LTL and Beyond: Formal Languages for Goal Specification in Reinforcement Learning

Alberto Camacho[†‡] Rodrigo Toro Icarte[†‡] Toryn Q. Klassen[†] Richard Valenzano[§] Sheila A. McIlraith[†‡]

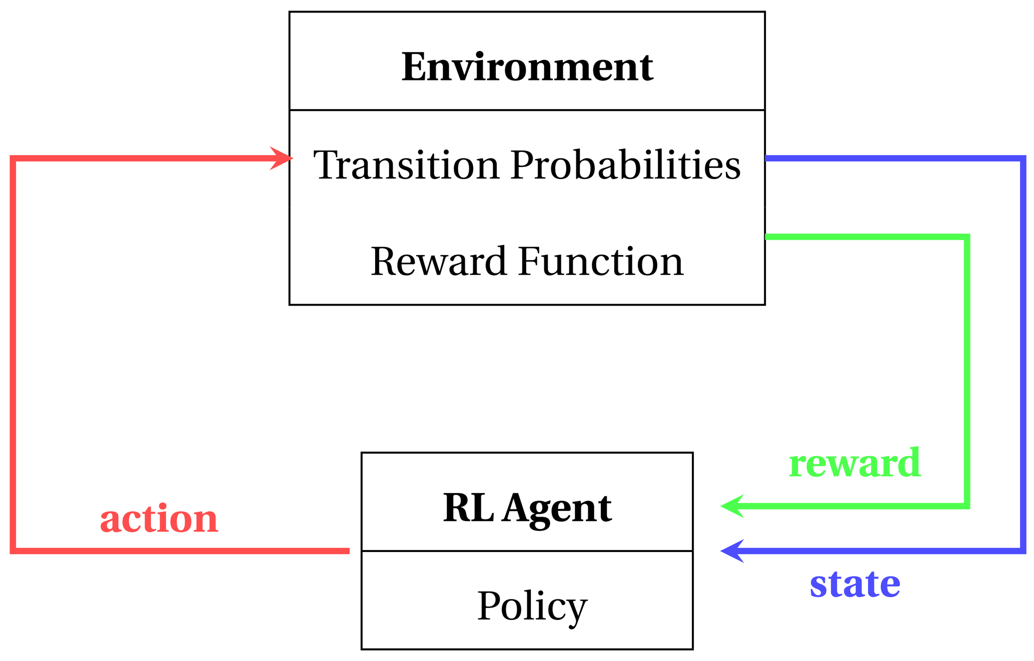
[†]Department of Computer Science, University of Toronto [‡]Vector Institute [§]Element AI

[†]{acamacho, rntoro, toryn, sheila}@cs.toronto.edu, [§]rick.valenzano@elementai.com

Overview

- We propose using reward machines (RMs) as a normal form representation for reward functions.
- LTL formulas and other regular languages can be used to specify reward-worthy behavior that is automatically converted into RMs (via DFAs).
- RM structure can be exploited by Q-learning (QRM) and automated reward shaping to learn policies faster, solving problems that cannot reasonably be solved otherwise.

What is reinforcement learning (RL)?



Reward Specification

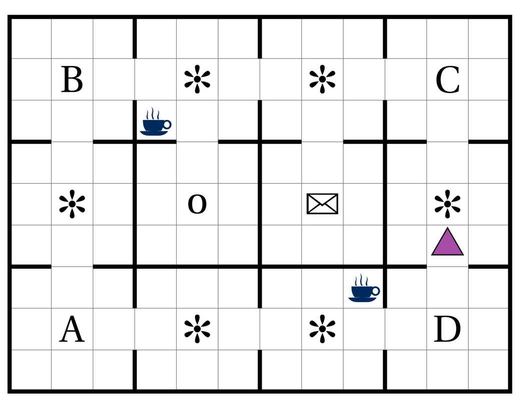
Example Reward-worthy behavior

- Do not vacuum while someone is sleeping
  Always[¬ *(vacuum ∧ sleeping)*]
- If the sign points right then turn right at the intersection
  Always[*right* → **Eventually**[*intersection* → *turn(right)*]]
- While there are dirty dishes on the counter, load them into the dishwasher.
  While ∃*x.dish(x)* ∧ *on(x,Counter)* ∧ *dirty(x)* **do**
  pickup(x); load(x,Dishwasher)
  End while

Challenge: represent the above behavior as a reward function – $R(s,a) \to \mathbb{R}$. Typically done by a programmer via Python code.

