

How (formal) language can help AI agents learn, plan, and remember

Sheila A. McIlraith

Department of Computer Science, University of Toronto
Vector Institute for Artificial Intelligence
Schwartz Reisman Institute for Technology and Society



LANGUAGE

Humans have evolved languages over tens of thousands of years to provide useful abstractions for understanding and interacting with each other and with the physical world.

The claim advanced by some is that language influences what we think, what we perceive, how we focus our attention, and what we remember.

We use language to capture our understanding of the world around us, to communicate high-level goals, intentions and objectives, and to support coordination with others.

In Computer Science and in the study of AI, we use knowledge representation languages and programming languages to capture our understanding of the world and to communicate unambiguously with computers.

Importantly, language can provide us with useful and purposeful abstractions that can help us to generalize and transfer knowledge to new situations.



FORMAL LANGUAGE

(Some) affordances of formal language

- **Well-defined semantics** (unambiguously computer interpretable)
- **Compositional** syntax and semantics
- Some languages are **declarative** (elaboration tolerant)
- Typically **concise**
- Generally have a “proof system” (of sorts) for **sound reasoning**
- Often supports **generalization**
- ...
- **Gateway to established theories and methods:**
 - **Reasoning with theories:** E.g., physics, chemistry (stoichiometry), music, ...
 - **Formal methods:** verification, automated synthesis, ...
 - **AI KR&R:** default & commonsense reasoning, deontics, PDDL, probabilistic relational models, ...
 - ...

A Thought Experiment



Photo: Javier Pierin (Getty Images)

A Thought Experiment

How do we advise, instruct, task, ... and impart knowledge to our AI?



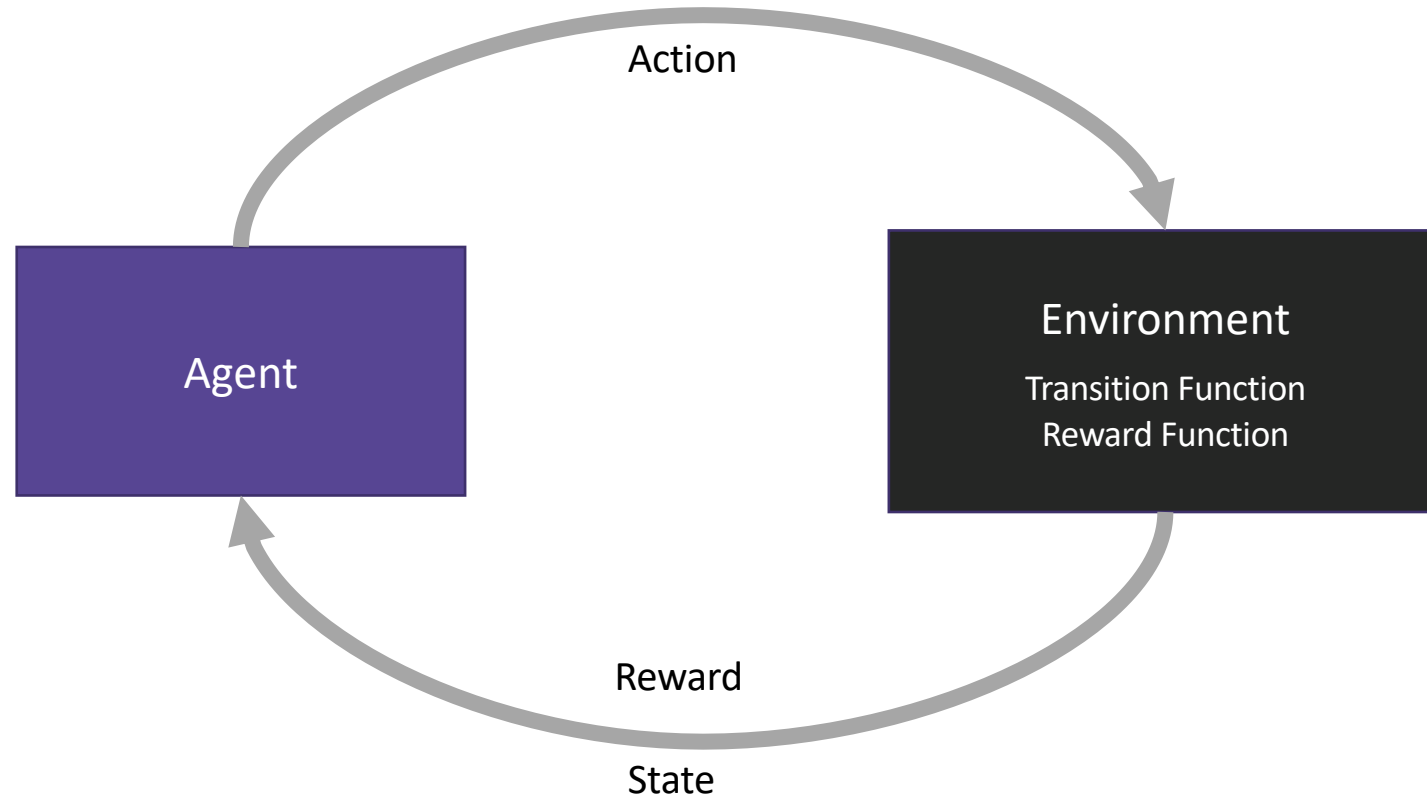
A Thought Experiment

How do we advise, instruct, task, ... and impart knowledge to our AI?

... and how does an AI exploit this knowledge to learn and act?

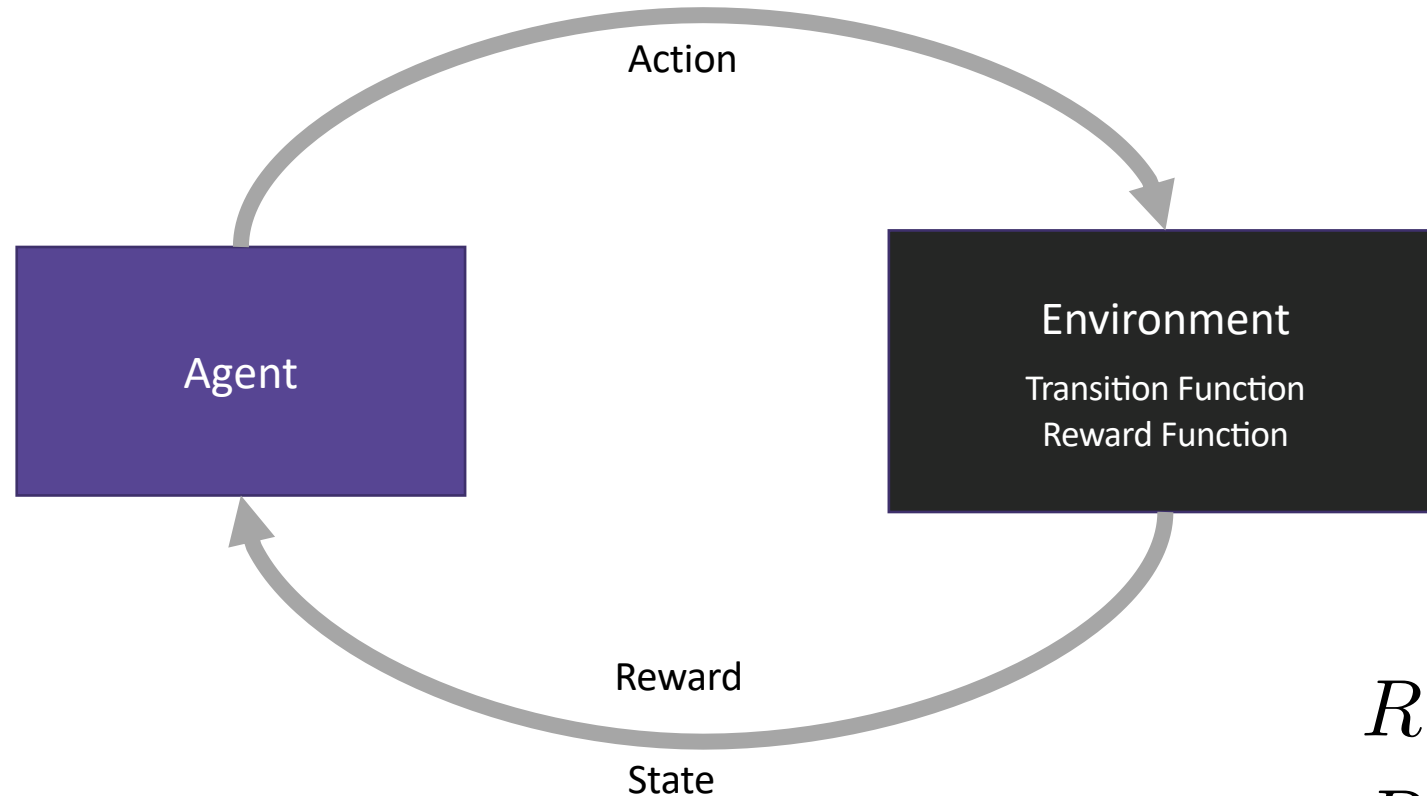


Reinforcement Learning (RL)



Following Sutton and Barto, 2018

Reinforcement Learning (RL)



$$R(s) \rightarrow \mathbb{R}$$
$$R(s, a, s') \rightarrow \mathbb{R}$$

Challenges to RL

- **Reward Specification:** It's hard to define reward functions for complex tasks.
- **Sample Efficiency:** RL agents might require billions of interactions with the environment to learn good policies.
- **Partial Observability**

Goals, Preferences, Norms, Laws ...

- Run the dishwasher when it's full or when dishes are needed for the next meal.
- Make sure the bath temperature is between 32 – 40 celcius immediately before letting someone enter the bathtub.
- Do not vacuum while someone in the house is sleeping.
- ...
- Always serve the person who has been waiting the longest
- If you're driving in Quebec, never turn right on a red light.

... (Procedural) Instructions

When assisting someone from a car to the sidewalk, please always open the car door closest to the person, help the person to standing, move them beyond the car door, close the car door, walk them to the sidewalk.

How do we communicate this to our RL agent?

How do we communicate this to our RL agent?

Formal Language

Linear Temporal Logic (LTL)

A compelling logic to express temporal properties of traces.

Syntax

Logic connectives: \wedge, \vee, \neg

LTL basic operators:

- **next:** $\bigcirc\varphi$
- weak next: $\bigcirc\!\!\!\bigcirc\varphi$
- **until:** $\psi \text{ U } \chi$

Other LTL operators:

- **eventually:** $\diamond\varphi \stackrel{\text{def}}{=} \text{true U } \varphi$
- **always:** $\square\varphi \stackrel{\text{def}}{=} \neg\diamond\neg\varphi$
- release: $\psi \text{ R } \chi \stackrel{\text{def}}{=} \neg(\neg\psi \text{ U } \neg\chi)$

Properties

- Interpreted over **finite** or **infinite** traces.
- Can be **transformed into automata**.



