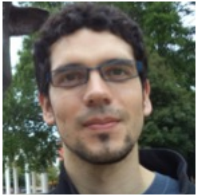
LTL and Beyond:
Formal Languages for Reward Function Specification in Reinforcement Learning

Alberto Camacho

Rodrigo Toro Icarte

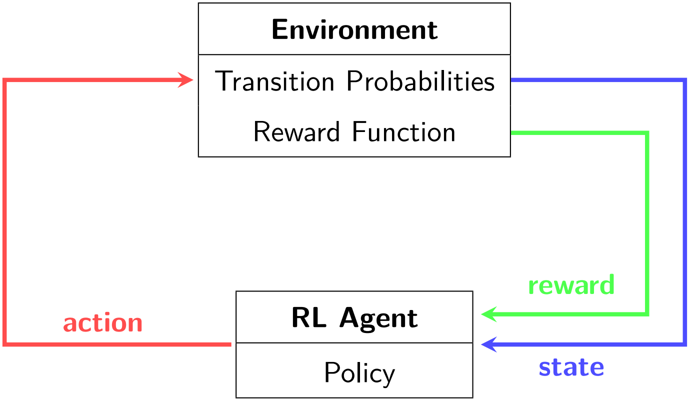
Toryn Q. Klassen

Richard Valenzano

Sheila A. McIlraith

Previously published in IJCAI 2019, pp. 6065–6073

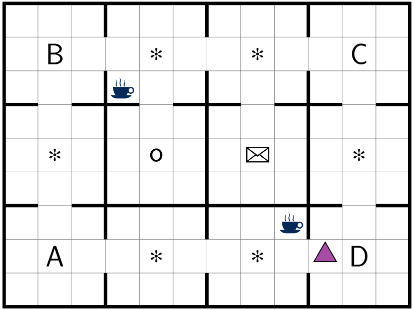
Reinforcement learning (RL)



Takeaway points

- **Reward machines (RMs)** are a form of automaton that are a way of representing (temporally extended) reward functions.
- Formulas in many temporal languages (e.g., LTL_f) can be **translated** into RMs.
- Once a reward function is represented as an RM, its structure can be exploited by various RM-specific **algorithms** for more efficient reinforcement learning.

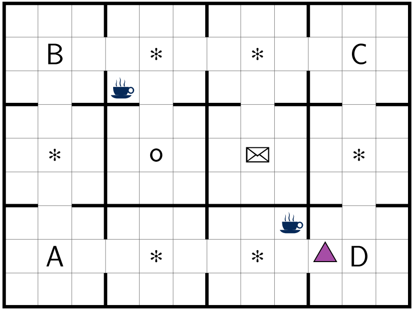
Example (the Office World)



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

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

Example (the Office World)

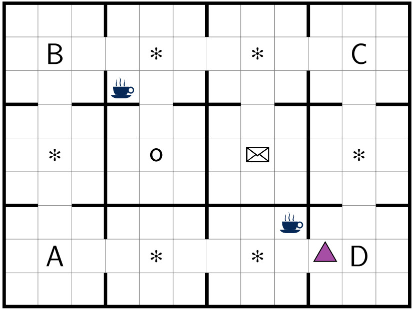


Symbol	Meaning
△	Agent
✻	Decoration
☕	Coffee machine
✉	Mail room
o	Office
A,B,C,D	Marked locations

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

- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.

Example (the Office World)

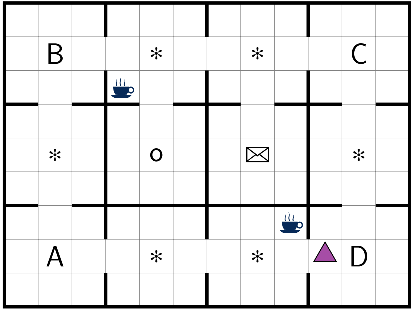



```python
m = 0 # global variable
def get_reward(s):
  if m == 0 and s.at("A"):
    m = 1
  if m == 1 and s.at("B"):
    m = 2
  if m == 2 and s.at("C"):
    m = 3
  if m == 3 and s.at("D"):
    m = 0
    return 1
  return 0
```

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

- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.

Example (the Office World)



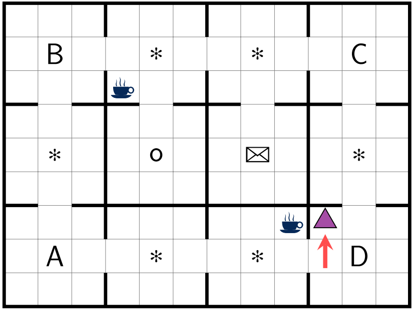
[redacted]

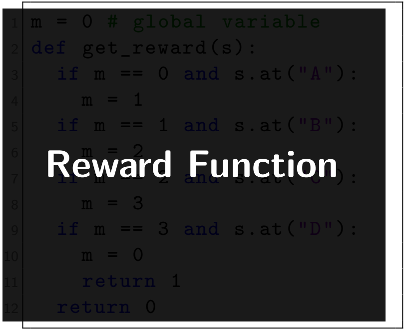
Reward Function

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

- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.
- The reward function is then often treated as a **black box**.