What is a Reward Machine (RM)?

Running Example



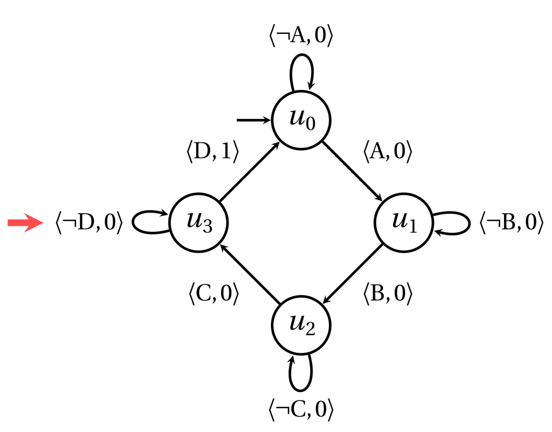
Symbol	Meaning
▲	Agent
✻	Furniture
☕	Coffee machine
✉	Mail room
o	Office
A, B, C, D	Marked locations

Task: Patrol A, B, C, and D.

Reward Machines

Idea: Encode reward functions using finite state machines.
Idea: The vocabulary, $\mathscr{P}$, can be (but need not be) abstracted and human-understandable, realized via low-level event, property, or feature detectors. E.g., $\mathscr{P} = \{☕, ✉, o, ✻, A, B, C, D\}$.

```
m = 0 # global variable
def get_reward(s):
  if m == 0 and s.at("A"):
    m = 1
  if m == 1 and s.at("B"):
    m = 2
  if m == 2 and s.at("C"):
    m = 3
  if m == 3 and s.at("D"):
    m = 0
    return 1
  return 0
```

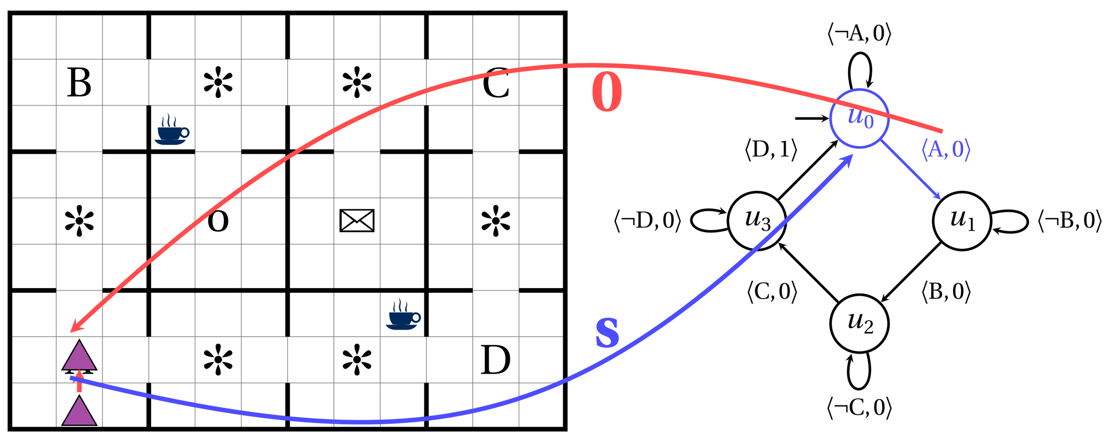


Reward Machines (RM) are Mealy machines where the input alphabet is the set of possible labels and the output alphabet is a set of reward functions. They consist of the following elements:

