We test relational reinforcement learning agents using all three of our specialization criteria (see right) as we add respecialization functionality. We add with Boost+Concrete. It was Bloch and Laird’s [2015] hypothesis that it would be more efficient to specialize the value function quickly, making potentially suboptimal specializations as a result, and to later modify that value function in the event that the agent finds it could have made significantly better specializations.

**Reinforcement Learning**

- Must learn how to act, given experience perceiving states, trying actions, and receiving rewards
- Explore with an -greedy exploration strategy
- Our agents learn using GQA
- At the most abstract:
  - \( \pi(s,a) \) represents the target policy
  - \( \phi(i) \) represents the set of possible features
  - \( \theta(i) \) stores weights which sum to provide value estimates for different actions

**Relational RL**

- Each state is described by a set of relations, such as \(<\text{stack> top</block}>\)
- Each feature in \( \phi(i) \) represents a conjunction of any number of such relations
- Value function computation could dominate CPU time since variable bindings are expensive
- We use the Rete algorithm
  - It was designed for expert system rules
  - Handles variable bindings very efficiently
  - CPU time proportional to changes in environment rather than the total size of the environment
  - Shares work between similar rules

**Our Relational Blocks World**

- Full representation of the goal presented by the environment
- Allows variable goals from episode to episode
- Allows variable numbers of blocks from episode to episode
- Complex training goal requires testing more than one relation

**Agents Version #1: No Despecialization**

- We start with agents which are capable of dynamically specializing the value function
- However, they cannot choose to undo seemingly suboptimal specializations in favor of potentially better ones

**Agents Version #2: Respecialization**

- We add the ability to respecialize
- However, we provide no mechanism to guarantee convergence on feature selection
- Thrashing results for two of our specialization criteria

**Agents Version #3: Respecialization + Boost**

- We provide a mechanism to incrementally boost the likelihood of reselecting previously selected features
- Features chosen early are expected to be quite good, so eventually converging on one of them is likely to allow efficient learning
- While this guarantees eventual value function convergence, it does so at a significant computational cost

**Agents Version #4: Respecialization + Boost + Concrete**

- Freezing feature selection choices after a certain amount of evaluations without respecialization allows us to reduce computational load
- It additionally can result in a reduction in regret, even when compared to the agents using boost without the concrete mechanism

**Final Cumulative Reward | Average Runtimes of Our Agents**

<table>
<thead>
<tr>
<th></th>
<th>No Respecialization</th>
<th>Respecialization</th>
<th>With Boost</th>
<th>With Boost+Concrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATDE</td>
<td>-12.5</td>
<td>-15.8</td>
<td>-5.3s</td>
<td>9.9s</td>
</tr>
<tr>
<td>Policy</td>
<td>-20.2</td>
<td>15.0s</td>
<td>-45.1</td>
<td>1.2s</td>
</tr>
<tr>
<td>Value</td>
<td>-5.1</td>
<td>19.2s</td>
<td>-7.4</td>
<td>1.6s</td>
</tr>
</tbody>
</table>

**Dynamic Specialization**

- 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 \( \theta(i) \)

**Value Function Specialization**

- Less refined tilings correspond to conjunctions of few features
- More refined tilings correspond to conjunctions of many features
- The most refined tilings correspond to fringe nodes i.e. candidate conjunctions for inclusion in the value function

**Contributions**

- This work implemented much of the future work suggested by Bloch and Laird [2015].
- We provide evidence to support Bloch and Laird’s [2015] hypothesis that it is more efficient to specialize the value function quickly, making potentially suboptimal specializations as a result, and to later modify that value function in the event that the agent finds it could have made significantly better specializations

**Future Work**

- A higher order grammar for adding variables and new relations using these variables – for scenarios where an unbounded number of objects may be of interest
- More domains (suggestions welcome!)