a value function

What’s missing or different:
- Manipulating WMEs from the RHS has not been tested
- Operators (as you know them) and impasses do not exist
- Chunking, SMem, EpMem, and SVS do not exist
A Sneak Peak:

- All refinement criteria have corresponding unrefinement criteria.
- Results demonstrating that rerefinement:
  - Can save CPU time
  - Can reduce regret
  - Can improve the terminal policy
Reinforcement Learning

- **Goals:**
  - Approximate $\pi(s, a)$, the optimal target policy
  - Minimize regret to the extent possible

- **How:**
  - Build up the set of features, $\phi(i)$
    - Rules are specialized/despecialized
  - Update the weights, $\theta(i)$ using $\text{GQ}(\lambda)$ during $\epsilon$-greedy exploration
    - The value function is refined/unrefined
  - Use incremental algorithms for efficient, stable computation times
Value Function Refinement Over Time

$w_0, f_{1.1}, f_{1.2}, f_{2.1}, f_{2.2}, f_{3.1}, f_{3.2}$

Before any refinement
Value Function Refinement Over Time

After one refinement
Value Function Refinement Over Time

After two refinements

\begin{align*}
& w_0 + w_{1.1} + w_{3.1}, \quad f_{2.1}, f_{2.2} \\
& w_0 + w_{1.2}, \quad f_{2.1}, f_{2.2}, \quad f_{3.1}, f_{3.2} \\
& w_0 + w_{1.1} + w_{3.2}, \quad f_{2.1}, f_{2.2}
\end{align*}
Relational Blocks World

Full representation of the goal presented by the environment
Allows variable goals and numbers of blocks for different episodes
Significantly more complex training goal
  - Must test more than one relation
A Carli Agent Rule

\[
\begin{align*}
\text{sp} \ \{ \text{blocks-world*rl-fringe*u16} \\
\quad : \text{feature 3 unsplit blocks-world*rl-fringe*s3} \\
\quad (\langle s \rangle \ ^{\text{blocks}} \langle \text{blocks} \rangle) \\
\quad (\langle s \rangle \ ^{\text{goal}} \langle \text{goal} \rangle)
\end{align*}
\]

: # Rule abbreviated

\[
\begin{align*}
\quad -\{ (\langle \text{goal} \rangle \ ^{\text{stack}} \langle \text{goal-stack} \rangle) \\
\quad (\langle \text{stack} \rangle \ ^{\text{matches}} \langle \text{goal-stack} \rangle) \}
\end{align*}
\]

\[
\text{--->}
\]

\[
\begin{align*}
\quad = 0.3290046905701842217
\end{align*}
\]
A Successor Carli Agent Rule

\[
\text{sp } \{ \text{blocks-world*rl-fringe*u38} \\
\quad : \text{feature 3 unsplit blocks-world*rl-fringe*s16} \\
\quad \langle s \rangle \ ^\text{blocks} \langle \text{blocks} \rangle \\
\quad \langle s \rangle \ ^\text{goal} \langle \text{goal} \rangle \\
\quad \} \\
\quad \quad \# \text{ Rule abbreviated} \\
\quad -\{ \langle \text{goal} \rangle \ ^\text{stack} \langle \text{goal-stack} \rangle \} \\
\quad \quad \quad \langle \text{stack} \rangle \ ^\text{matches} \langle \text{goal-stack} \rangle \} \\
\quad +\{ \langle \text{goal} \rangle \ ^\text{stack} \langle \text{goal-stack} \rangle \} \\
\quad \quad \quad \langle \text{dest-stack} \rangle \ ^\text{matches} \langle \text{goal-stack} \rangle \} \\
\quad \} \\
\quad \quad \quad \quad \quad \rightarrow \\
\quad \quad \quad \quad \quad \quad \quad = 0.0 \\
\}
Each state is described by a set of relations, such as \((<\text{stack}> \ ^\text{top} <\text{block}>)\).

Each RL-rule / feature in \(\phi(i)\) represents a conjunction of any number of such relations.

Given \(\phi(i)\), \(\theta(i)\), and other metadata, which features are most likely to improve the value function?

We’ve explored the following criteria:
- Cumulative Absolute Temporal Difference Error
- Policy – Maximal change in \(\pi(s, a)\)
- Value – Maximal change in \(\theta(i)\)

We’ve done so with:
- No unrefinement
- Unlimited rerefinement
- Rerefinement with blacklists
- Rerefinement with boosts for previous choices
Cumulative Absolute Temporal Difference Error

- Focus on regions of high activity and error.
- Track TD error experienced at each leaf node in the value function.
- The nodes with highest error are eligible for specialization when their features match.
- Despecialization:
  - Test whether error experienced at the internal “fringe” node is greater than that accumulated by the child nodes.
Focus on modifying policy (Whiteson 2007)
Choose features which maximize the change in the greedy set of actions.
Despecialization:
- Simulate criterion as through no further refinement had been done.
- Evaluate whether a feature that was not chosen results in a larger change in the greedy set of actions.
Focus on improving value estimates (Whiteson 2007)
Choose features which maximize value spread on specialization.
Despecialization:
- Simulate criterion as through no further refinement had been done.
- Evaluate whether a feature that was not chosen creates a wider spread than the actual value function.
Cumulative Absolute Temporal Difference Error

- Metric focuses on regions of high activity and error
- No unrefinement and boost are best

Reward Within an Episode vs. Step Number

Unrefinements:
- catde-catde-none-5k
- catde-catde-bkls-5k
- catde-catde-bst-5k
- catde-none-5k

Mitchell Keith Bloch (University of Michigan)
Cumulative Absolute Temporal Difference Error

- Metric focuses on regions of high activity and error
- No unrefinement and boost are best

![Cumulative Absolute Temporal Difference Error Graph]

**Automatic $\phi(s, a)$ (Un)Refinement for RRL**
Policy Criterion

- Focus on policy difference (Whiteson 2007)
- Unlimited rerefinement is best
Focus on policy difference (Whiteson 2007)

Unlimited rerefinement is best
Focus on improving value estimates (Whiteson 2007)
Unlimited rerefinement and boost are best
Focus on improving value estimates (Whiteson 2007)
Unlimited rerefinement and boost are best
Unrefinement helps with two of the three criteria.

Boost helps both CATDE and Value.

Boost’s hurting Policy casts doubt on it as a criterion. Perhaps the dependency between the number of actions and environmental relations makes it unsuitable for relational reinforcement learning.

Boost is expensive without some new new optimization.

<table>
<thead>
<tr>
<th>CPU Time</th>
<th>No Unrefinement</th>
<th>Unrestricted</th>
<th>Blacklisting</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATDE</td>
<td>40.9</td>
<td>148.3</td>
<td>146.3</td>
<td>167.2</td>
</tr>
<tr>
<td>Policy</td>
<td>25.0</td>
<td>15.2</td>
<td>21.8</td>
<td>61.9</td>
</tr>
<tr>
<td>Value</td>
<td>36.2</td>
<td>20.8</td>
<td>21.6</td>
<td>115.0</td>
</tr>
</tbody>
</table>
Nuggets and Coal

Nuggets:

- Rerefinement is implemented!
- It can save CPU time and improve performance.
- I can finish my thesis now.

Coal:

- Internal “fringe” nodes necessary for policy and value unrefinement criteria hurt performance.
- Boost hurts performance even more without providing additional guarantees about the terminal policy.
- The policy criterion may unsuitable for relational reinforcement learning.