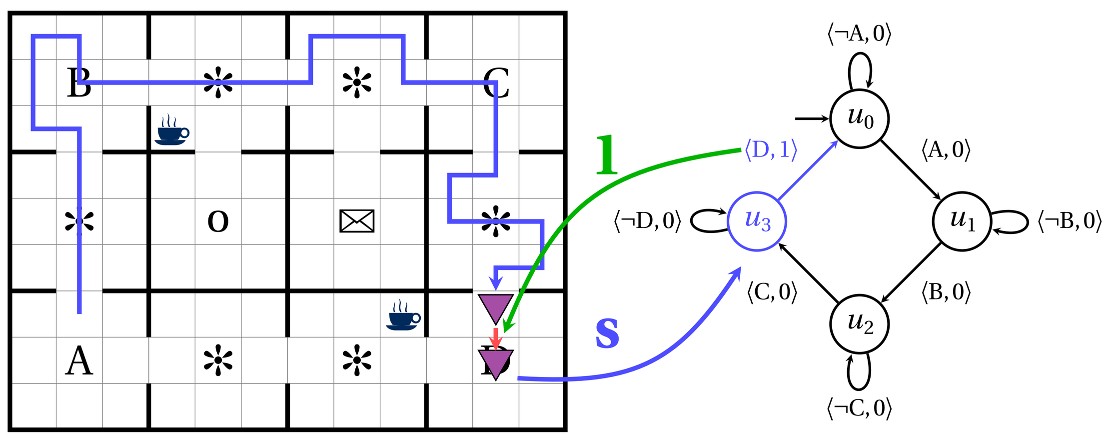
- A finite set of states U.
- An initial state $u_0 \in U$.
- A set of transitions, each labelled by:
  - a logical condition defined over the vocabulary
  - and a reward function.

Reward Machines in Action

This RM starts in u_0 and transitions to u_1 when A is reached. The agent gets reward 0 from that transition's reward function.

