Advising and Instructing Reinforcement Learning Agents with LTL and Automata

Toryn Q. Klassen
toryn@cs.toronto.edu

Department of Computer Science
University of Toronto

October 29, 2018

Credits

Rodrigo Toro Icarte

Richard Valenzano

Sheila A. McIlraith

Credits

Rodrigo Toro Icarte

Richard Valenzano

Sheila A. McIlraith

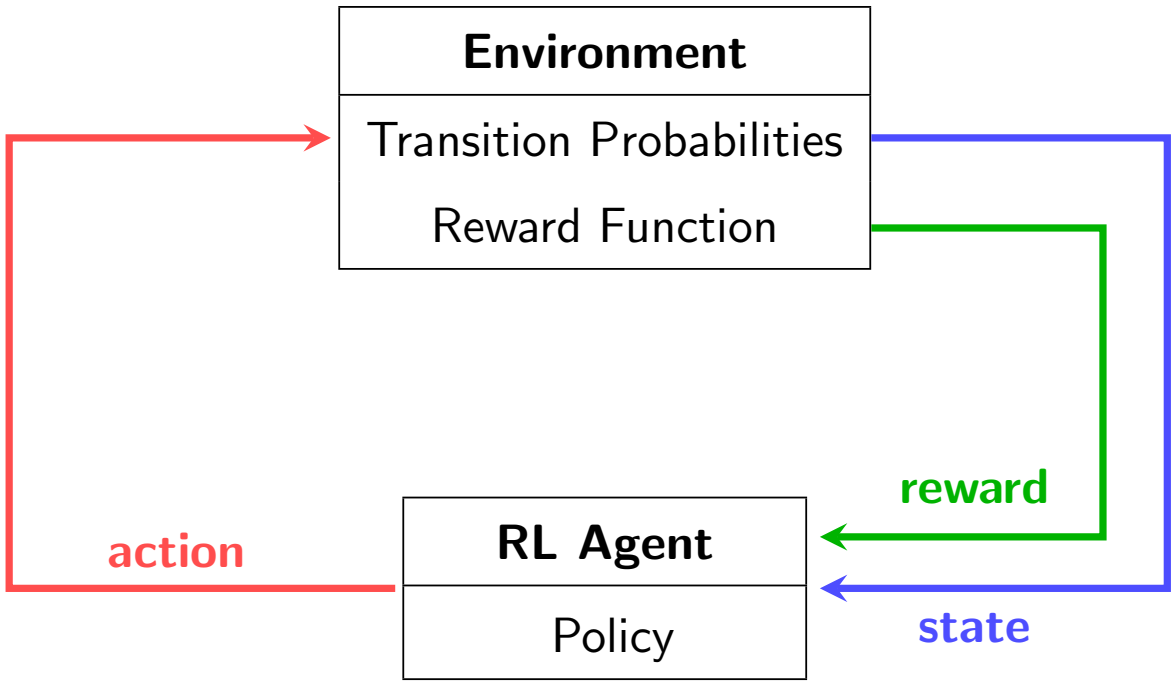
Papers appearing at

- Canadian AI (Toro Icarte et al., 2018b),
- AAMAS 2018 (Toro Icarte et al., 2018a),
- and ICML 2018 (Toro Icarte et al., 2018c)

Outline

1. Reinforcement Learning (RL):
    - What is RL?
    - Two difficulties in applying RL
2. Instructions for Reinforcement Learning
    - LTL formulas
    - Reward Machines
3. Advice for Reinforcement Learning
4. Summary

How does Reinforcement Learning work?