**Remember
this!**

Goals and Preferences

- Do not vacuum while someone is sleeping

$$\Box[\neg(\textit{vacuum} \wedge \textit{sleeping})]$$

i.e., **always** $[\neg(\textit{vacuum} \wedge \textit{sleeping})]$

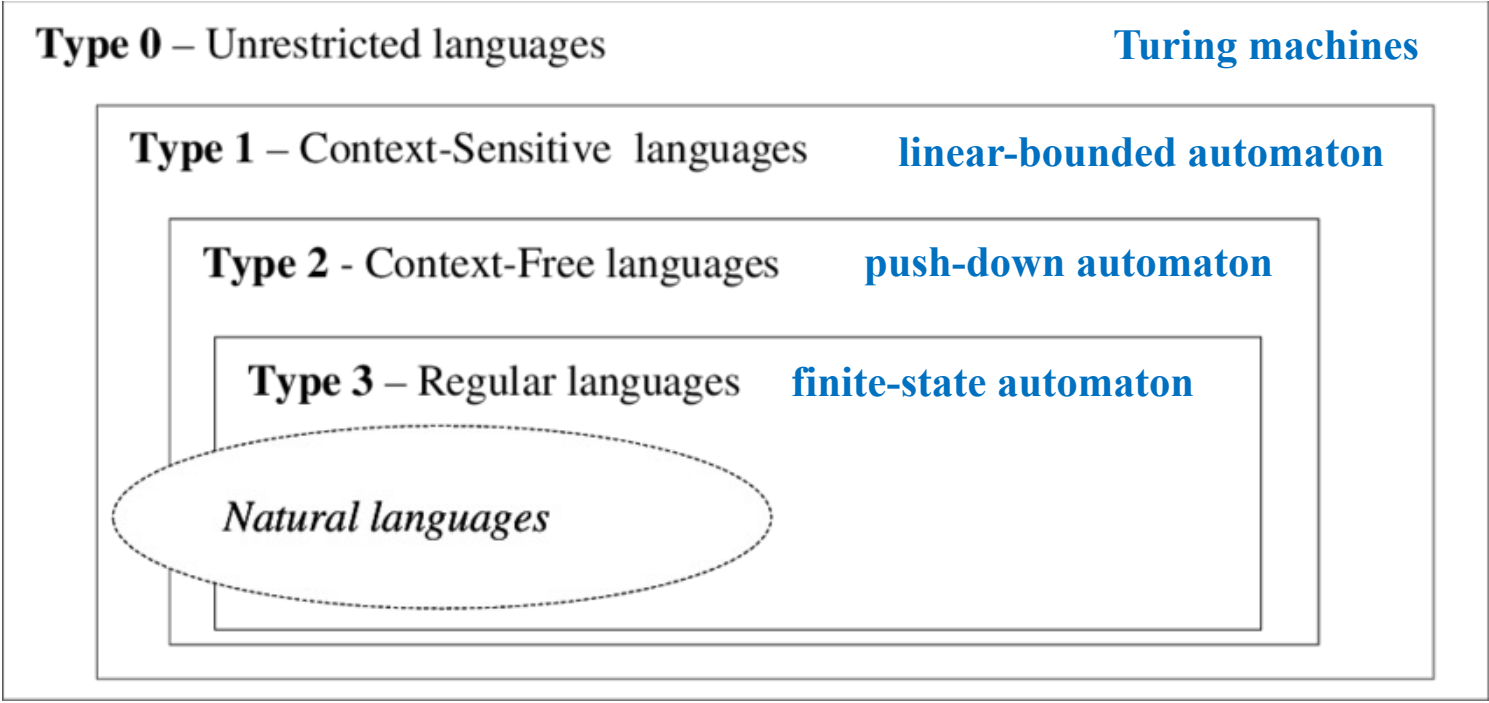
How do we communicate this to our RL agent?

Formal Language



Automata

Formal Language - Automata Correspondence



Noam Chomsky

Chomsky Hierarchy

(Some) affordances of automata

- **Normal-form representation**
- **Exposes function structure** – allowing for compositional learning and transfer
- Distinguish **contexts** – a Markovian decomposition
- A pathway to understanding computation ...

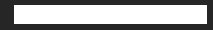
As we will see

- Supports **counterfactual reasoning**
- Define a sublanguage
 - Serve as **generators and a recognizers** of the traces accepted by an automaton.
Can be used to **generate data**.

The alphabet, compositional syntax, and semantics of (formal) language can help RL agents learn, learn what to remember, and plan.

The Rest of the Talk

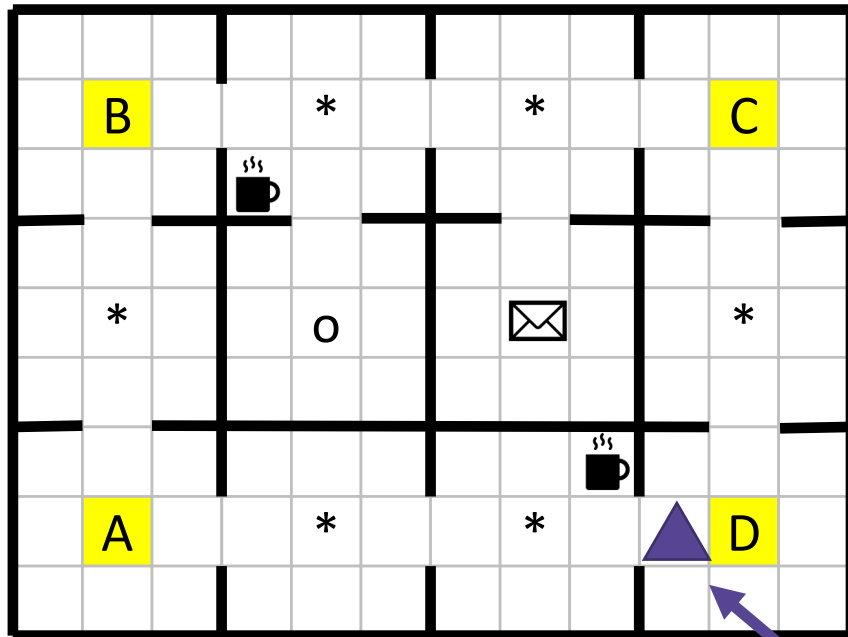
- ▶ **Reward Machines (RM)**
 - **Exploiting RM Structure in Learning**
 - **Experiments**
 - **Creating Reward Machines**
 - **Concluding Remarks**



REWARD MACHINES

Simple Example:

Go to A, then B, then C, then D

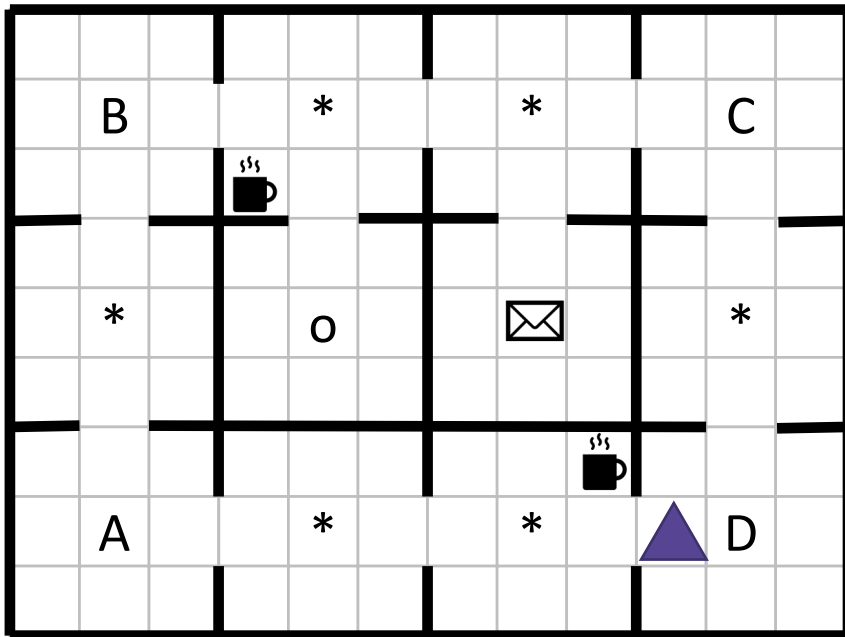


Office World

Agent

Reward Function:

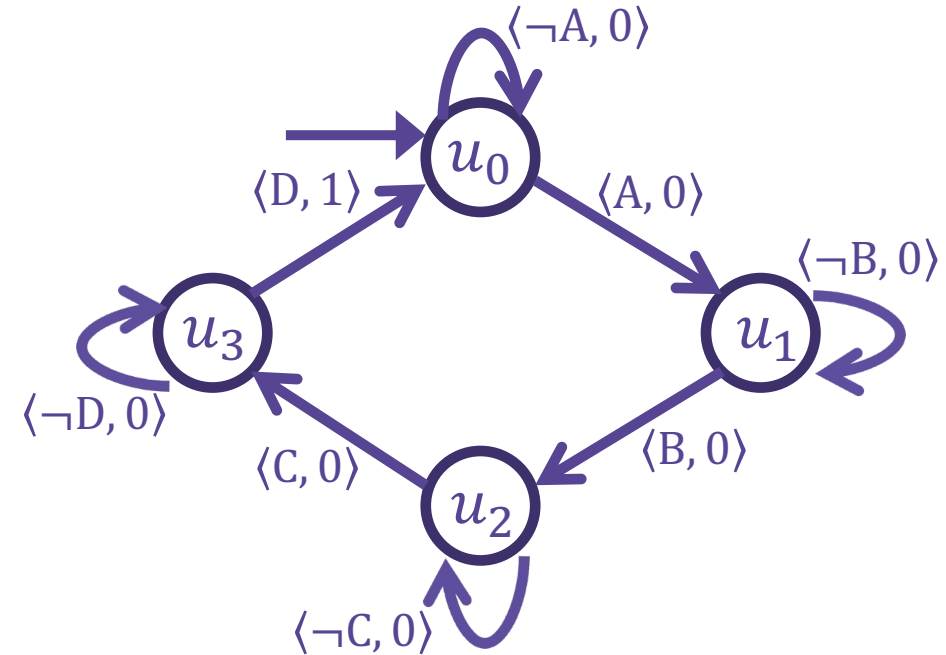
Go to A, then B, then C, then D



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


Define a Reward Function using a Reward Machine

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

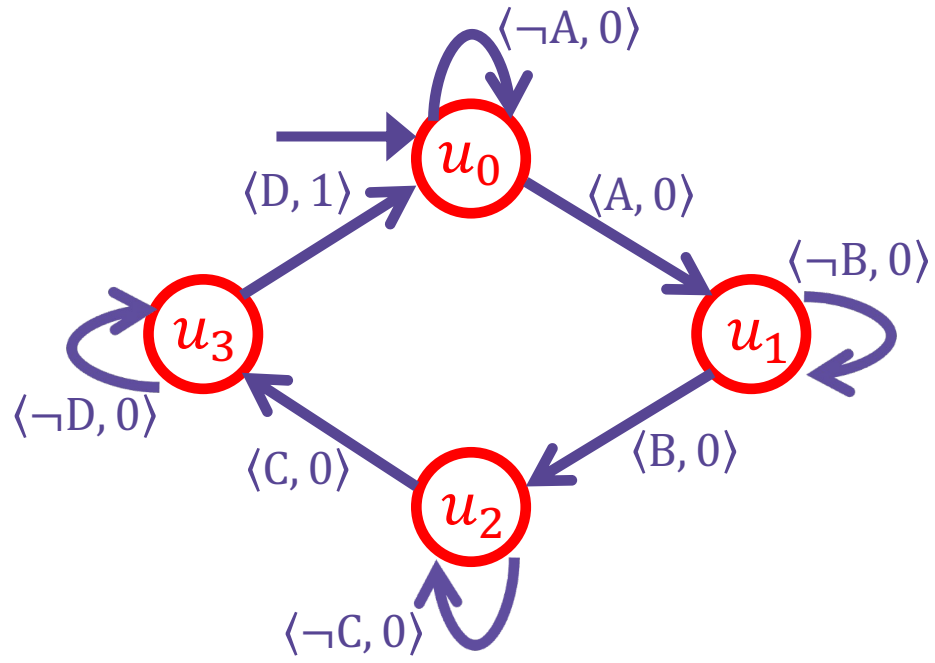


Encode reward function in an automata-like structure

using a **vocabulary** $P = \{\text{start}, \text{end}, o, *, A, B, C, D\}$

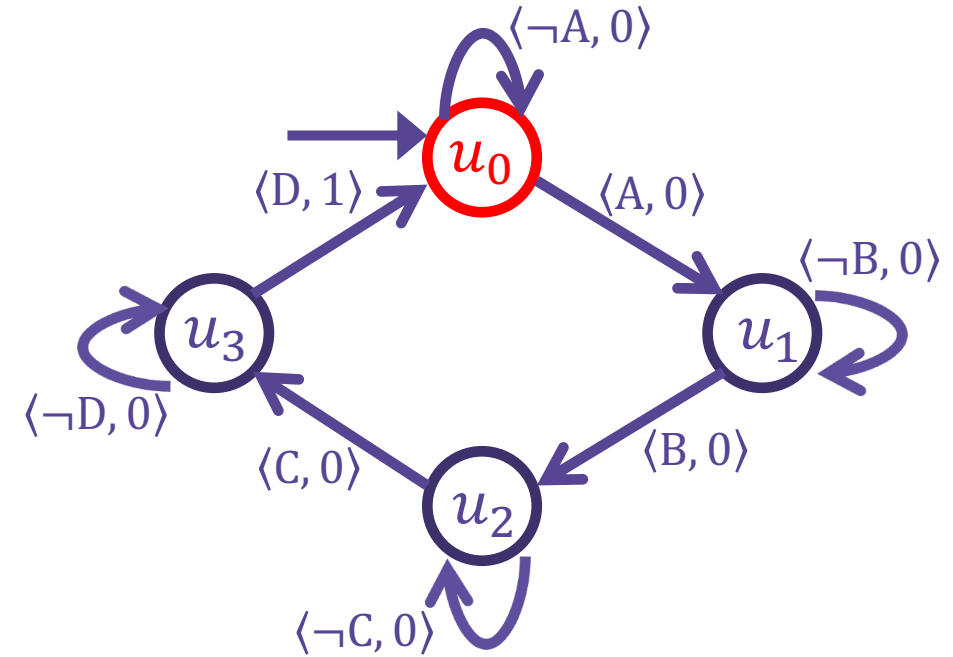
Reward Machine

- finite set of states U



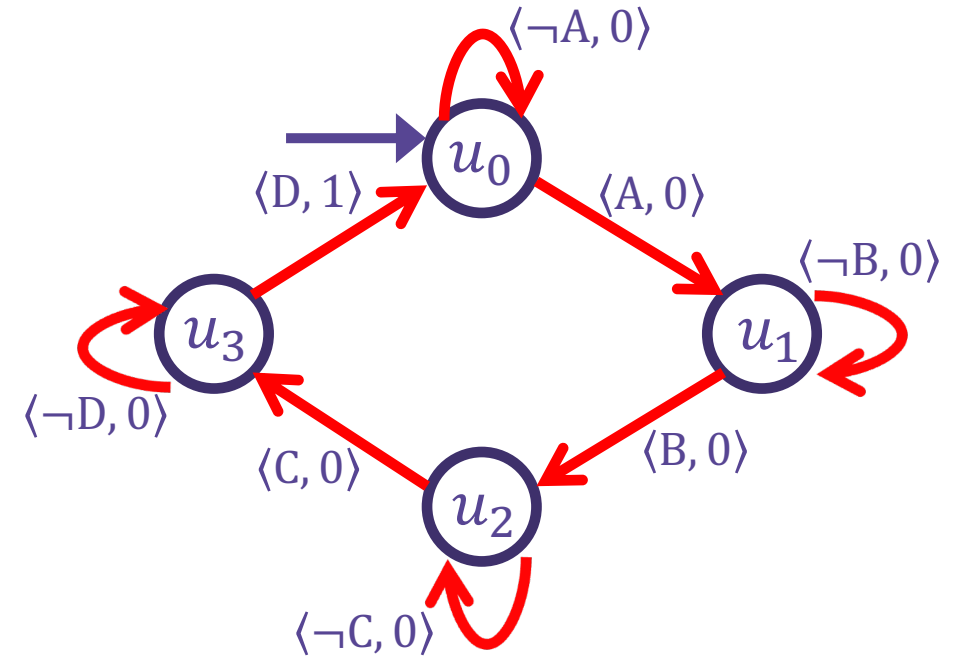
Reward Machine

- finite set of states U
- initial state $u_0 \in U$



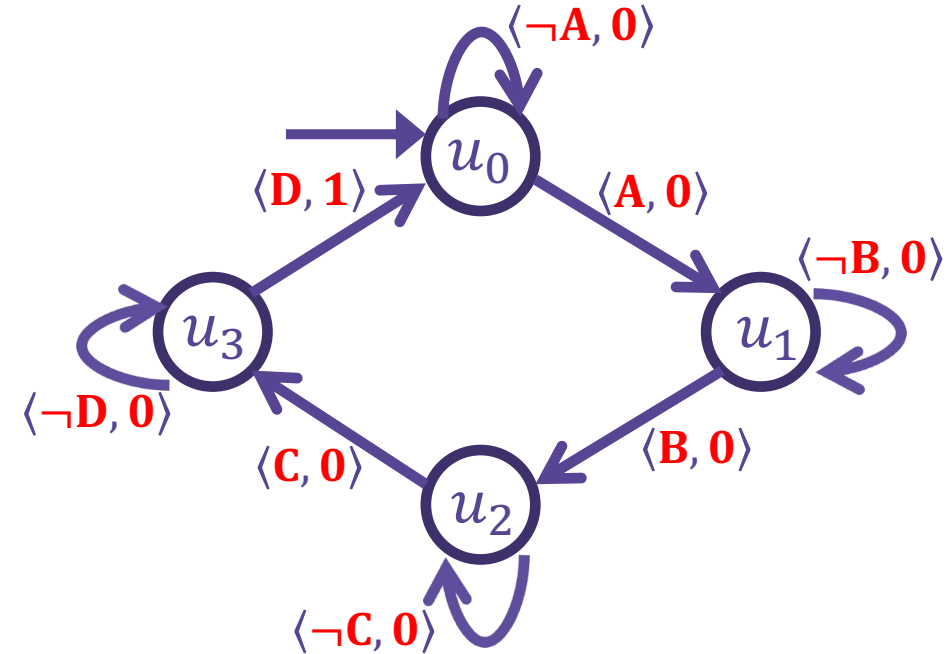
Reward Machine

- finite set of states U
- initial state $u_0 \in U$
- set of transitions labelled by:



Reward Machine

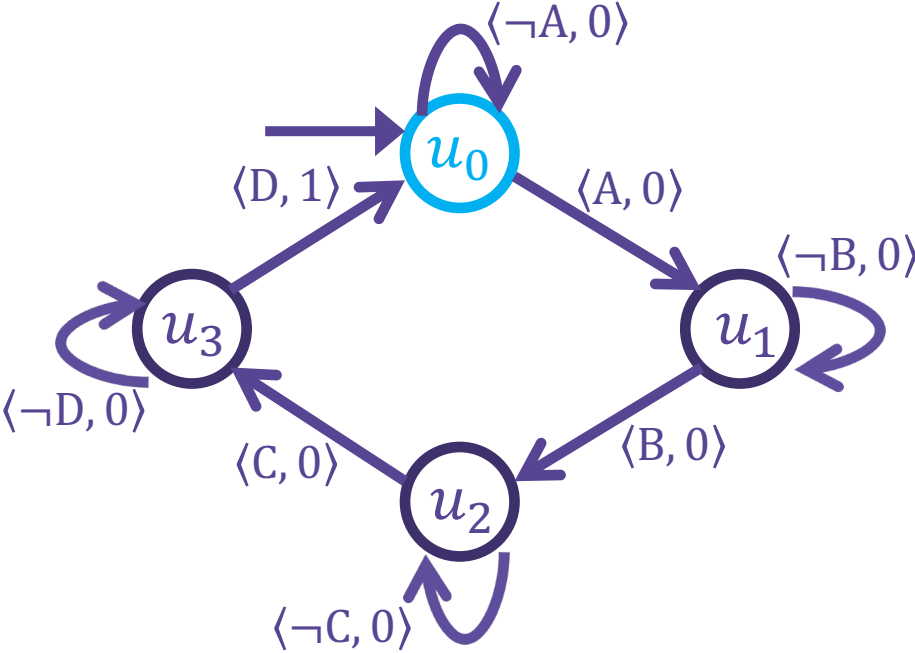
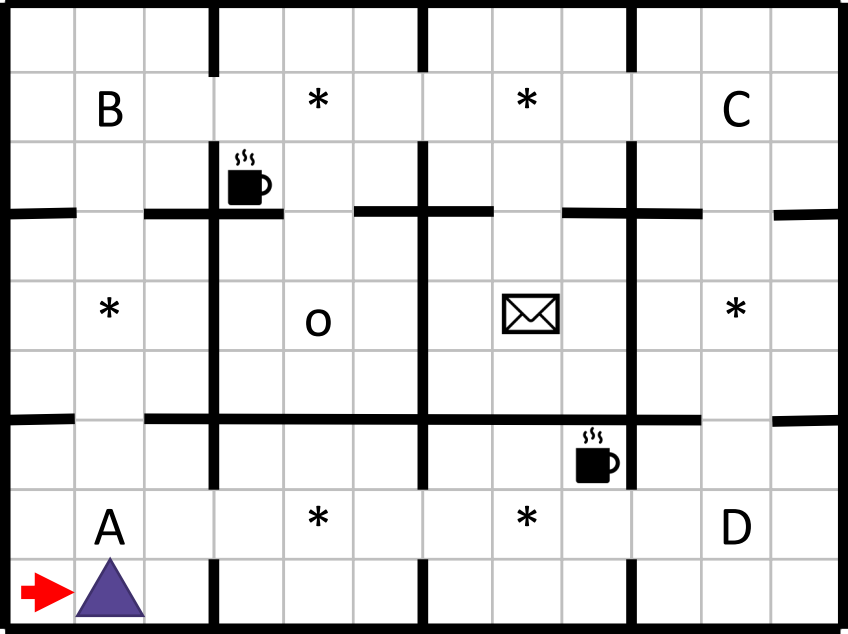
- finite set of states U
- initial state $u_0 \in U$
- set of transitions labelled by:
 - A logical condition* (guards)
 - A reward **function** (or **constant**)



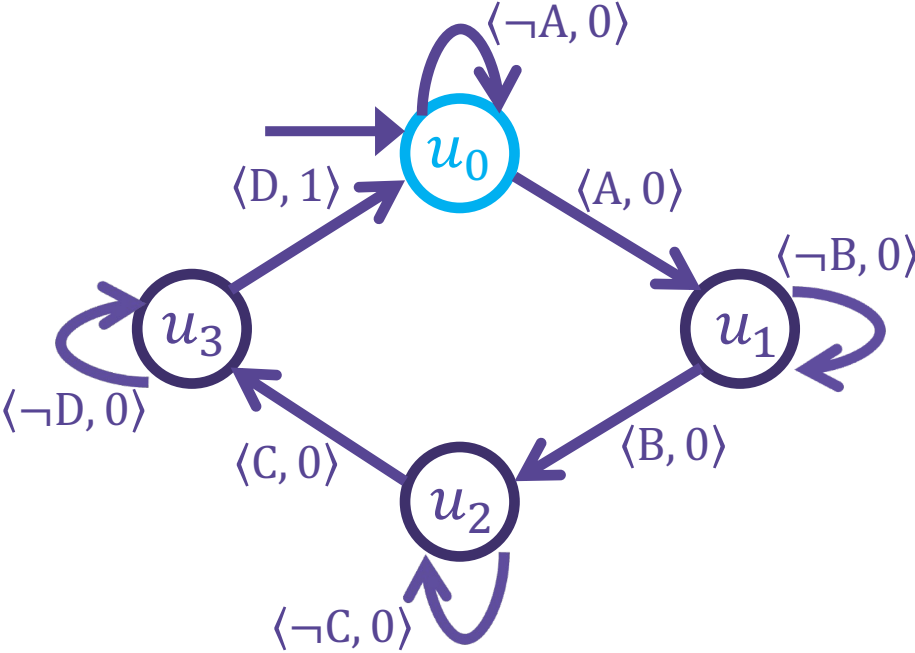
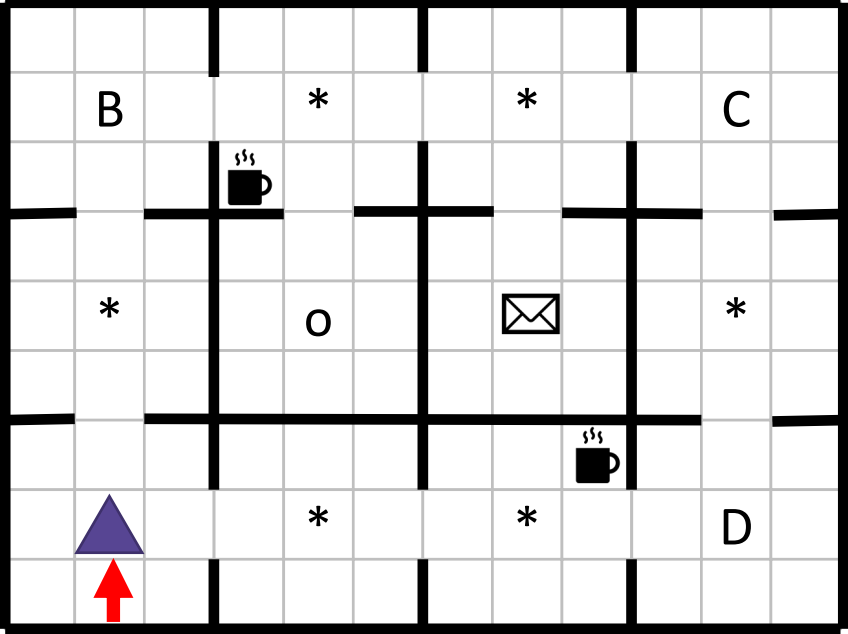
*logical formula --- property of the current state in the vocabulary P