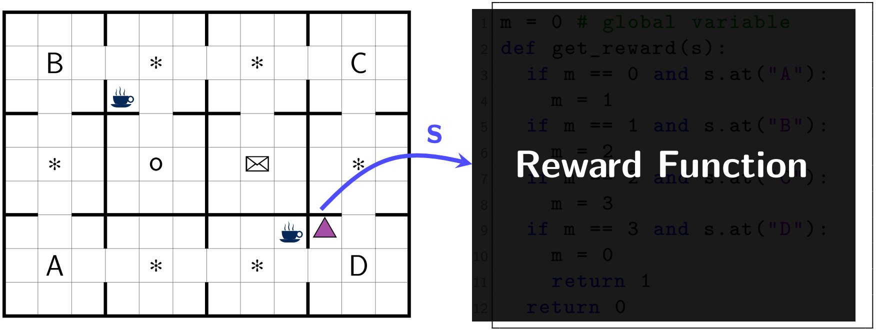
Example (the Office World)



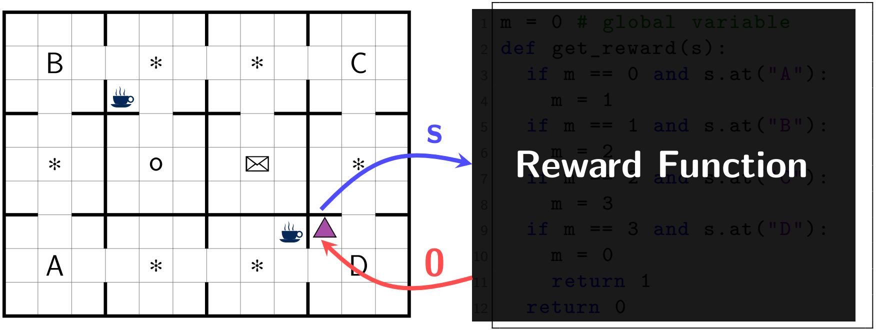


- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.
- The reward function is then often treated as a **black box**.

Example (the Office World)



- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.
- The reward function is then often treated as a **black box**.



- For RL, the task has to be specified in terms of a reward function
  - which might be derived from a formula in **Linear Temporal Logic (LTL)** or some other formal language, or programmed directly.
- The reward function is then often treated as a **black box**.

What if we exposed the reward function structure?

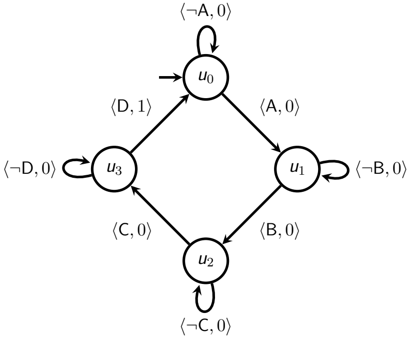
We previously introduced **reward machines** in an ICML paper.[1]

Suppose we have a vocabulary $\mathcal{P}$ to label environment states, e.g.,

$$\mathcal{P} = \{☕, \boxtimes, \mathsf{o}, \circledast, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}.$$

A **reward machine (RM)** has

- A finite set of states, with an initial state u_0
- A set of transitions labelled by:
  - a logical condition (using the vocabulary) and
  - a reward (or more generally a reward function).



[1] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *ICML*. 2018, pp. 2112–2121.

What if we exposed the reward function structure?

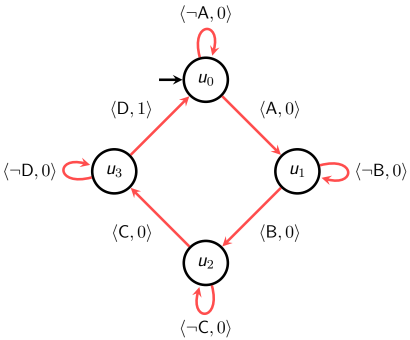
We previously introduced **reward machines** in an ICML paper.[1]

Suppose we have a vocabulary $\mathcal{P}$ to label environment states, e.g.,

$$\mathcal{P} = \{☕, \boxtimes, \mathsf{o}, \circledast, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}.$$

A **reward machine (RM)** has

- A finite set of states, with an initial state u_0
- **A set of transitions labelled by:**
  - a logical condition (using the vocabulary) and
  - a reward (or more generally a reward function).



[1] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *ICML*. 2018, pp. 2112–2121.

What if we exposed the reward function structure?

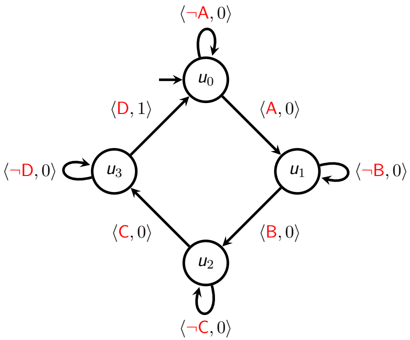
We previously introduced **reward machines** in an ICML paper.[1]

Suppose we have a vocabulary $\mathcal{P}$ to label environment states, e.g.,

$$\mathcal{P} = \{\text{☕}, \boxtimes, \mathsf{o}, \circledast, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}.$$

A **reward machine (RM)** has

- A finite set of states, with an initial state u_0
- A set of transitions labelled by:
  - **a logical condition** (using the vocabulary) and
  - a reward (or more generally a reward function).



[1] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *ICML*. 2018, pp. 2112–2121.

What if we exposed the reward function structure?

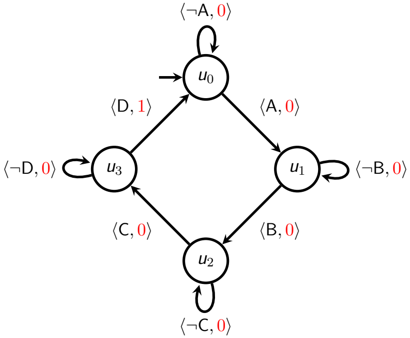
We previously introduced **reward machines** in an ICML paper.[1]

Suppose we have a vocabulary $\mathcal{P}$ to label environment states, e.g.,

$$\mathcal{P} = \{☕, ✉, o, *, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}.$$