Based on diagram from Sutton and Barto (1998, Figure 3.1)

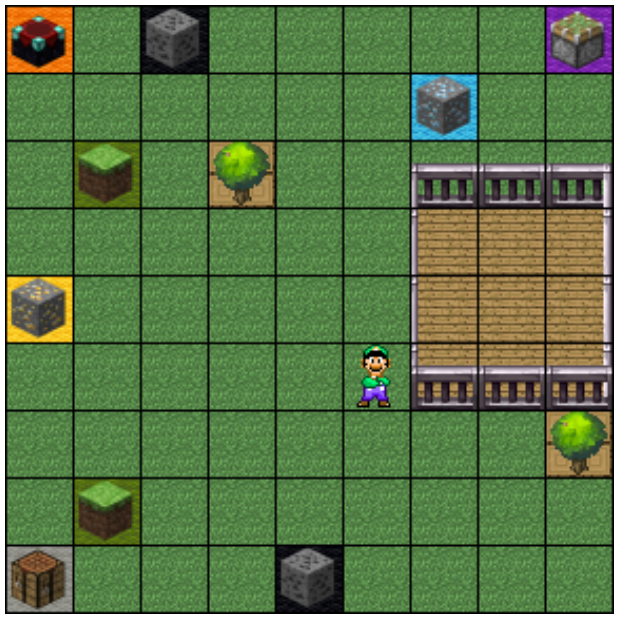
Two difficulties in applying RL

- **Reward specification**: It is really hard to define proper reward functions for complex tasks.
- **Sample efficiency**: RL agents might require billions of interactions with the environment to learn good policies.

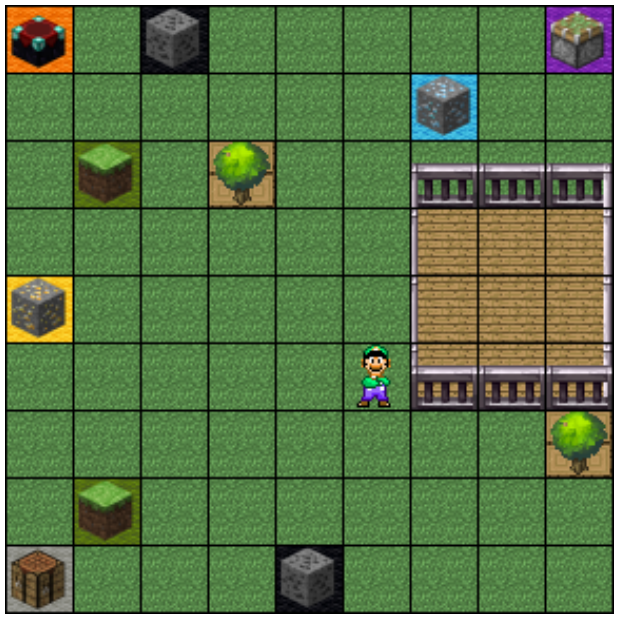
Outline

1. Reinforcement Learning (RL):
    - What is RL?
    - Two difficulties in applying RL
2. Instructions for Reinforcement Learning
    - LTL formulas
    - Reward Machines
3. Advice for Reinforcement Learning
4. Summary

Example environment



Example environment



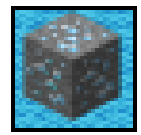
Luigi can collect raw materials:


wood

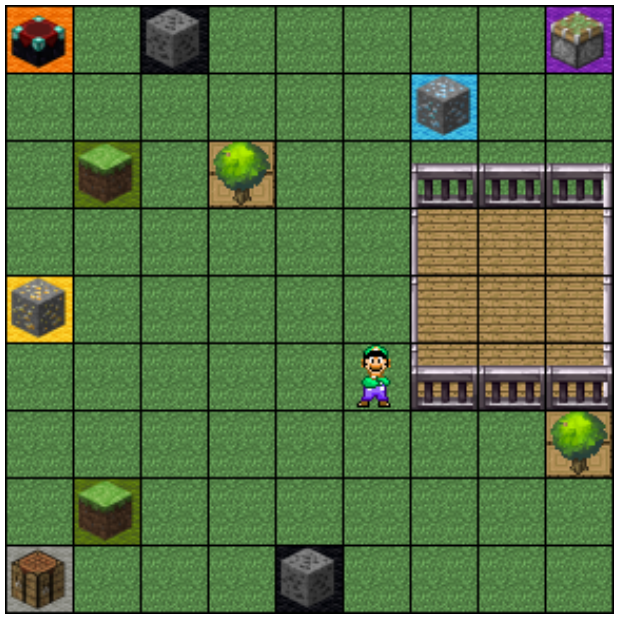

grass


iron


gold


gems

Example environment



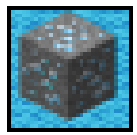
Luigi can collect raw materials:


wood


grass

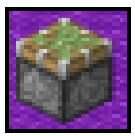

iron


gold


gems

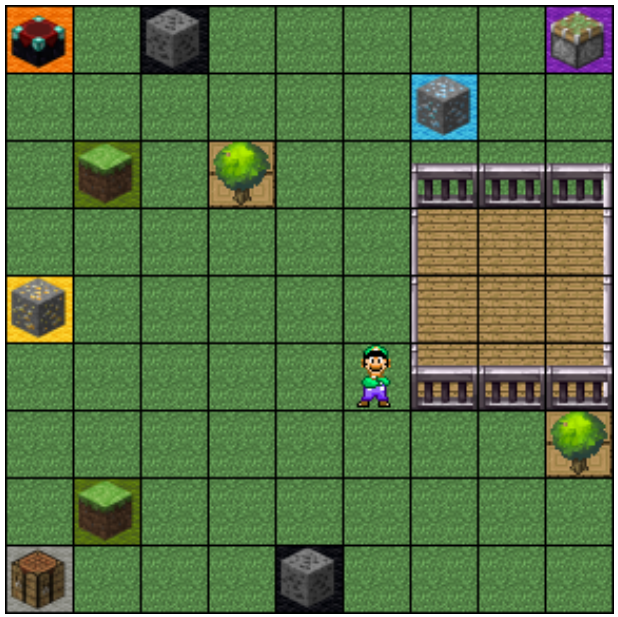
... and make new objects in:


factory


toolshed


workbench

Example environment



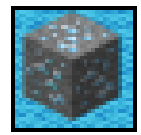
Luigi can collect raw materials:


wood


grass


iron


gold


gems

... and make new objects in:


factory


toolshed


workbench

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

Linear Temporal Logic (LTL) (Pnueli, 1977)

LTL augments propositional logic with the **temporal** operators $\bigcirc$ (*next*), $\Diamond$ (*eventually*), and U (*until*):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \Diamond\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

where p is an atomic symbol.

Linear Temporal Logic (LTL) (Pnueli, 1977)

LTL augments propositional logic with the **temporal** operators $\bigcirc$ (*next*), $\Diamond$ (*eventually*), and U (*until*):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \Diamond\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

where p is an atomic symbol.

Examples:

$$\Diamond \texttt{got\_wood} \tag{1}$$

$$\Diamond(\texttt{got\_grass} \wedge \Diamond \texttt{used\_factory}) \tag{2}$$

$$\Diamond \texttt{got\_wood} \vee \Diamond \texttt{got\_iron} \tag{3}$$

$$\Diamond \texttt{got\_grass} \wedge \Diamond \texttt{got\_iron} \tag{4}$$

$$(\texttt{is\_night} \rightarrow \texttt{at\_shelter})\,\mathsf{U}\,\texttt{got\_wood} \tag{5}$$

Instructing RL agents with co-safe LTL

General idea:

- Reward the agent when it satisfies the formula.
- Therefore, an optimal policy would satisfy the formula **as soon as possible**.

Instructing RL agents with co-safe LTL

General idea:

- Reward the agent when it satisfies the formula.
- Therefore, an optimal policy would satisfy the formula **as soon as possible**.