$$P = \{ \text{☹}, \text{✉}, o, *, A, B, C, D \}$$

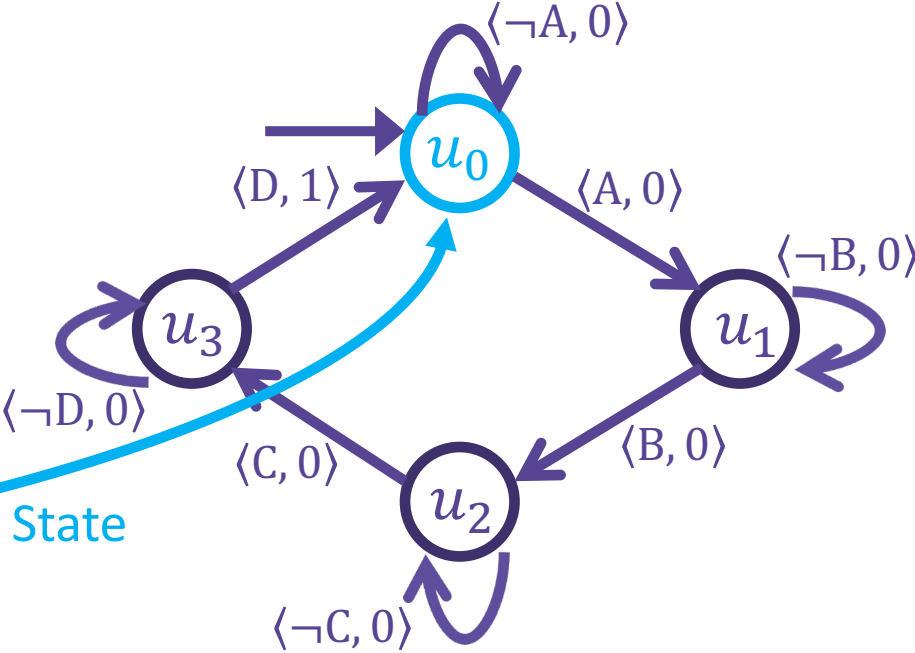
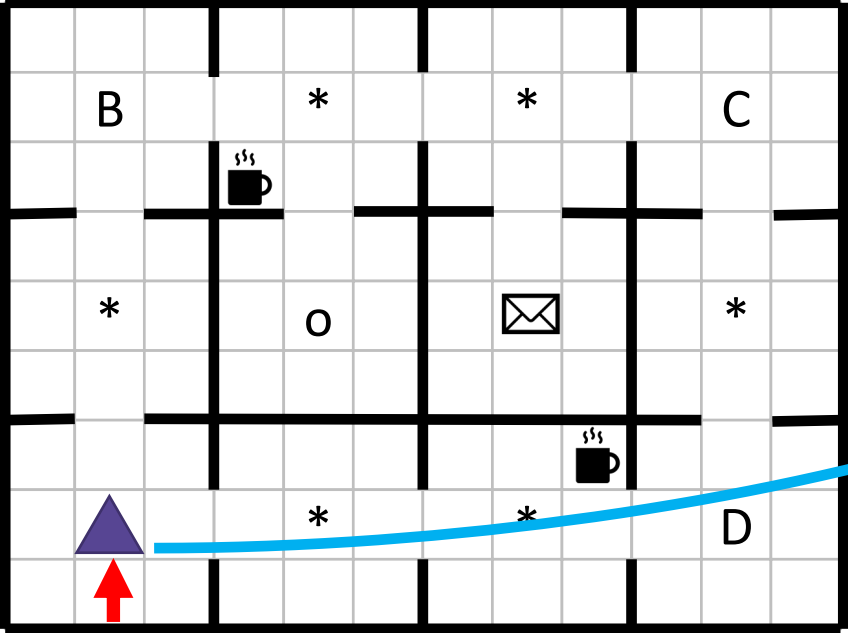
Reward Machines in Action



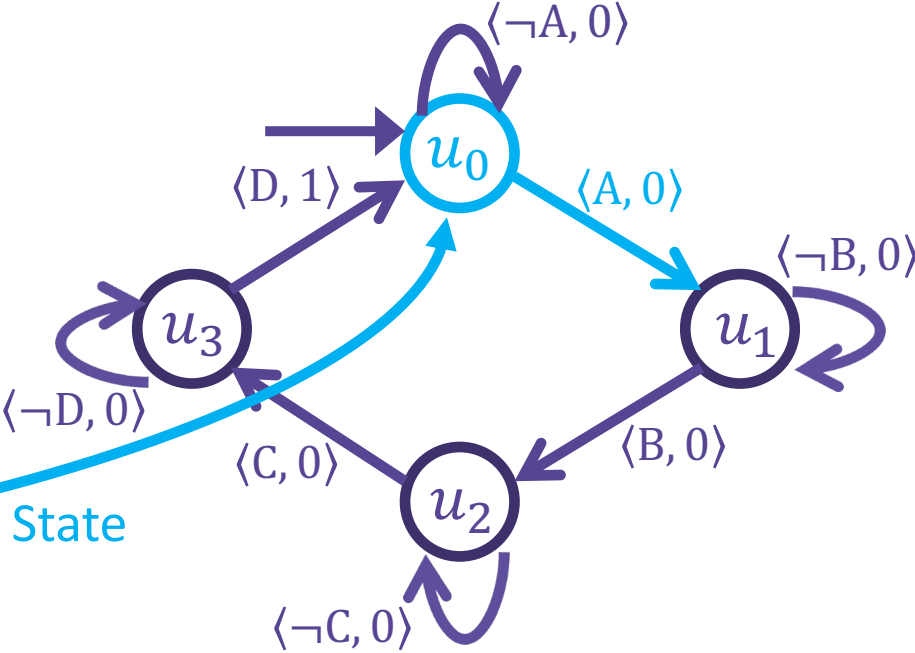
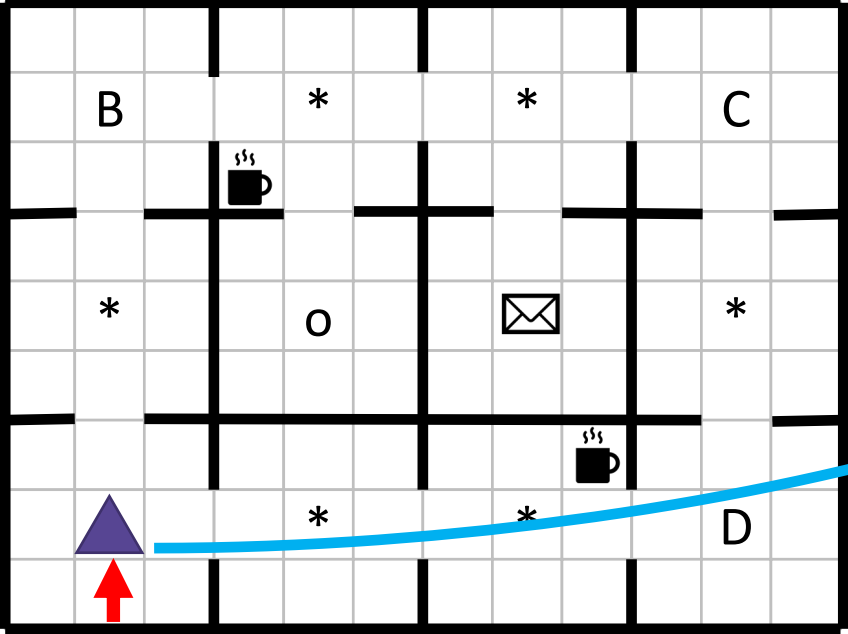
Reward Machines in Action



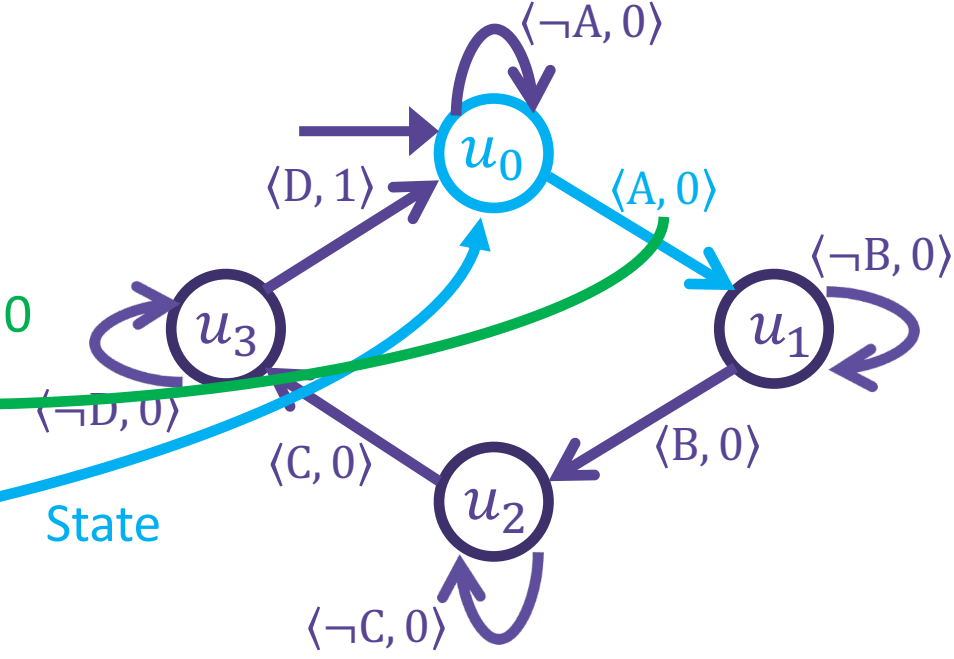
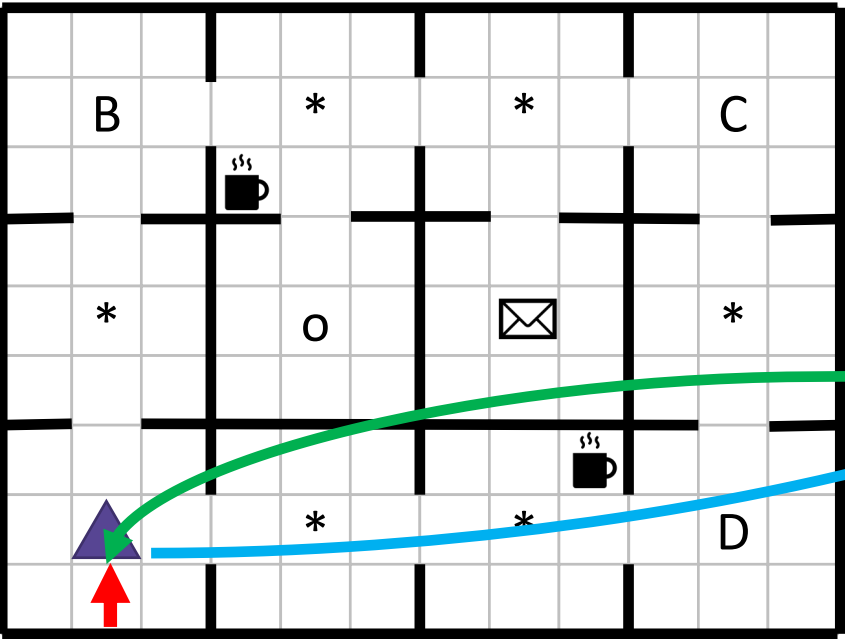
Reward Machines in Action