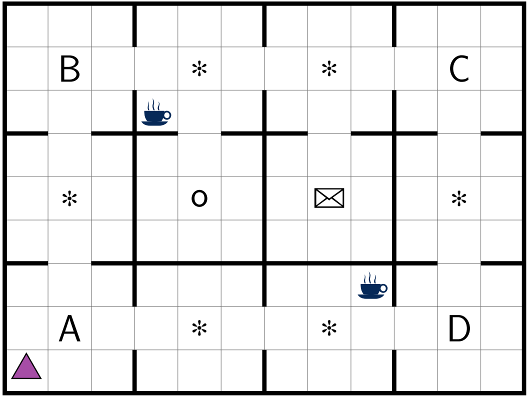
A **reward machine (RM)** has

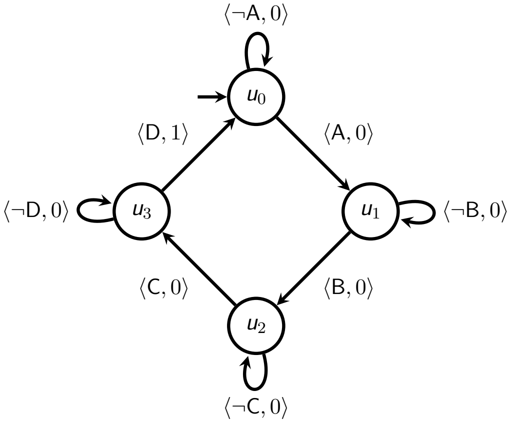
- A finite set of states, with an initial state u_0
- A set of transitions labelled by:
  - a logical condition (using the vocabulary) and
  - **a reward** (or more generally a reward function).



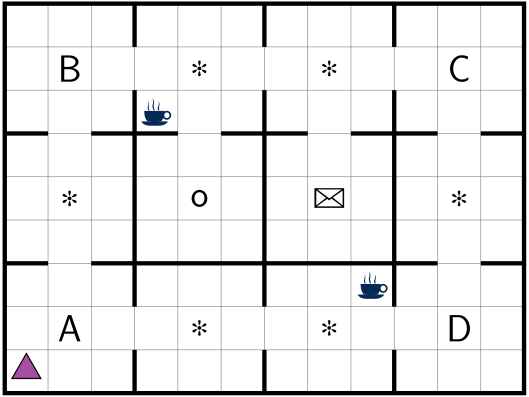
[1] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *ICML*. 2018, pp. 2112–2121.

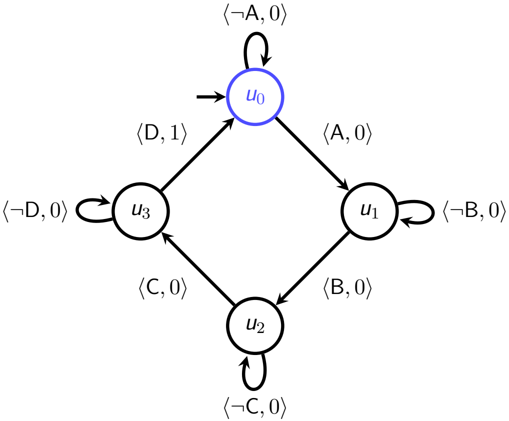
How does a reward machine work?



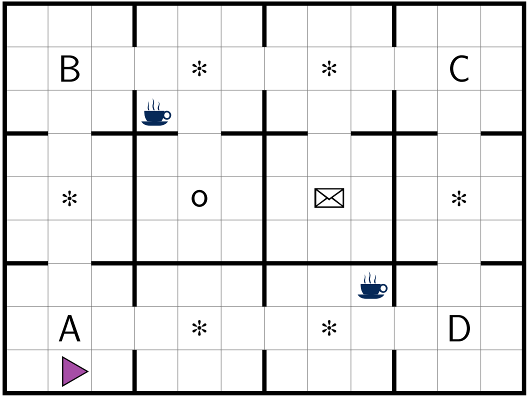


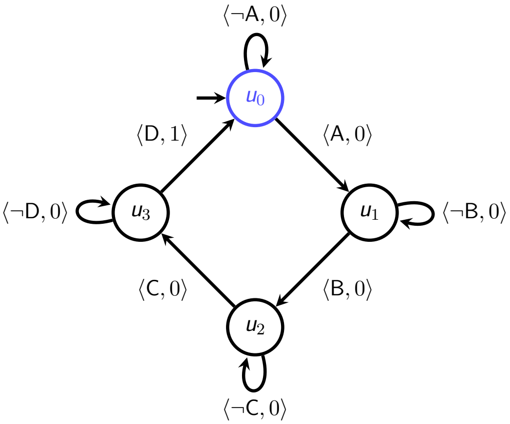
How does a reward machine work?



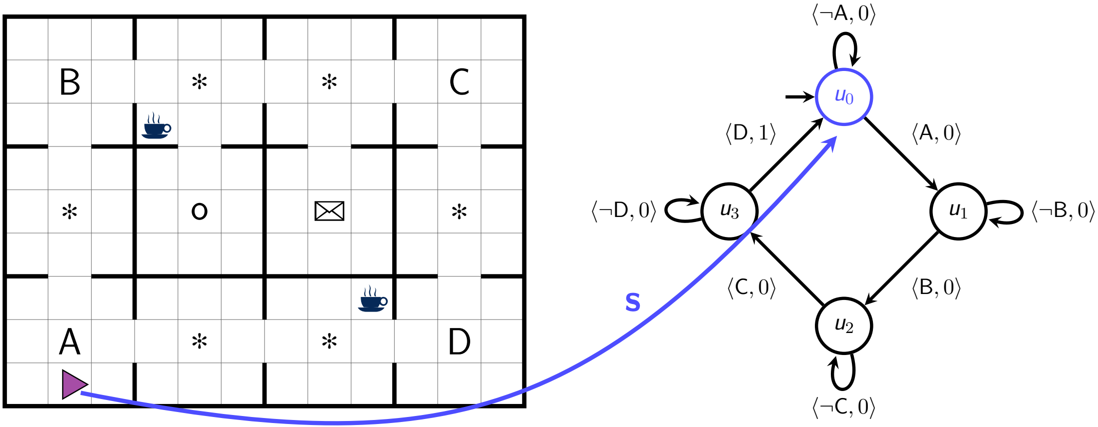


How does a reward machine work?