Main advantage:

- **Standard RL**: The reward function is a black box.
- **RL with LTL**: The LTL formula exposes the task's structure to the agent.

Example

Consider telling the agent to learn a policy for the following task:

$$\varphi = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$$

Example

Consider telling the agent to learn a policy for the following task:

$$\varphi = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$$

Then, the agent knows that at some point it might have to satisfy some of the following formulas:

$$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$$
$$\varphi_2 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$$
$$\varphi_3 = \Diamond\texttt{used\_factory}$$
$$\varphi_4 = \Diamond\texttt{got\_gold}$$

Example

Consider telling the agent to learn a policy for the following task:

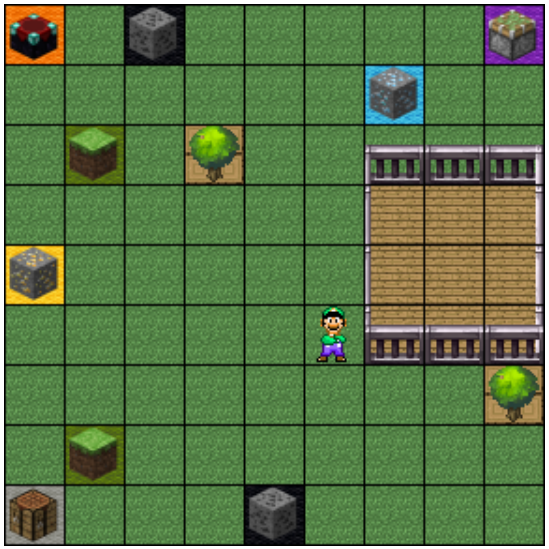
$$\varphi = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$$

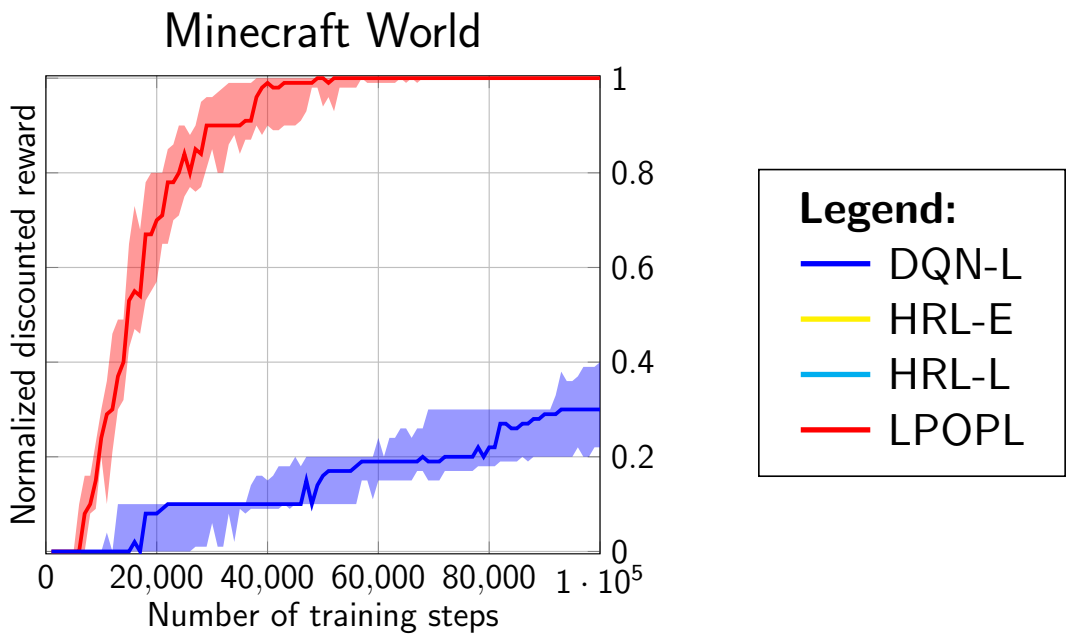
Then, the agent knows that at some point it might have to satisfy some of the following formulas:

$$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$$
$$\varphi_2 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$$
$$\varphi_3 = \Diamond\texttt{used\_factory}$$
$$\varphi_4 = \Diamond\texttt{got\_gold}$$

We proposed to combine this knowledge with off-policy (deep) RL to learn optimal policies for the task and each subtask in parallel.