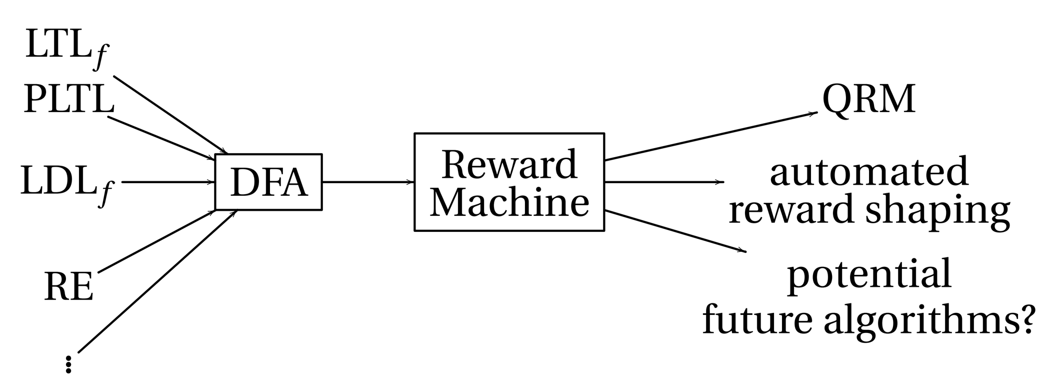


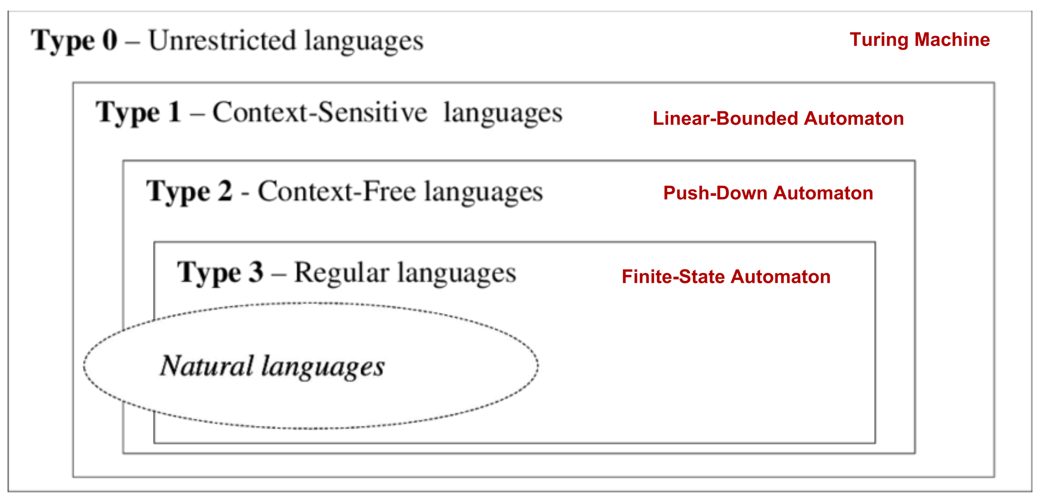
Positive reward is given only when the agent completes a cycle.



RMs as a normal form

Formal Languages RM algorithms



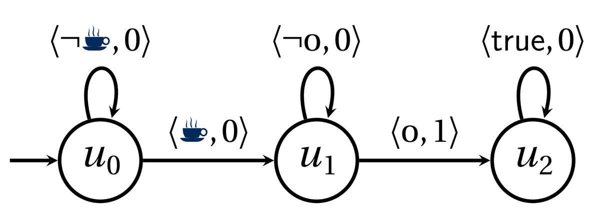


The Chomsky Hierarchy

Example: "*Get coffee and bring it to the office.*"

LTL: **Eventually**[☕ ∧ **Next** [**Eventually** o]]

RM:



Merit:

- Specify reward-worthy behavior in language(s) of choice – RM serves as lingua franca. Behaviors composable.
- Exploit reward function structure without multiple language-specific learning algorithms.

How to exploit an RM's structure

Idea: Give the RM-specified reward function to RL algorithms and tailor learning to the function structure.

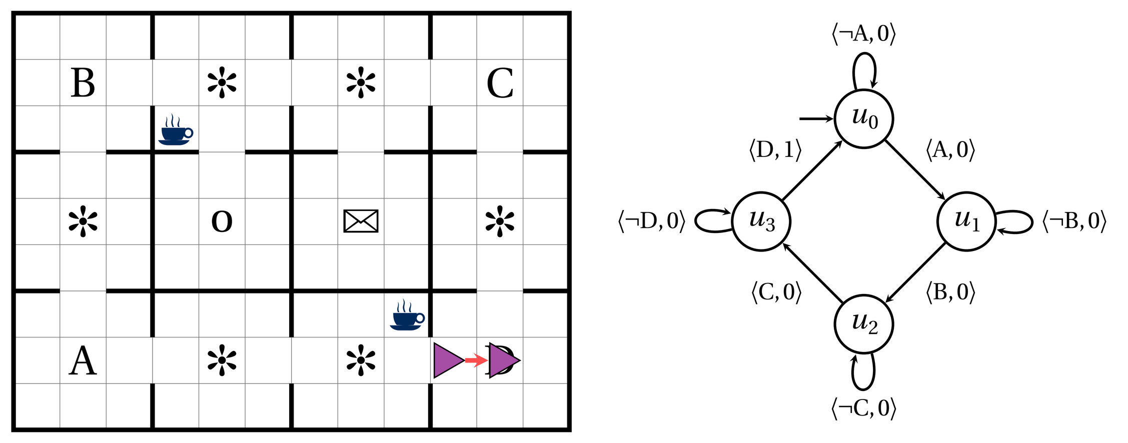
Cross-product baseline

RMs can produce non-Markovian rewards, but we can add the RM state to the agent's state representation and use q-learning:

- Observe state $\langle s,u\rangle$ and execute action $a \sim \pi(a|\langle s,u\rangle)$.
- Observe next state $\langle s',u'\rangle$ and the reward r.
- Improve policy π using experience $\langle\langle s,u\rangle, a, r, \langle s',u'\rangle\rangle$.
- $\langle s,u\rangle \leftarrow \langle s',u'\rangle$.