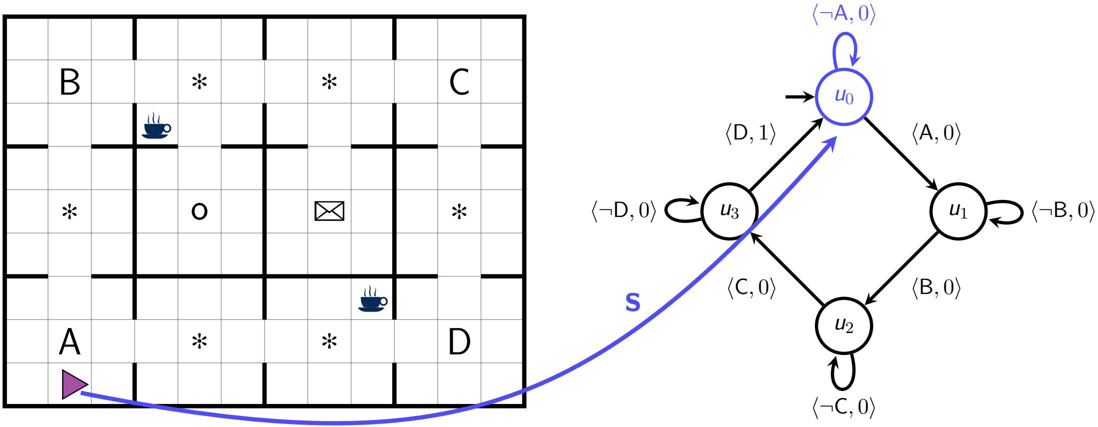




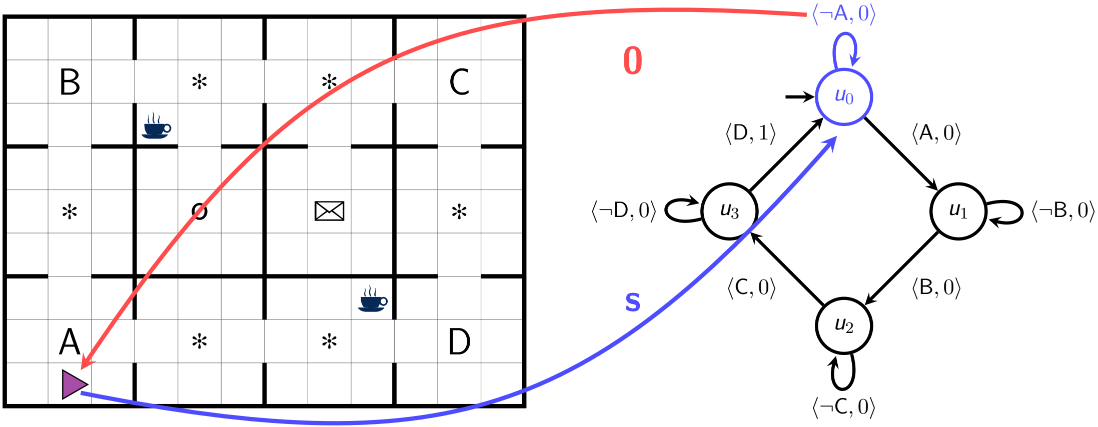
How does a reward machine work?



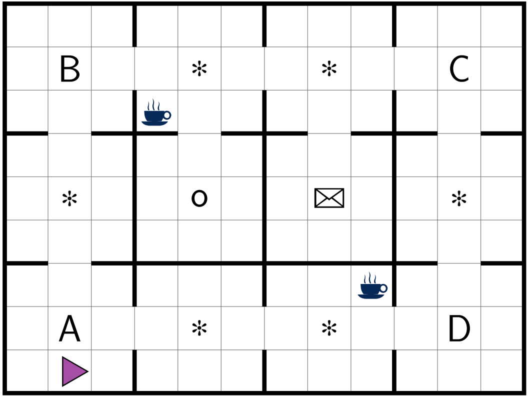
How does a reward machine work?

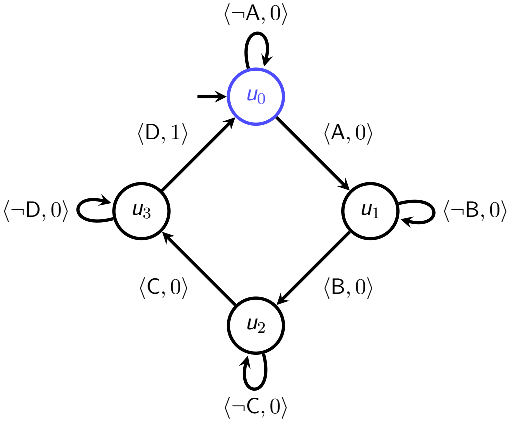


How does a reward machine work?

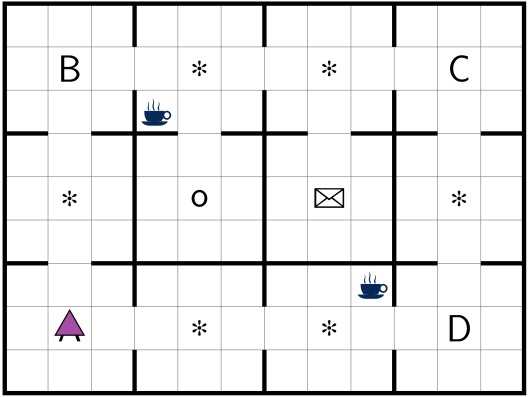


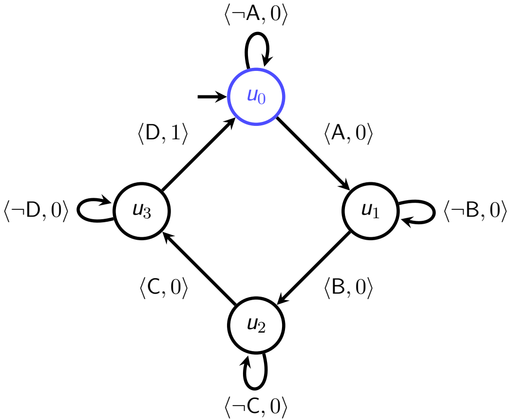
How does a reward machine work?