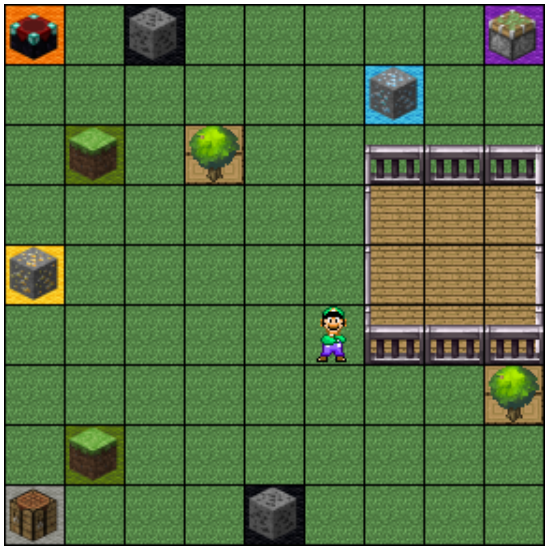
Results

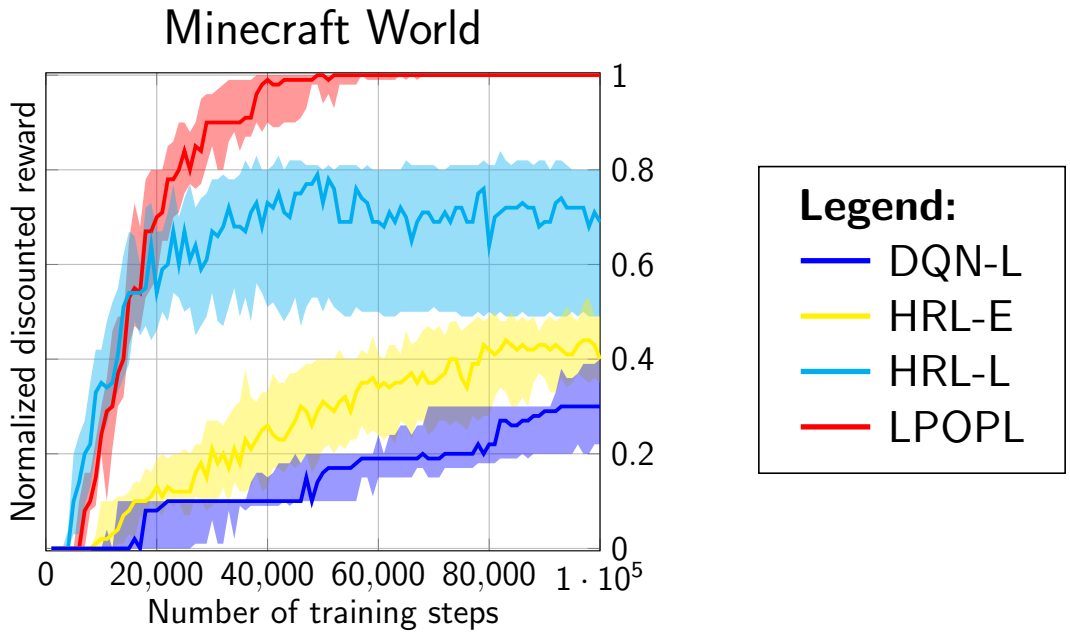




Our approach (red curve) finds better policies faster than standard DRL (blue curve)

Results





Our approach (red curve) finds better policies faster than standard DRL (blue curve) and Hierarchical DRL (yellow and cyan curves).