(1) QRM (Q-learning for Reward Machines)

- Observe state $\langle s,u\rangle$ and execute action $a \sim \pi(a|\langle s,u\rangle)$.
- Observe next state $\langle s',u'\rangle$ and the reward r.
- Improve policy π using $\langle\langle s,u_i\rangle, a, r_{ij}, \langle s',u_j\rangle\rangle$ for all $u_i \in U$.
- $\langle s,u\rangle \leftarrow \langle s',u'\rangle$.

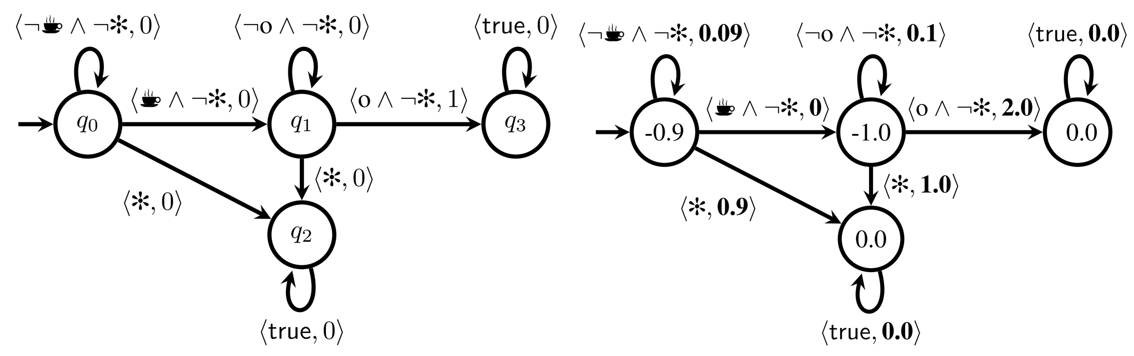


$$\pi \leftarrow \langle\langle s,u_0\rangle, \texttt{move\_right}, 0, \langle s',u_0\rangle\rangle$$
$$\pi \leftarrow \langle\langle s,u_1\rangle, \texttt{move\_right}, 0, \langle s',u_1\rangle\rangle$$
$$\pi \leftarrow \langle\langle s,u_2\rangle, \texttt{move\_right}, 0, \langle s',u_2\rangle\rangle$$
$$\pi \leftarrow \langle\langle s,u_3\rangle, \texttt{move\_right}, 1, \langle s',u_0\rangle\rangle$$

Theorem: QRM converges to an optimal policy in the limit.

(2) Automated Reward Shaping

Idea: Treat the RM as a deterministic MDP, and use value iteration to determine the value of each state. Then, use these values to define potentials for potential-based reward shaping.

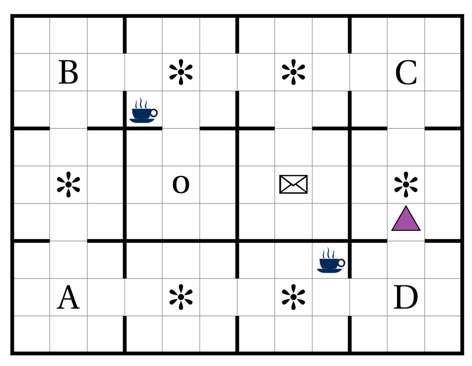


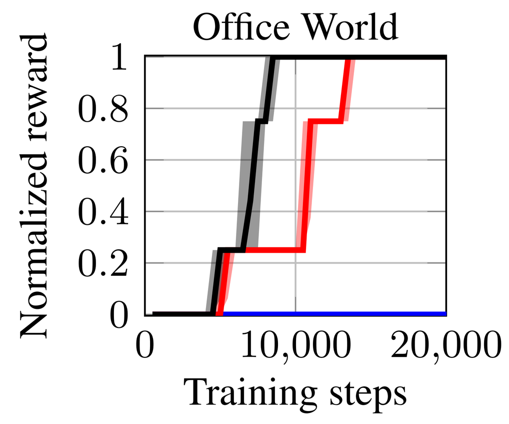
Theorem: Optimal policies are preserved.