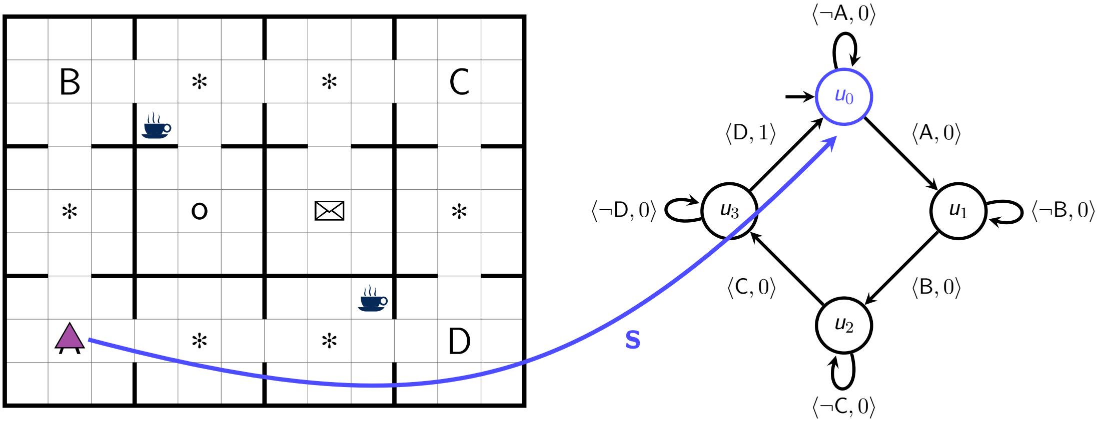


How does a reward machine work?

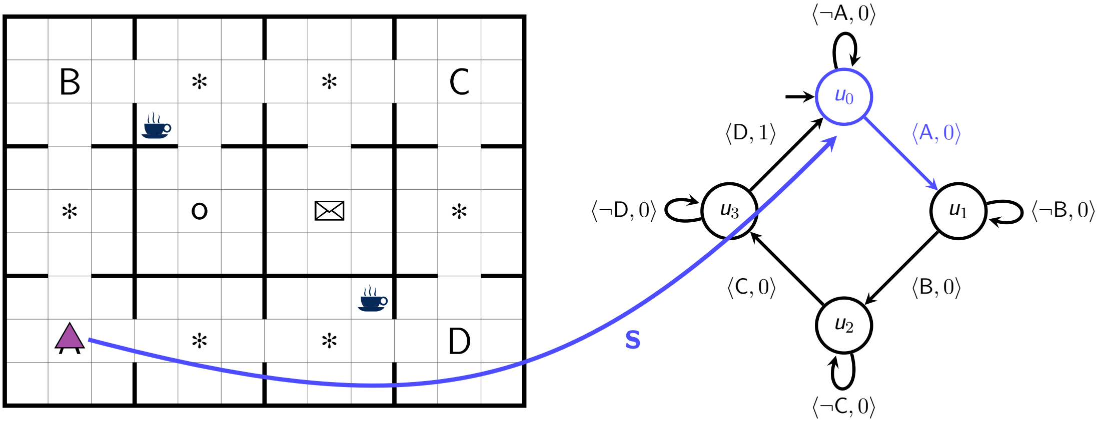




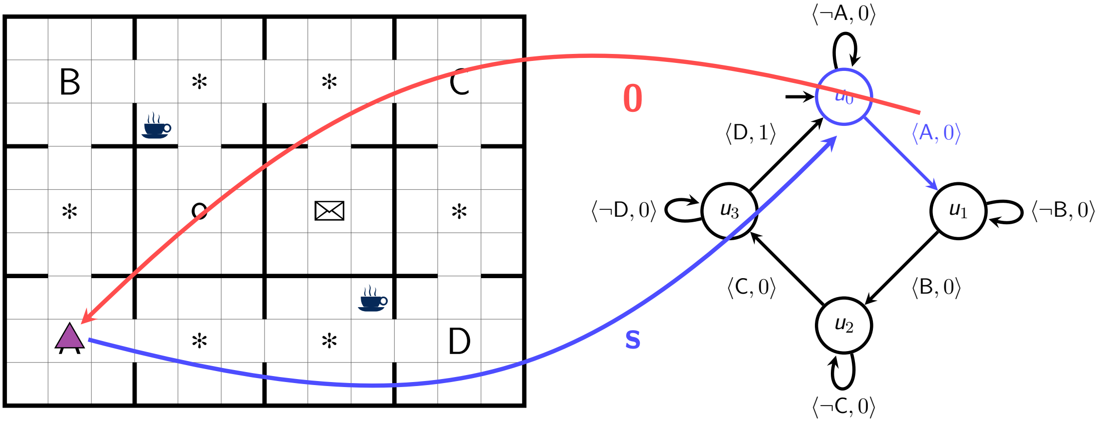
How does a reward machine work?



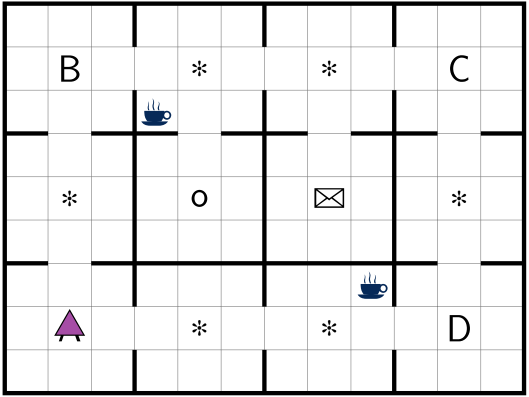
How does a reward machine work?