Paper: "Teaching Multiple Tasks to an RL Agent using LTL"
Code: `https://bitbucket.org/RToroIcarte/lpopl`

Instructing RL agents with automata

Our ICML paper generalizes the previous idea to work over automata representations of the reward function.

Instructing RL agents with automata

Our ICML paper generalizes the previous idea to work over automata representations of the reward function.

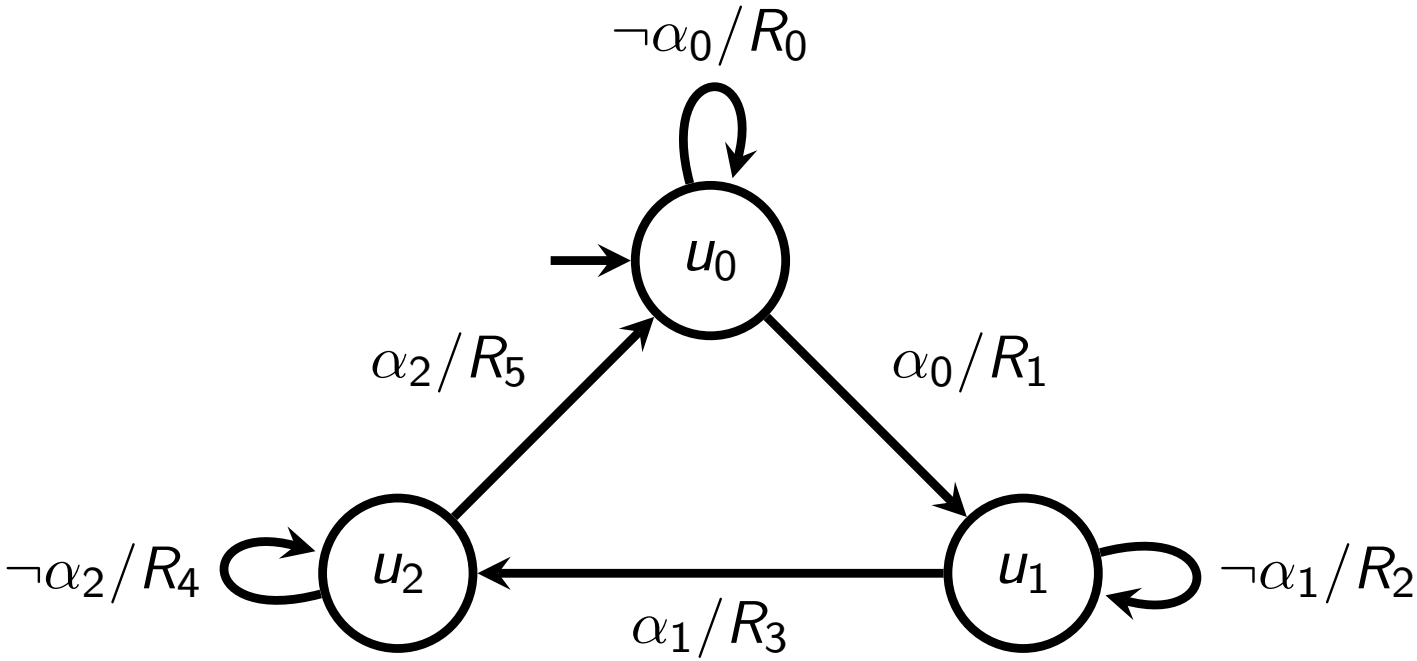


Figure: A **reward machine**

Instructing RL agents with automata

Our ICML paper generalizes the previous idea to work over automata representations of the reward function.