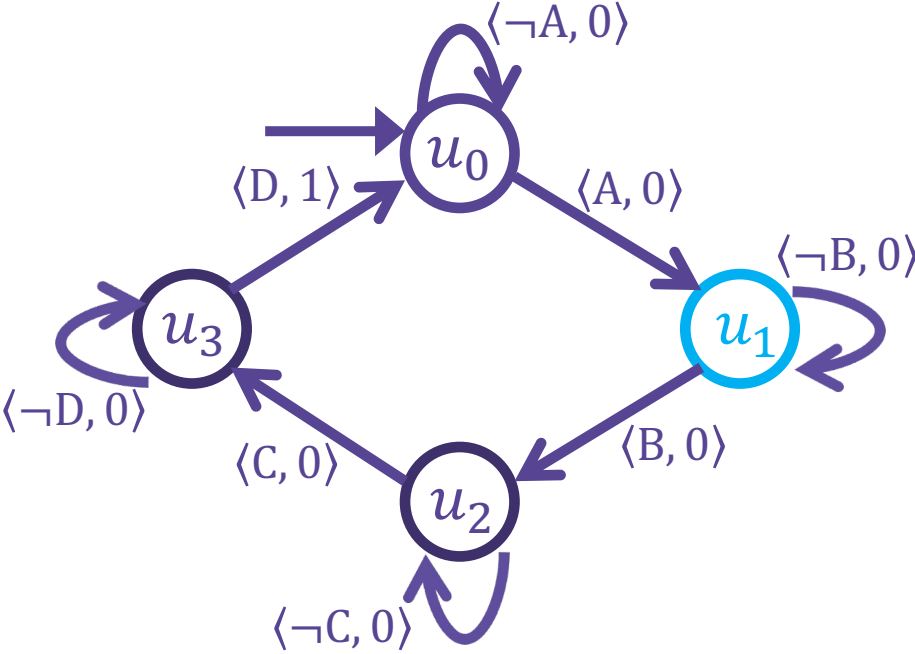
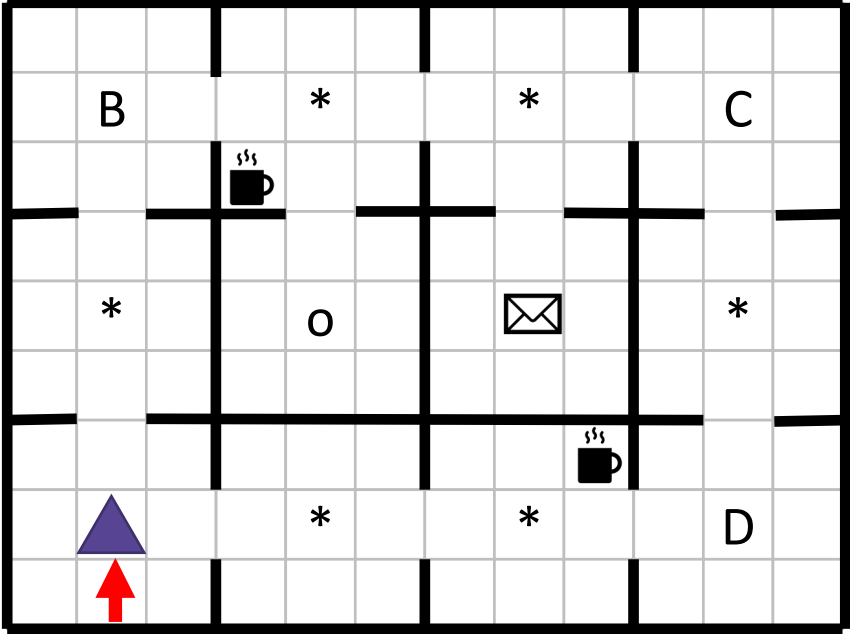
Reward Machines in Action



Reward Machines in Action

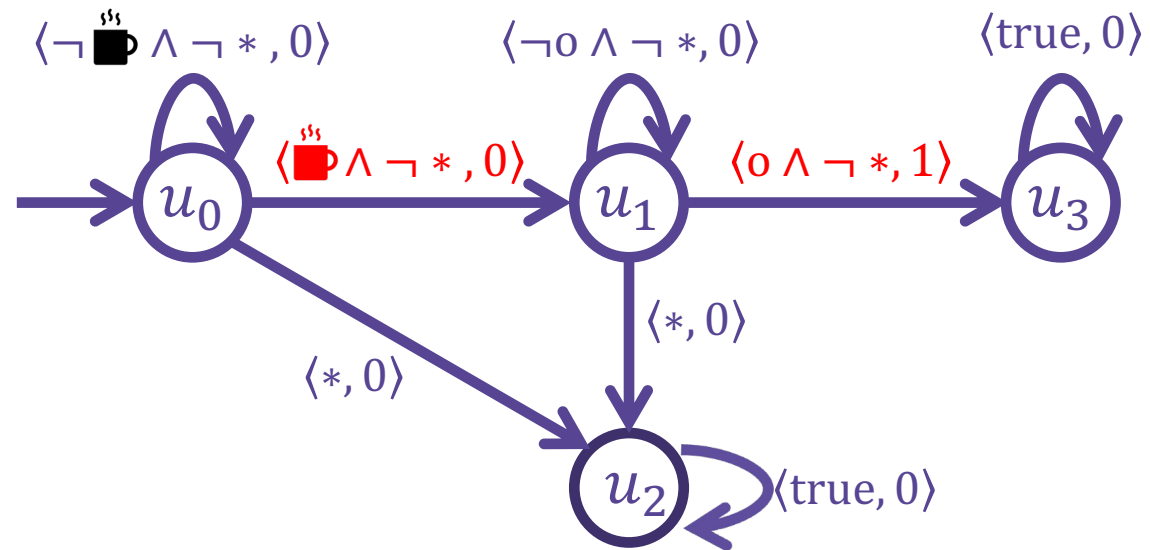


Reward Machines in Action



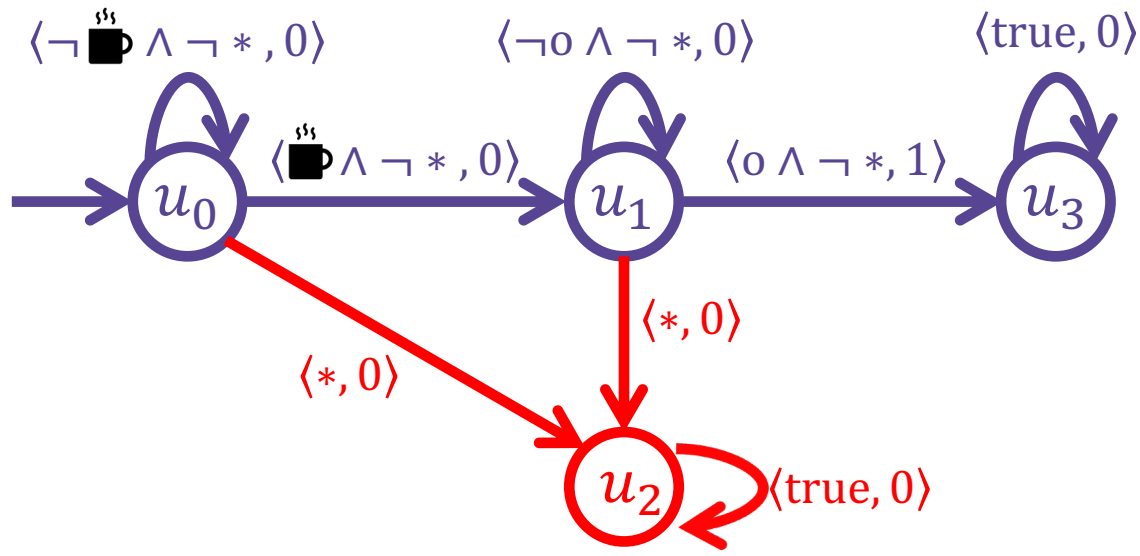
Other Reward Machines

Task: Deliver coffee to the office, while avoiding furniture.



Other Reward Machines

Task: Deliver coffee to the office, while avoiding furniture.



The Rest of the Talk

- **Reward Machines (RM)**
- ▶ **Exploiting RM Structure in Learning**
- **Experiments**
- **Creating Reward Machines**
- **Concluding Remarks**



EXPLOITING RM STRUCTURE IN LEARNING

Methods for Exploiting RM Structure

Our Approaches:

1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
2. Hierarchical RL based on options (HRL) **Baseline**
3. HRL with RM-based pruning (HRL-RM)
4. Q-learning for Reward Machines (QRM)
5. QRM + Reward Shaping for Reward Machine (QRM + RS)
6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
7. CRM+ Reward Shaping for Reward Machine (CRM + RS)

[Toro Icarte, Klassen, Valenzano, M., ICML18]

[Toro Icarte, Klassen, Valenzano, M., JAIR22]

Simple Idea:

- Give the **learning algorithm access to the reward function**
- ...
- ...

Simple Idea:

- Give the **learning algorithm access to the reward function**
- Exploit **compositional reward function structure** in learning
- ...

Methods for Exploiting RM Structure

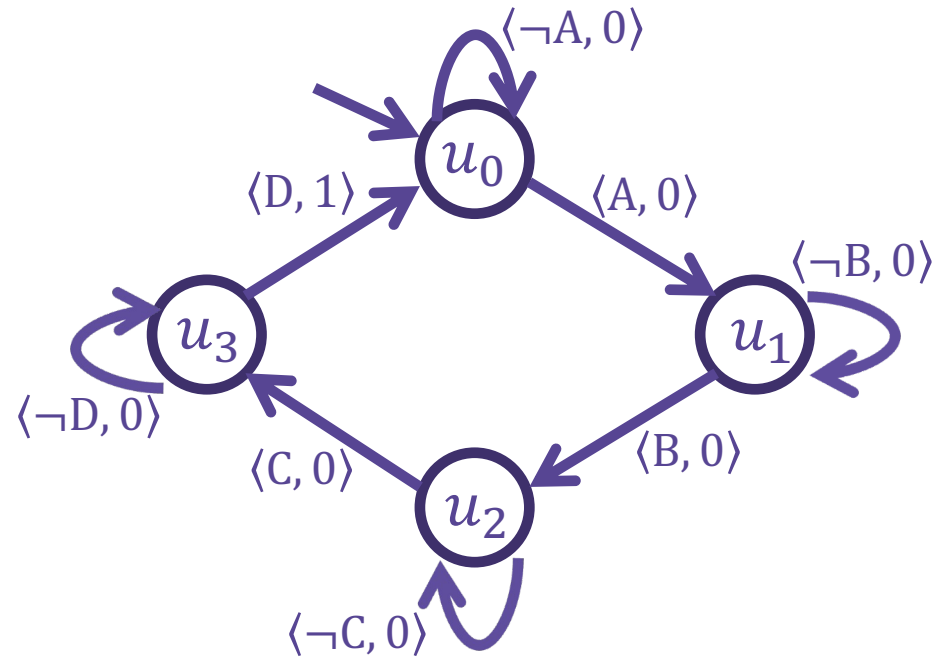
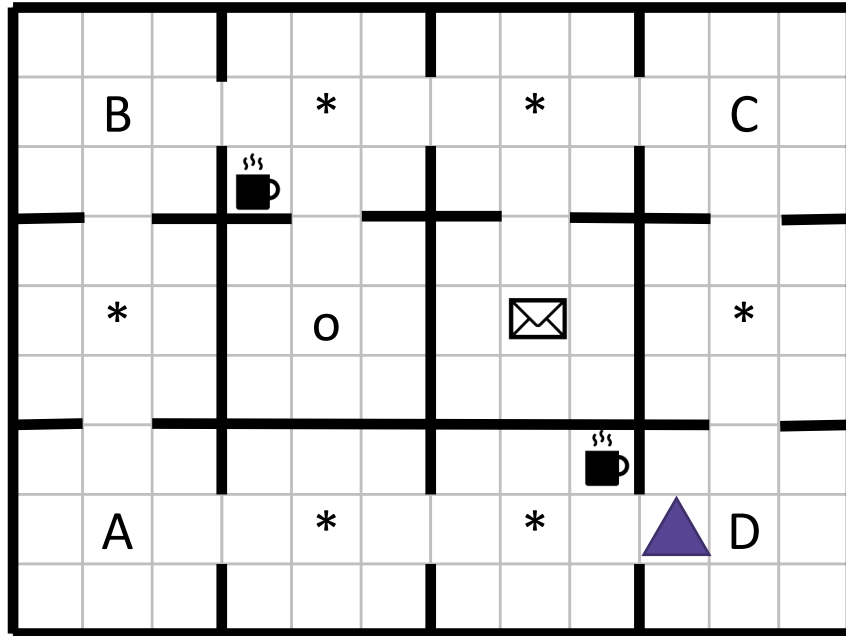
Our Approaches:

- ➔ 1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
- 2. Hierarchical RL based on options (HRL) **Baseline**
- 3. HRL with RM-based pruning (HRL-RM)
- 4. Q-learning for Reward Machines (QRM)
- 5. QRM + Reward Shaping for Reward Machine (QRM + RS)
- 6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
- 7. CRM+ Reward Shaping for Reward Machine (CRM + RS)

[Toro Icarte, Klassen, Valenzano, M., ICML18]

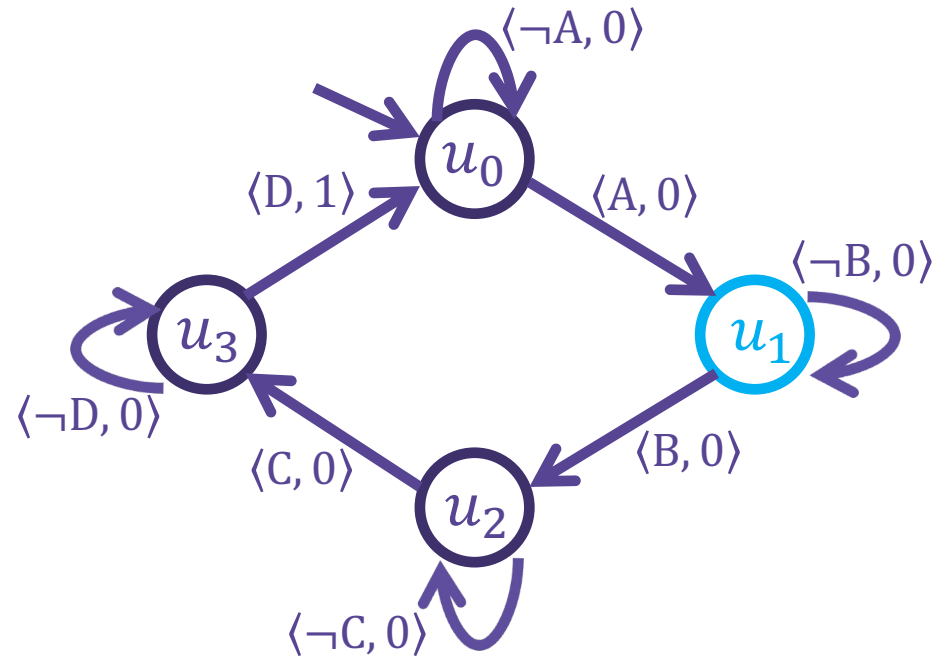
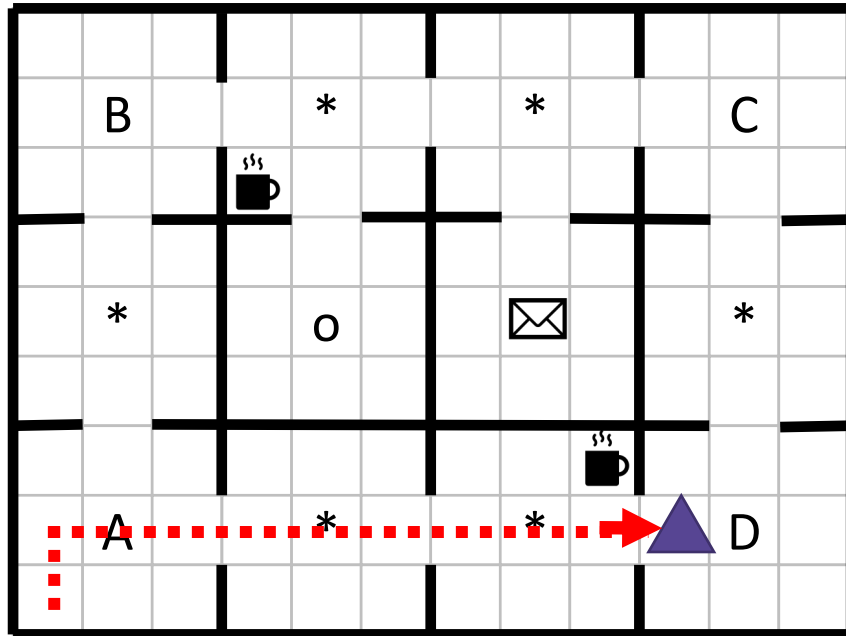
[Toro Icarte, Klassen, Valenzano, M., JAIR22]

1. Q-Learning over Cross-Product - Baseline



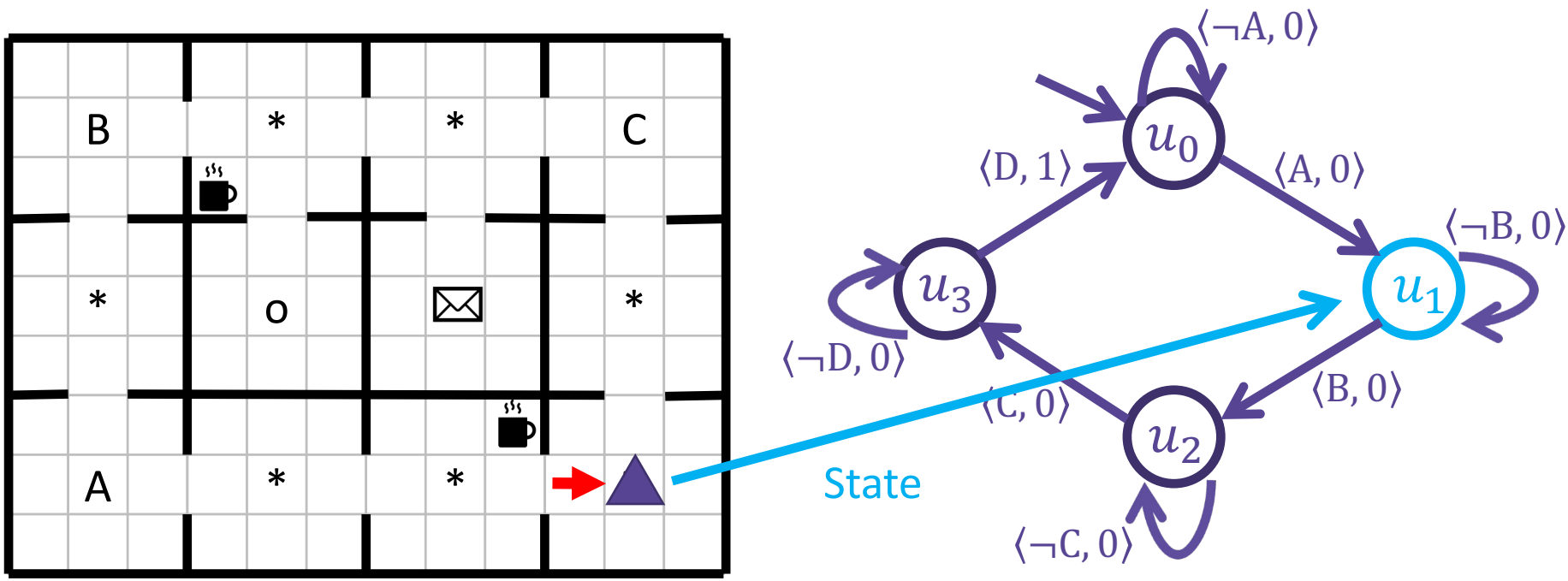
A Reward Machine may define a **non-Markovian** reward function.

1. Q-Learning over Cross-Product - Baseline



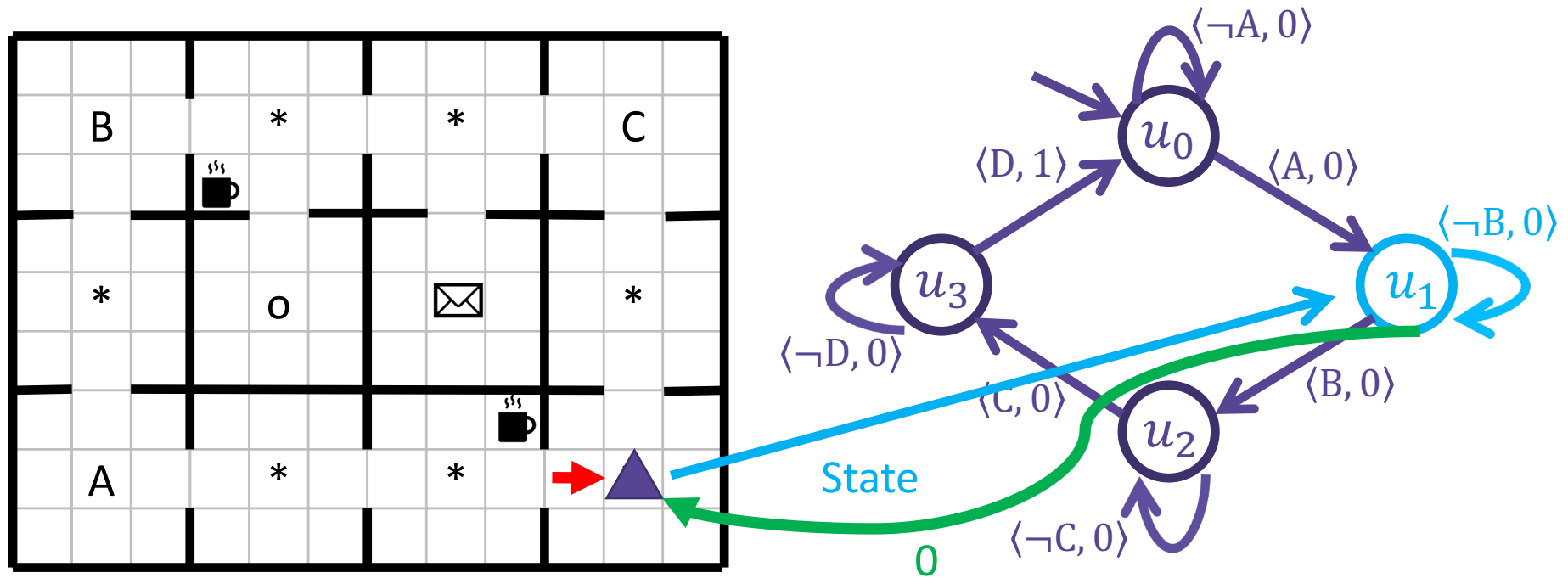
A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



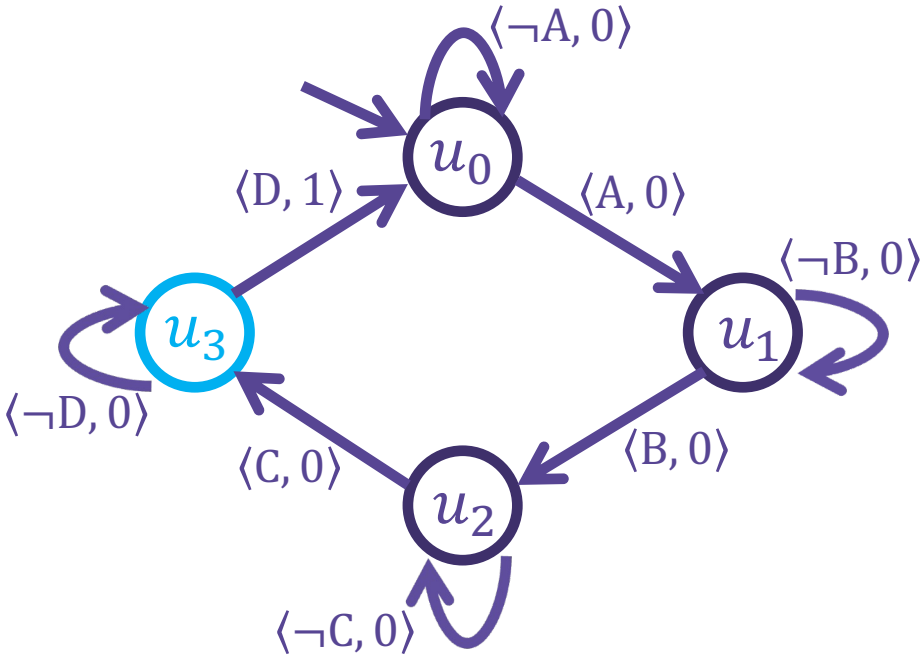
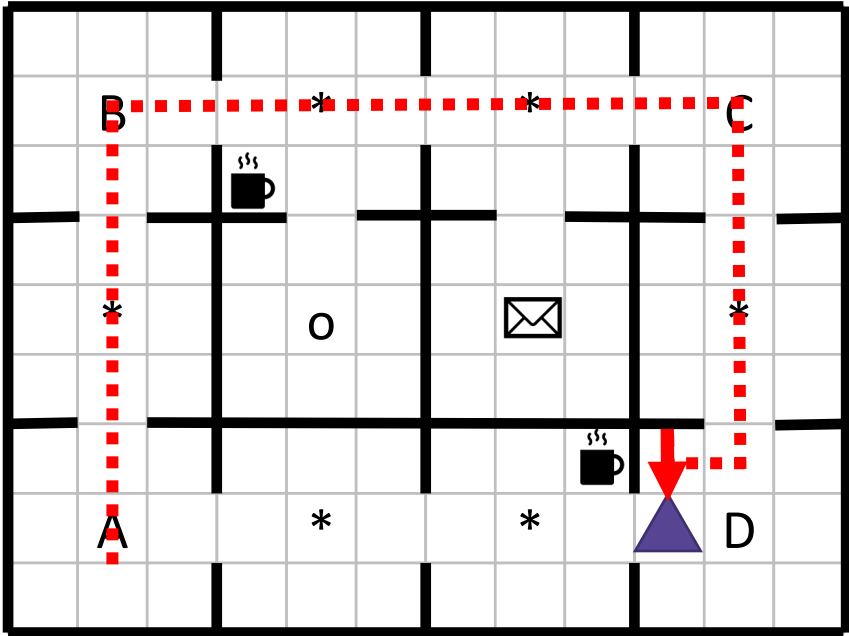
A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