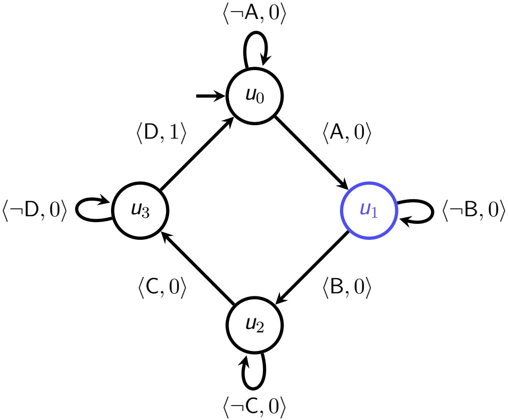


How does a reward machine work?

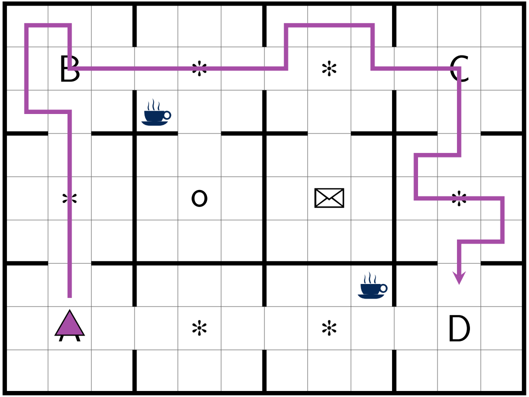


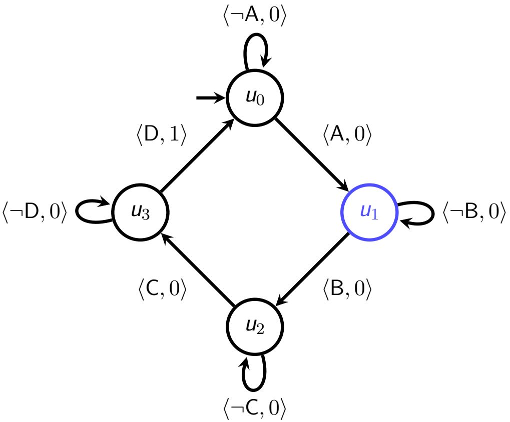
How does a reward machine work?



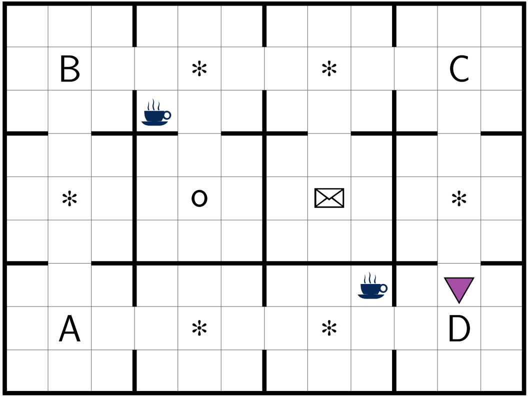


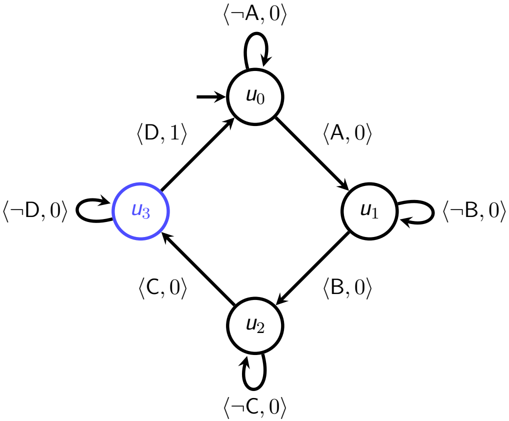
How does a reward machine work?



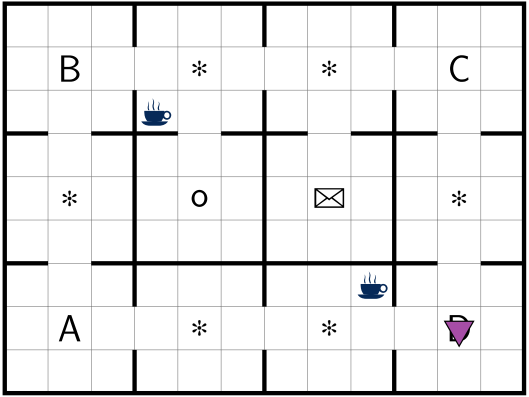


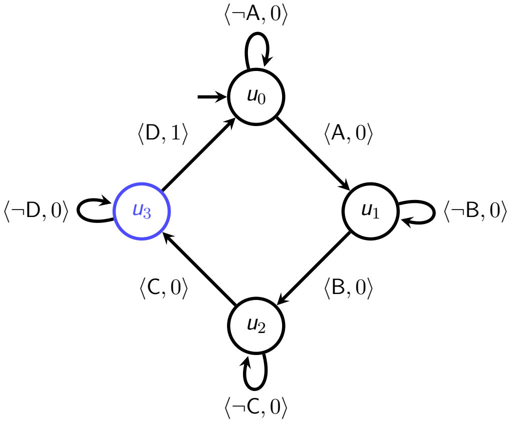
How does a reward machine work?



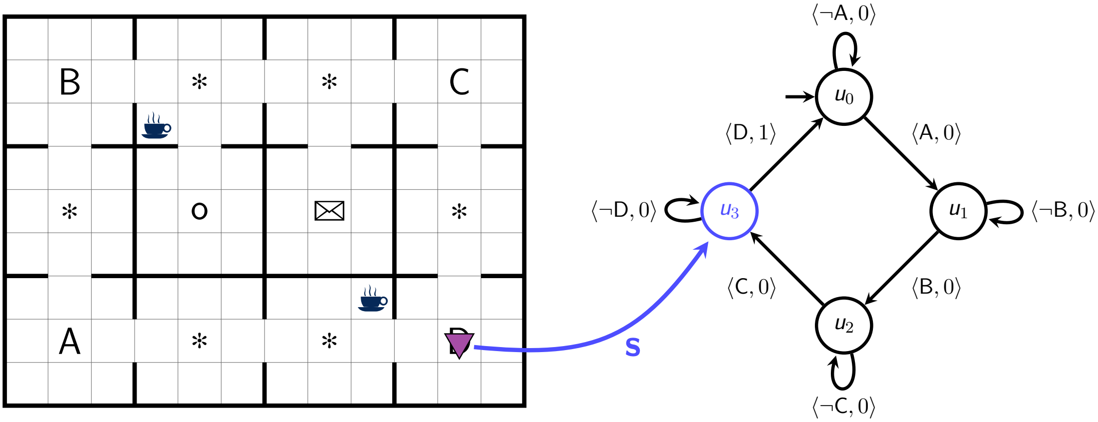


How does a reward machine work?