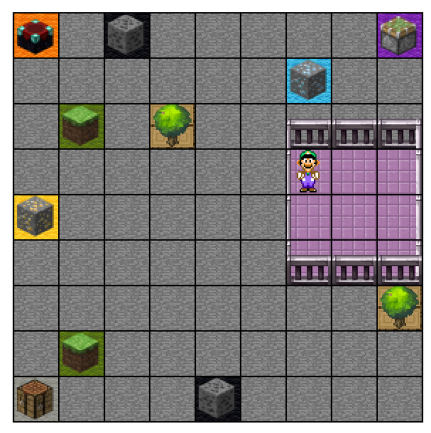
Results

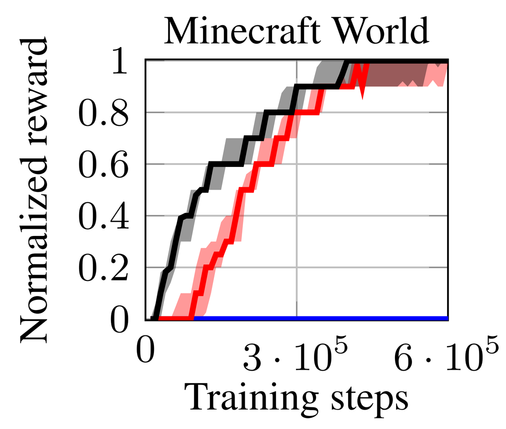
Note: Impressive gain over (deep) Q-learning (blue).

Discrete Domains





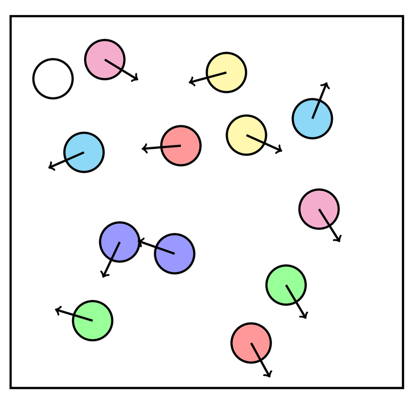


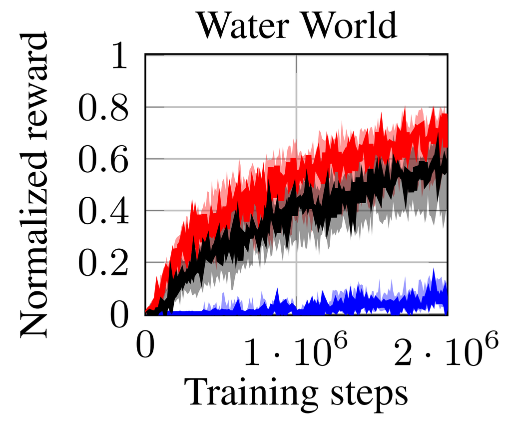


Legend: — Q-Learning — QRM — QRM + RS

Continuous Domains

Deep QRM uses DDQN with prioritized experience replay.





Legend: — DDQN — DQRM — DQRM + RS

See also

Code: `bitbucket.org/RToroIcarte/qrm`

References

[1] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A. McIlraith. Non-Markovian rewards expressed in LTL: guiding search via reward shaping. In *SOCS*, pages 159–160, 2017. A longer version appeared at the First Workshop on Goal Specifications for Reinforcement Learning, colocated with ICML/IJCAI/AAMAS (2018).

[2] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Teaching multiple tasks to an RL agent using LTL. In *AAMAS*, pages 452–461, 2018.

[3] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, pages 2112–2121, 2018.