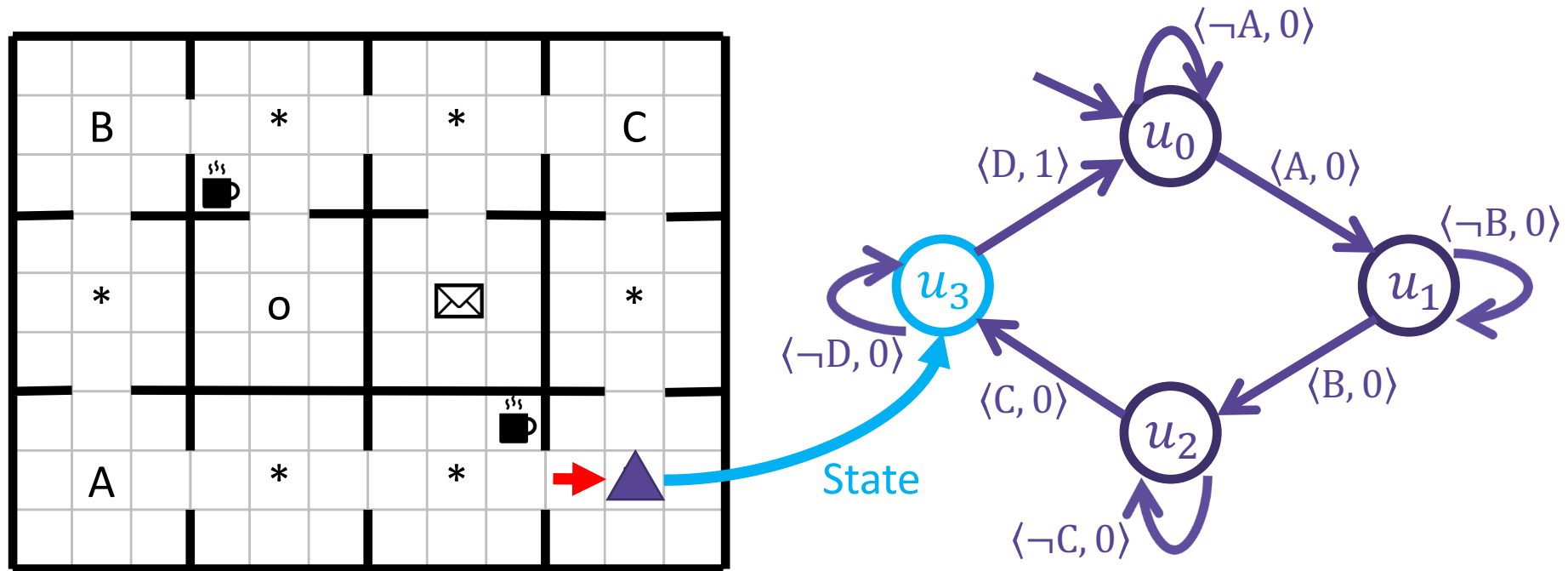
A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



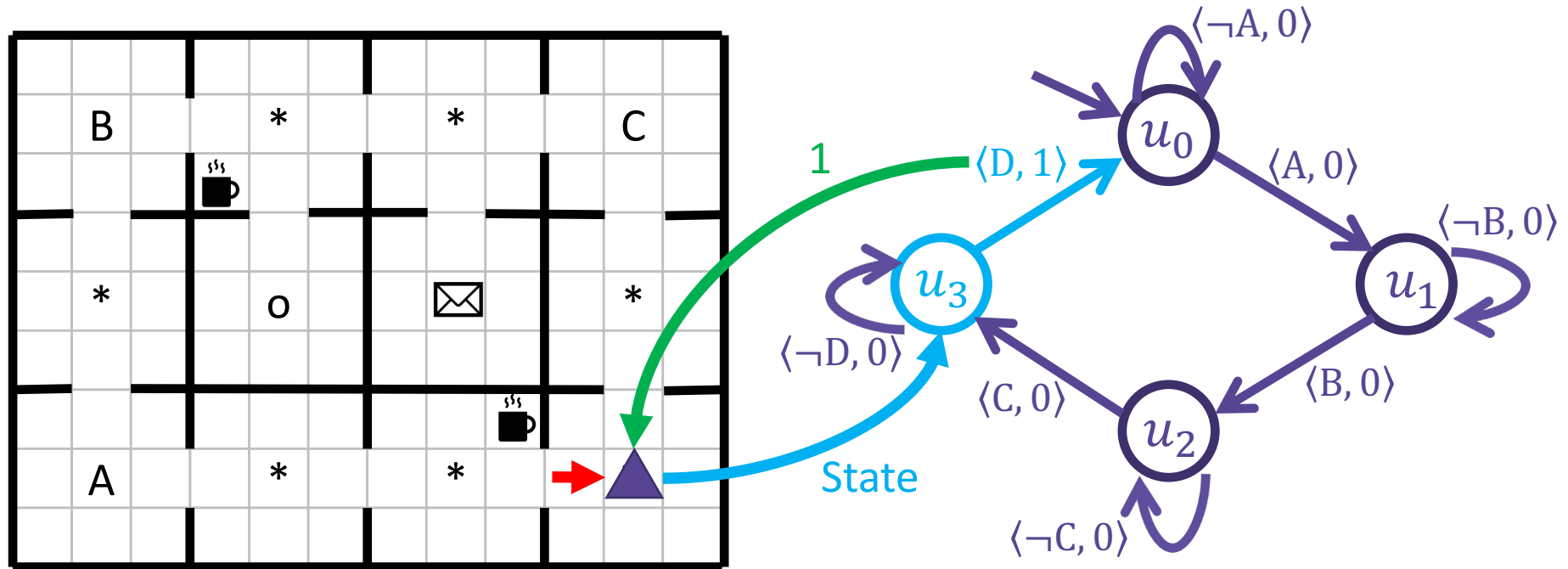
A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



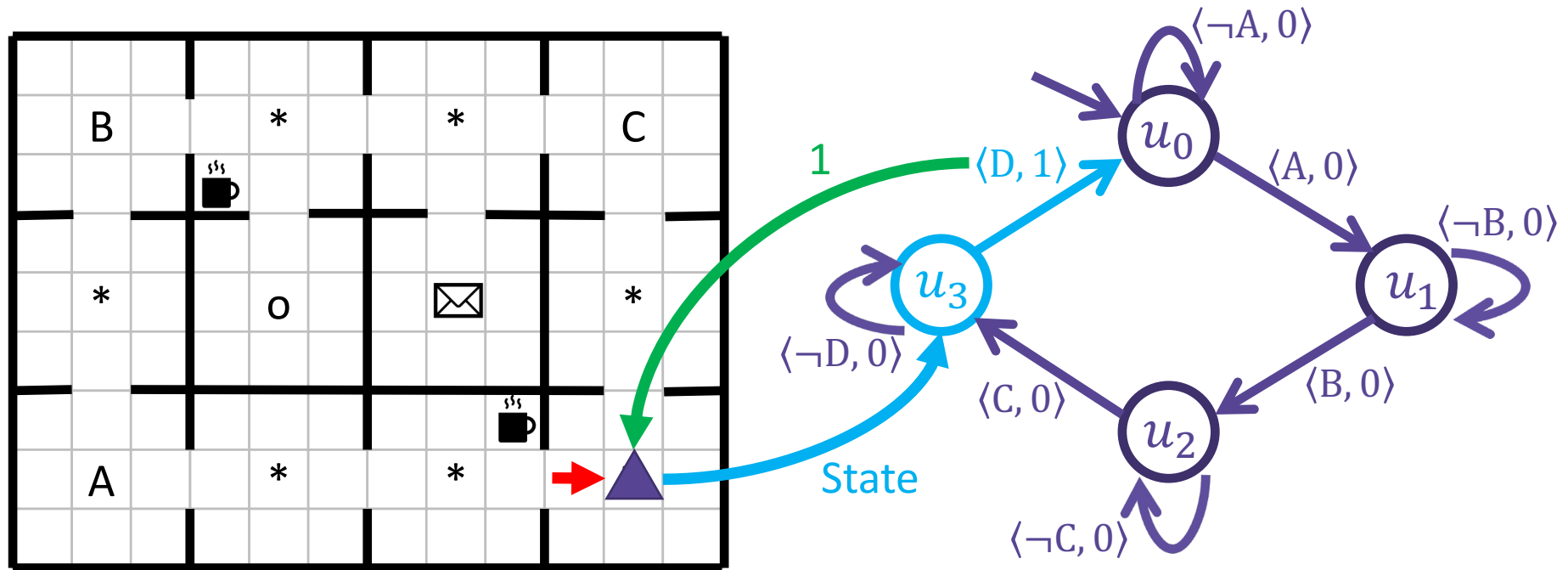
A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



A Reward Machine may define a non-Markovian reward function.

1. Q-Learning over Cross-Product - Baseline



Solution: Include **RM state as part of agent's state** representation.

Use standard Q-learning on resulting MDP.