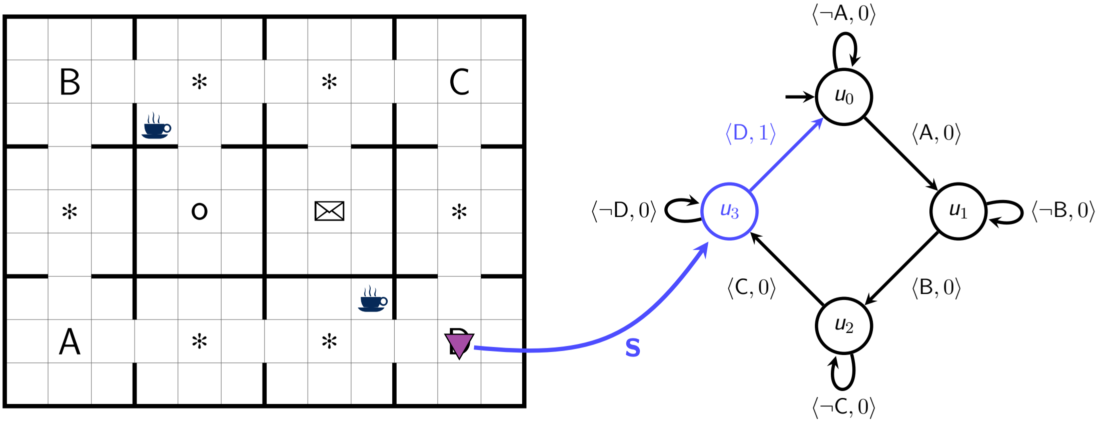




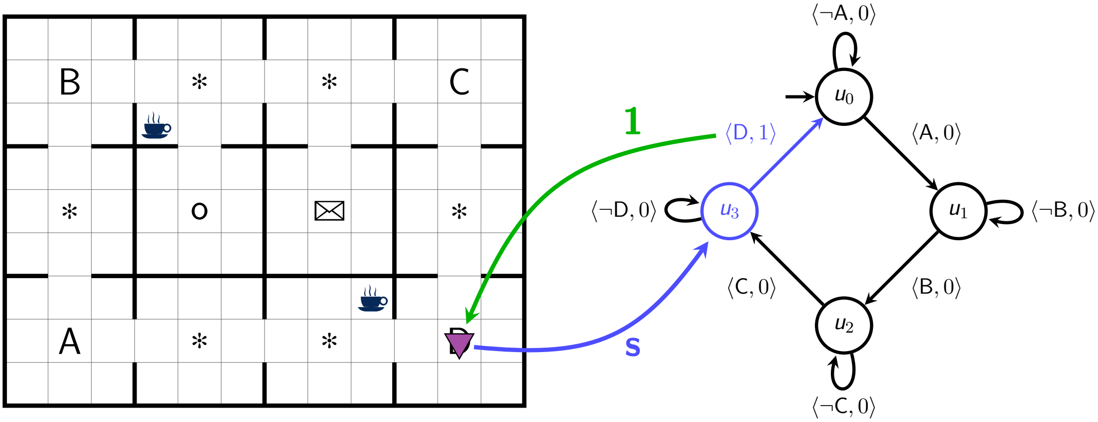
How does a reward machine work?



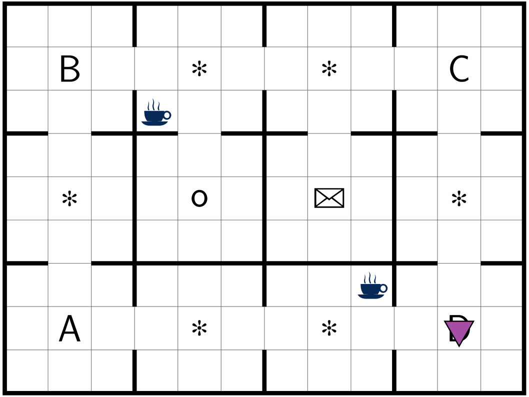
How does a reward machine work?

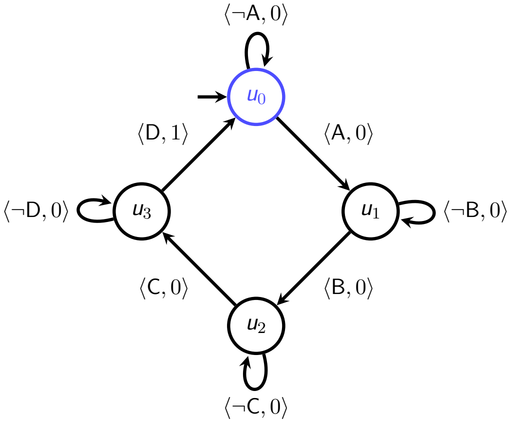


How does a reward machine work?