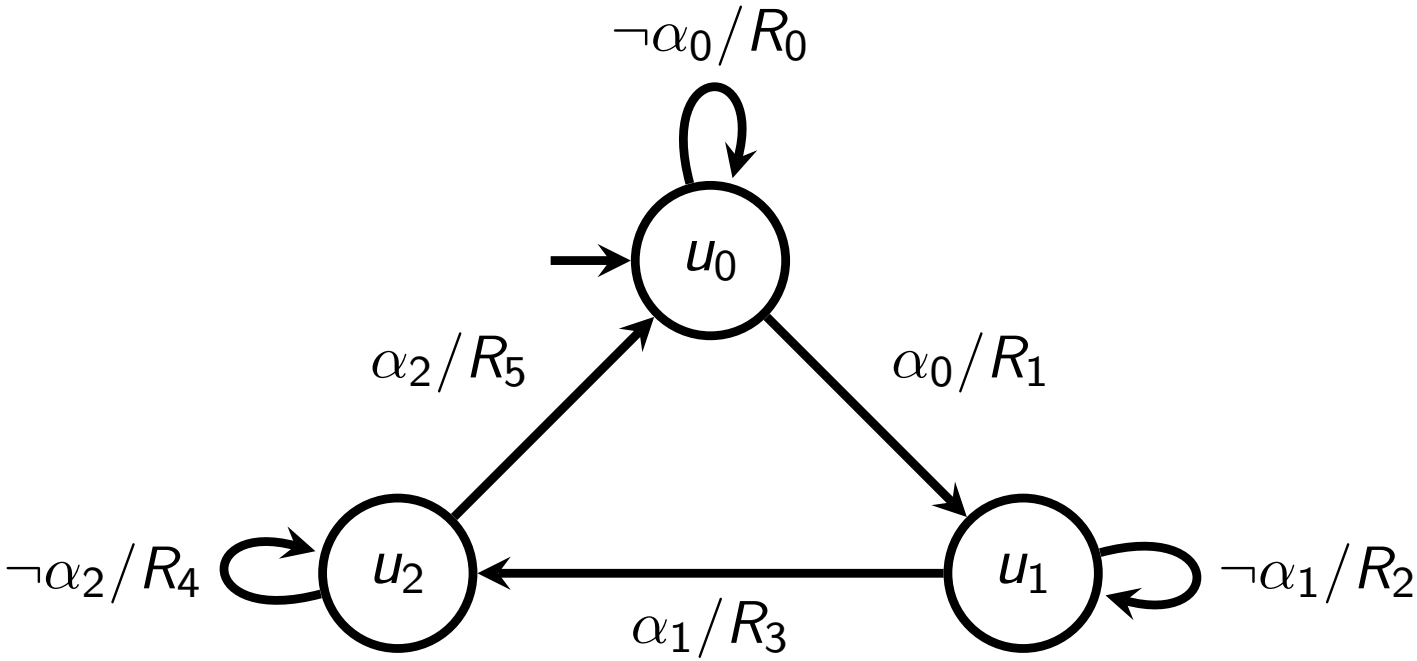
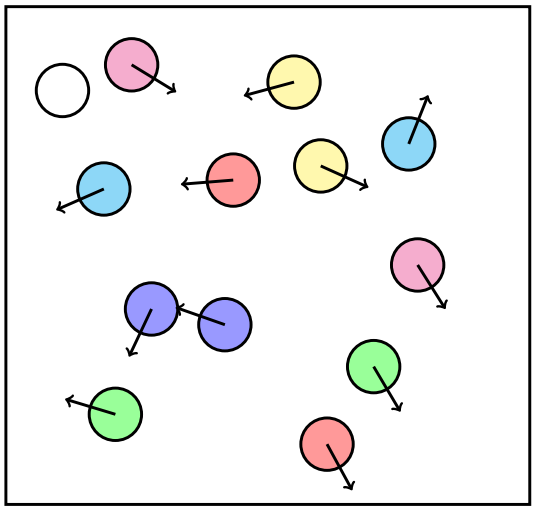
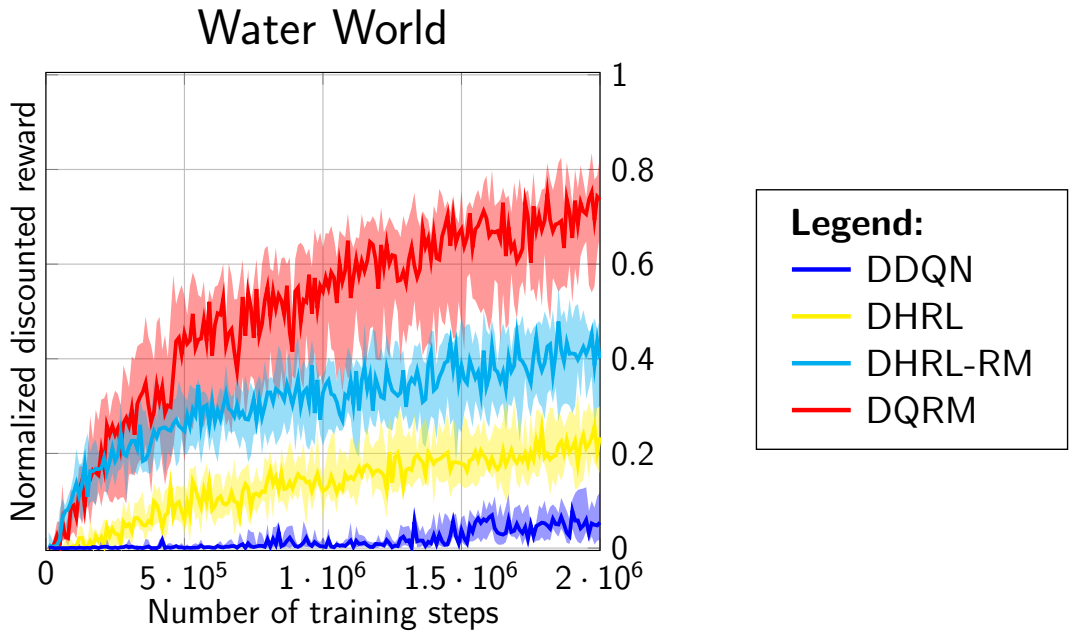


