Learning Reward Machines for Partially Observable Reinforcement Learning

Rodrigo Toro Icarte  Ethan Waldie  Toryn Q. Klassen  Richard Valenzano  Margarita P. Castro  Sheila A. McIlraith

UNIVERSITY OF TORONTO  VECTOR INSTITUTE

ELEMENT AI

KR 2020  September 16
Hi, I’m Rodrigo :)

The ultimate goal of AI is to create computer programs that can solve problems in the world as well as humans.

— John McCarthy

Our research incorporates insights from knowledge, reasoning, and learning, in service of building general-purpose agents.
Hi, I’m an AI researcher
Hi, I’m an AI researcher

“The ultimate goal of AI is to create computer programs that can solve problems in the world as well as humans.”

— John McCarthy
Hi, I’m an AI researcher

“The ultimate goal of AI is to create computer programs that can solve problems in the world as well as humans.”
— John McCarthy

Our research incorporates insights from knowledge, reasoning, and learning, in service of building general-purpose agents.
Reinforcement Learning (RL)

<table>
<thead>
<tr>
<th>RL Agent</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>Transition Probabilities</td>
</tr>
<tr>
<td></td>
<td>Reward Function</td>
</tr>
</tbody>
</table>
Reinforcement Learning (RL)

- RL Agent
  - Policy
- Action
- Environment
  - Transition Probabilities
  - Reward Function

This learning process captures some aspects of human intelligence.
Reinforcement Learning (RL)

This learning process captures some aspects of human intelligence.
Reinforcement Learning (RL)

This learning process captures some aspects of human intelligence.
Reinforcement Learning (RL)
How to enhance RL with KR
Long-standing RL problems that we tackled using KR:

- Reward specification.
- Sample efficiency.
- Memory.
- ...
Reward specification
Reward specification

Make a bridge: get wood, iron, and use the factory
**Reward specification**

- **Make a bridge**: get wood, iron, and use the factory

- **LTL specifications**\(^1\):
  \[ \Diamond (\text{got\_wood} \land \Diamond \text{used\_factory}) \land \Diamond (\text{got\_iron} \land \Diamond \text{used\_factory}) \]

---

\(^1\) Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18).
Reward specification

Make a bridge: get wood, iron, and use the factory

LTL specifications\(^1\):
\[ \Diamond (\text{got}_\text{wood} \land \Diamond \text{used}_\text{factory}) \land \Diamond (\text{got}_\text{iron} \land \Diamond \text{used}_\text{factory}) \]

Reward machines\(^2\):
Automata-based reward functions

---

\(^1\) Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18).

\(^2\) Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18).
Reward machine

Make a bridge: get wood, iron, and use the factory
Reward specification

**Make a bridge**: get wood, iron, and use the factory

**LTL specifications**\(^1\):
\[ \Diamond (\text{got\_wood} \land \Diamond \text{used\_factory}) \land \Diamond (\text{got\_iron} \land \Diamond \text{used\_factory}) \]

**Reward machines**\(^2\):
Automata-based reward functions

**Formal languages**\(^3\):
Many formal languages $\rightarrow$ Reward machines.

---

\(^1\) Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18).
\(^2\) Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18).
\(^3\) LTL and Beyond: Formal Languages for Reward Function Specification in RL (IJCAI-19).
Sample efficiency
Sample efficiency

Craft World

<table>
<thead>
<tr>
<th>Training steps (in thousands)</th>
<th>Avg. reward per step</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>1,000</td>
<td>0.2</td>
</tr>
<tr>
<td>1,500</td>
<td>0.4</td>
</tr>
<tr>
<td>2,000</td>
<td>0.6</td>
</tr>
<tr>
<td>2,500</td>
<td>0.8</td>
</tr>
<tr>
<td>3,000</td>
<td>1</td>
</tr>
</tbody>
</table>
Reward machine

How to exploit the reward machine’s structure:

- **CRM**: Counterfactual reasoning.
- **HRM**: Task decomposition.
- **RS**: Reward shaping.
### Sample efficiency