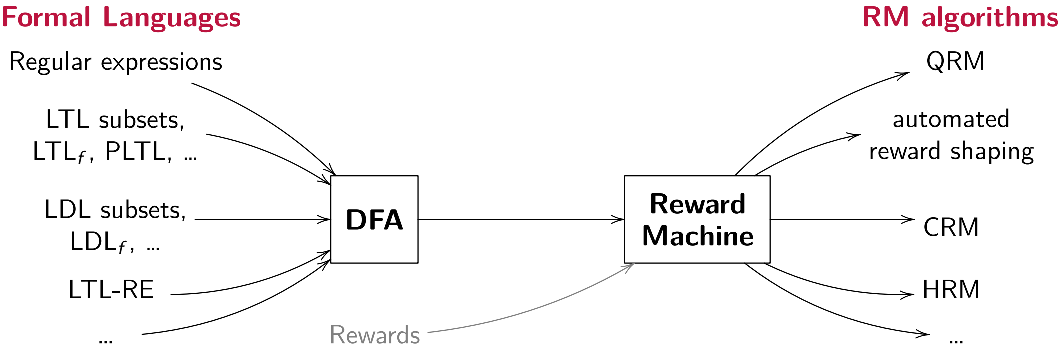


How does a reward machine work?



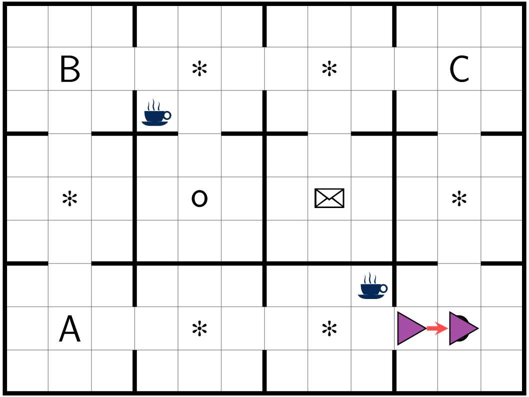


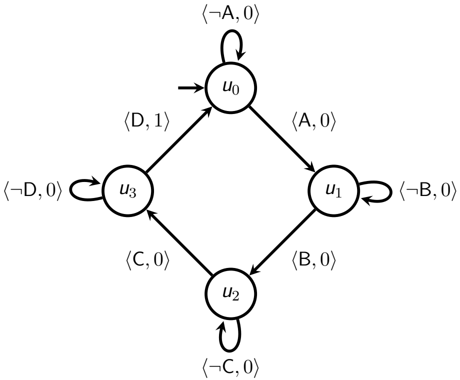
Transforming formal language specifications into reward machines



The QRM algorithm[2]

When the agent acts while in RM state u_i, we can compute what reward **would have been** received if had acted in any other RM state u_j.



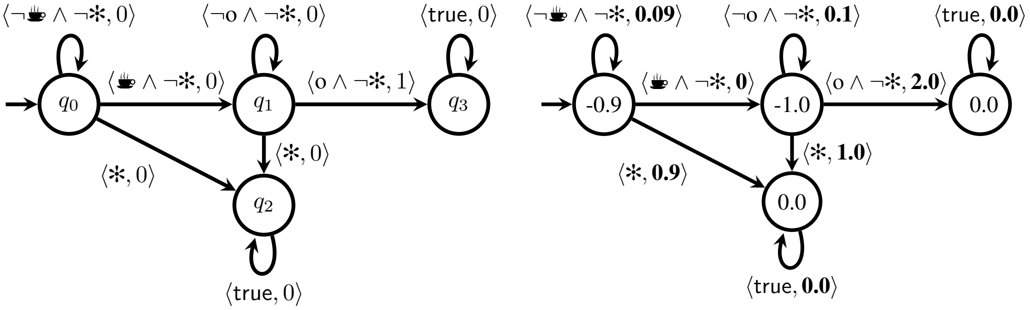


QRM generates such synthetic, **counterfactual** experiences for use in training.

[2] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *ICML*. 2018, pp. 2112–2121.

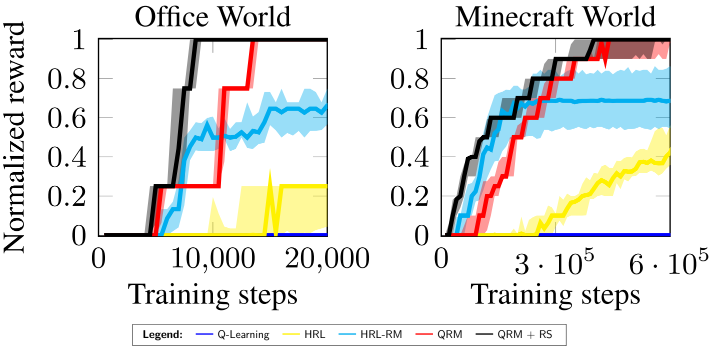
Automated reward shaping

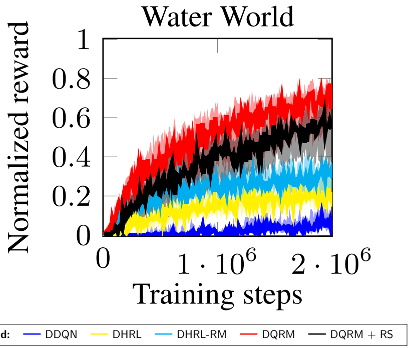
- Treat the RM itself as a (deterministic) MDP, and use **value iteration** to determine the value of each state.
- Then, use these values to define potentials for **potential-based**[3] reward shaping.



[3] Andrew Y. Ng et al. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML*. 1999, pp. 278–287.

Experimental results





By exploiting reward machine structure, our algorithms outperform the baselines.

Conclusion

- **Reward machines (RMs)** are a form of automaton that are a way of representing (temporally extended) reward functions.
- Formulas in many temporal languages (e.g., LTL_f) can be **translated** into RMs.
- Once a reward function is represented as an RM, its structure can be exploited by various RM-specific **algorithms** for more efficient reinforcement learning.

Code:

- Reward machine **algorithms**: https://bitbucket.org/RToroIcarte/qrm
- **Translating** formal languages into reward machines: http://fl-at.jaimemiddleton.cl/[4]

[4] Jaime Middleton et al. *FL-AT: A Formal Language–Automaton Transmogrifier*. System demonstration at ICAPS 2020. 2020.