Figure: A **reward machine**

In this case, our approach learns one policy for each node.

More results





Our approach (red curve) finds better policies faster than standard DRL (blue curve) and Hierarchical DRL (yellow and cyan curves).

Paper: "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning"
Code: `https://bitbucket.org/RToroIcarte/qrm`

Outline

1. Reinforcement Learning (RL):
    - What is RL?
    - Two difficulties in applying RL
2. Instructions for Reinforcement Learning
    - LTL formulas
    - Reward Machines
3. Advice for Reinforcement Learning
4. Summary

LTL as an advice language

Advice suggests how to achieve rewards, but does not define the rewards.

Idea:

- Use a **model-based** RL algorithm.
- Guide the exploration with a **heuristic** estimating what actions will make progress towards satisfying the (finite) LTL advice.
  - Good advice can reduce the amount of exploration required to learn a good policy,
  - Bad advice will eventually be recovered from.

Paper: "Advice-Based Exploration in Model-Based Reinforcement Learning"

Summary

Instructions:

- "Teaching Multiple Tasks to an RL Agent using LTL" (AAMAS 2018)
- "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning" (ICML 2018)

Summary

Instructions:

- "Teaching Multiple Tasks to an RL Agent using LTL" (AAMAS 2018)
- "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning" (ICML 2018)

Advice:

- "Advice-Based Exploration in Model-Based Reinforcement Learning" (Canadian AI 2018)

References

Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977. doi: 10.1109/SFCS.1977.32.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 452–461, 2018a.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Advice-based exploration in model-based reinforcement learning. In *Proceedings of the 31st Canadian Conference on Artificial Intelligence*, pages 72–83, 2018b.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2112–2121, 2018c.