<table>
<thead>
<tr>
<th>Training steps (in thousands)</th>
<th>Avg. reward per step</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>1,000</td>
<td>0.2</td>
</tr>
<tr>
<td>1,500</td>
<td>0.4</td>
</tr>
<tr>
<td>2,000</td>
<td>0.6</td>
</tr>
<tr>
<td>2,500</td>
<td>0.8</td>
</tr>
<tr>
<td>3,000</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Legend:  
- **QL**  
- **QL+RS**  
- **HRM**  
- **CRM**  
- **CRM+RS**
Sample efficiency

Half-Cheetah

Legend: DDPG DDPG+RS HRM CRM CRM+RS
Memory
Memory

![Diagram showing Memory Agent and Button connection]
Memory

Button

Diagram showing the relationship between Memory, Agent, and Button.
Memory

(Cookie)
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory
Memory

(+1 Reward)
Memory
Memory
Memory
Memory
The most popular approach:

Training LSTMs policies using a policy gradient method.

... starves in the cookie domain.

Legend:
- Optimal
- ACER
- A3C
- PPO
- DDQN
If the agent can detect the color of the rooms (⬛, ⬛, ⬛, ⬛), and when it presses the button (🔴), eats a cookie (🍪), and sees a cookie (🍪), then:

\[
\langle o/w, 0 \rangle \langle \text{⬛, 0} \rangle; \langle o/w, 0 \rangle \langle \text{⬜, 0} \rangle; \langle o/w, 0 \rangle \\
\langle \text{🟦, 0} \rangle \langle o/w, 0 \rangle \langle \text{⬜, 0} \rangle; \langle \text{⬛, 0} \rangle \langle o/w, 0 \rangle \\
\langle \text{⬜, 0} \rangle \langle o/w, 0 \rangle \\
\langle \text{⬛, 1} \rangle \langle o/w, 0 \rangle \\
\langle \text{⬜, 1} \rangle \langle o/w, 0 \rangle \\
\langle \text{⬛, 1} \rangle
\]

... becomes a “perfect” memory for the cookie domain.
Reward Machines as memory

If the agent can detect the color of the rooms (🟣, □, □, □), and when it presses the button (⭕), eats a cookie (🍪), and sees a cookie (🍪), then:

... becomes a **“perfect” memory** for the cookie domain.

Learning Reward Machines for Partially Observable Reinforcement Learning (NeurIPS-19).
*Note*: The detectors were also given to the baselines.
Summary
If you are interested in KR ∩ RL, consider reading our papers:

Advice-Based Exploration in Model-Based Reinforcement Learning (Canadian AI-18)
Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18)
Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18)
LTL and Beyond: Formal Languages for Reward Function Specification in RL (IJCAI-19)
Learning Reward Machines for Partially Observable RL (NeurIPS-19)
Symbolic Plans as High-Level Instructions for Reinforcement Learning (ICAPS-20)

Code:
https://bitbucket.org/RToroIcarte/

Thanks! :}
If you are interested in KR ∩ RL, consider reading our papers:

Advice-Based Exploration in Model-Based Reinforcement Learning (Canadian AI-18)
Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18)
Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18)
LTL and Beyond: Formal Languages for Reward Function Specification in RL (IJCAI-19)
Learning Reward Machines for Partially Observable RL (NeurIPS-19)
Symbolic Plans as High-Level Instructions for Reinforcement Learning (ICAPS-20)
If you are interested in KR ∩ RL, consider reading our papers:

Advice-Based Exploration in Model-Based Reinforcement Learning (Canadian AI-18)
Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18)
Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18)
LTL and Beyond: Formal Languages for Reward Function Specification in RL (IJCAI-19)
Learning Reward Machines for Partially Observable RL (NeurIPS-19)
Symbolic Plans as High-Level Instructions for Reinforcement Learning (ICAPS-20)

Code: https://bitbucket.org/RToroIcarte/
If you are interested in KR ∩ RL, consider reading our papers:

Advice-Based Exploration in Model-Based Reinforcement Learning (Canadian AI-18)
Teaching Multiple Tasks to an RL Agent using LTL (AAMAS-18)
Using Reward Machines for High-Level Task Specification and Decomposition in RL (ICML-18)
LTL and Beyond: Formal Languages for Reward Function Specification in RL (IJCAI-19)
Learning Reward Machines for Partially Observable RL (NeurIPS-19)
Symbolic Plans as High-Level Instructions for Reinforcement Learning (ICAPS-20)

**Code:** https://bitbucket.org/RToroIcarte/

Thanks! :)}