Methods for Exploiting RM Structure


Our Approaches:

1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
2. Hierarchical RL based on options **(HRL) Baseline**
3. HRL with RM-based pruning **(HRL-RM)**
4. Q-learning for Reward Machines (QRM)
5. QRM + Reward Shaping for Reward Machine (QRM + RS)
6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
7. CRM+ Reward Shaping for Reward Machine (CRM + RS)



Methods for Exploiting RM Structure

Our Approaches:

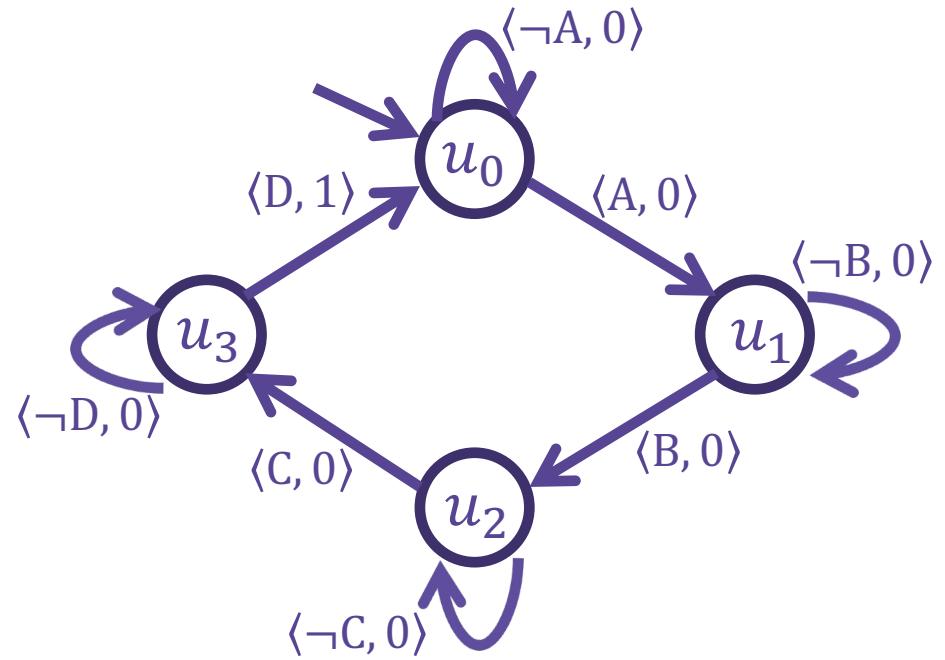
1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
2. Hierarchical RL based on options (HRL) **Baseline**
3. HRL with RM-based pruning (HRL-RM)
-  4. Q-learning for Reward Machines (QRM)
5. QRM + Reward Shaping for Reward Machine (QRM + RS)
6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
7. CRM+ Reward Shaping for Reward Machine (CRM + RS)



Simple Idea:

- Give the **learning algorithm access to the reward function**
- ➔ • Exploit **compositional reward function structure** in learning
- ➔ • Employ **Counterfactual Reasoning**

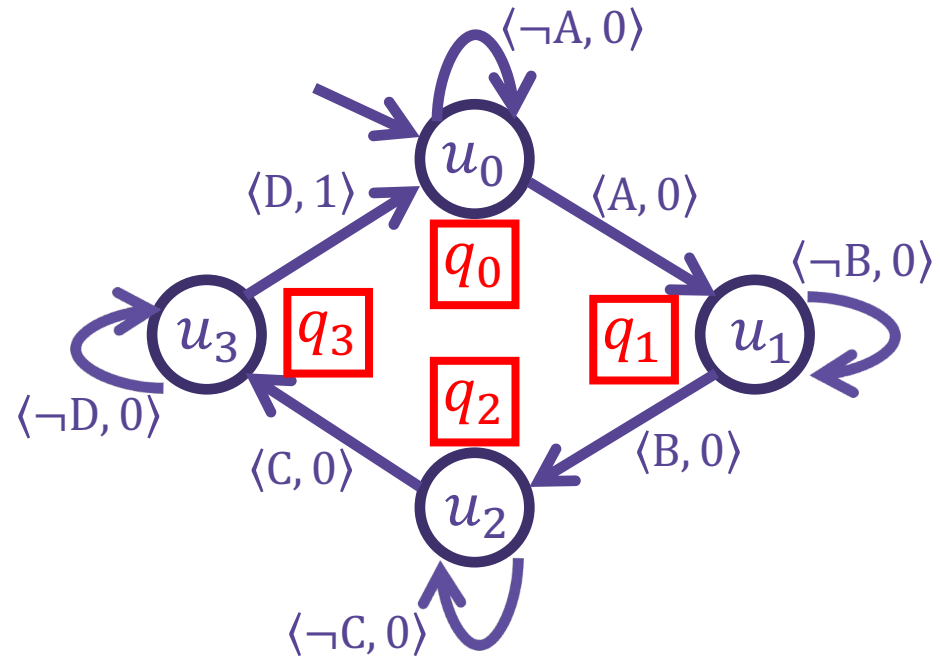
4. Q-Learning for Reward Machines (QRM)



4. Q-Learning for Reward Machines (QRM)

QRM (our approach)

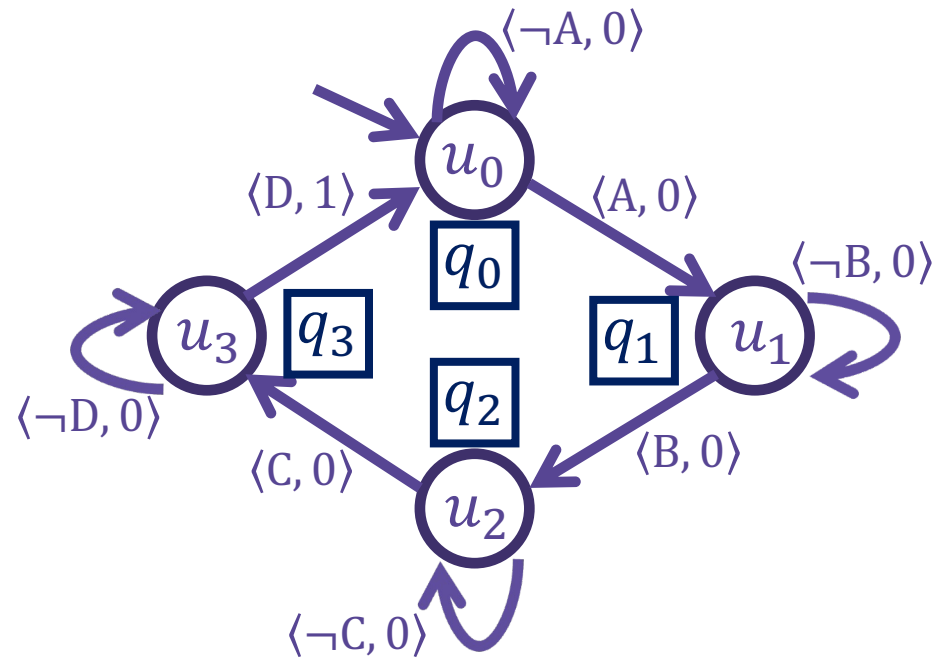
1. Learn one policy (Q-value function) per state in the Reward Machine.



4. Q-Learning for Reward Machines (QRM)

QRM (our approach)

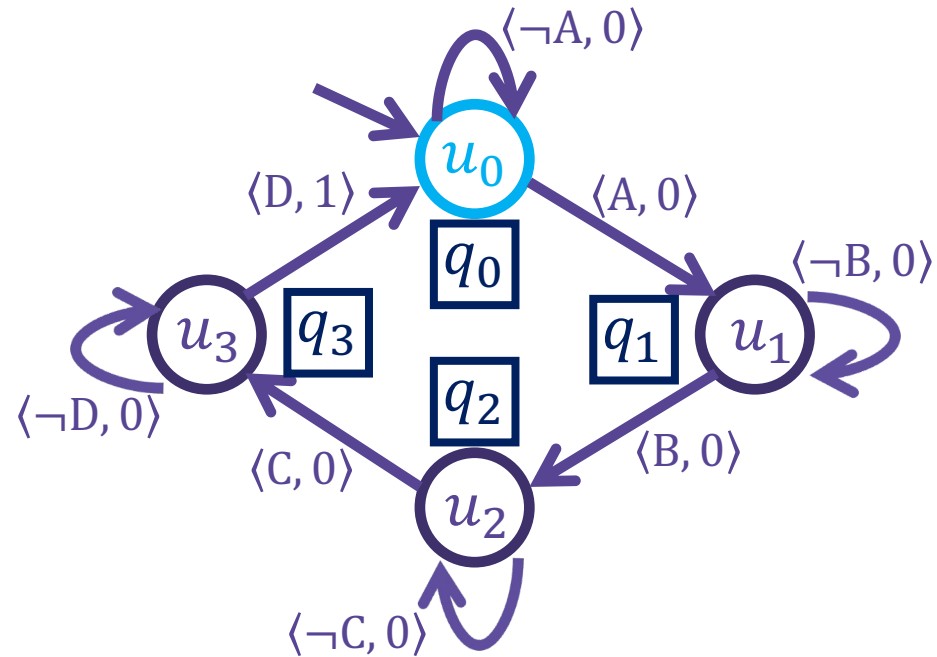
1. Learn one policy (Q-value function) per state in the Reward Machine.
2. Select actions using the policy of the current RM state.



4. Q-Learning for Reward Machines (QRM)

QRM (our approach)

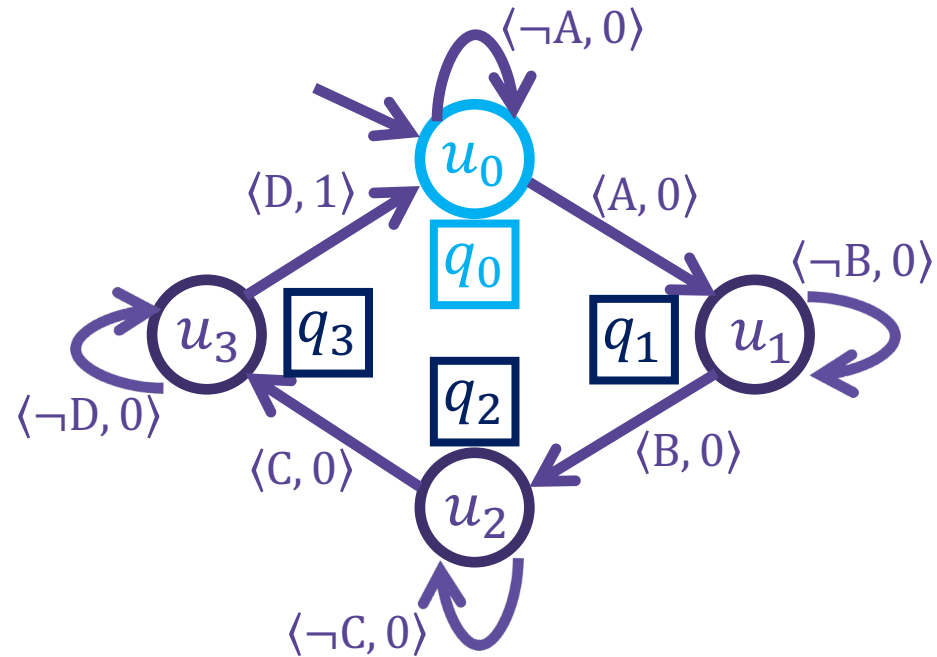
1. Learn one policy (Q-value function) per state in the Reward Machine.
2. Select actions using the policy of the current RM state.



4. Q-Learning for Reward Machines (QRM)

QRM (our approach)

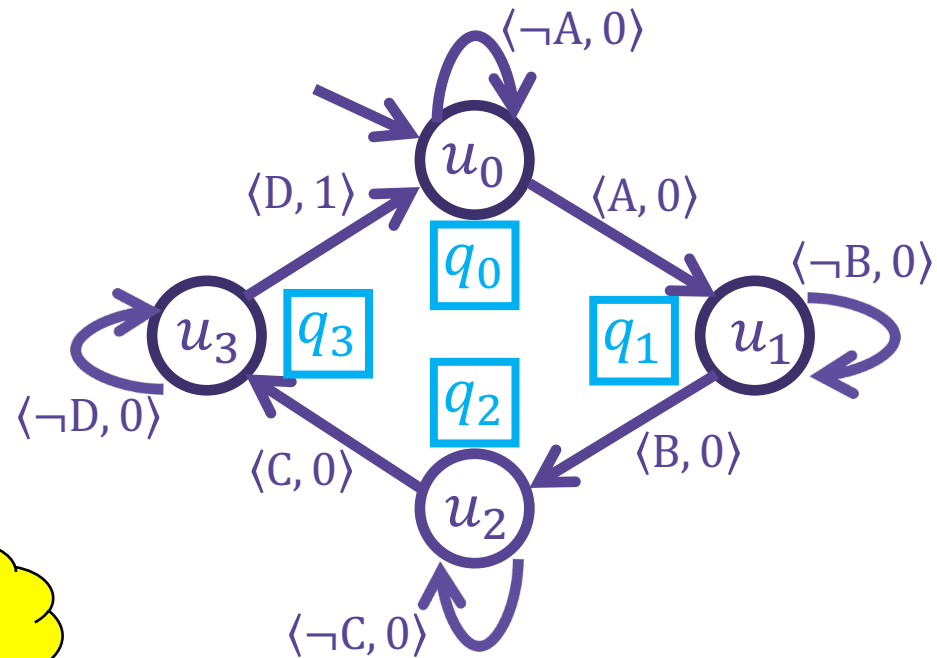
1. Learn one policy (Q-value function) per state in the Reward Machine.
2. Select actions using the policy of the current RM state.



4. Q-Learning for Reward Machines (QRM)

QRM (our approach)


1. Learn one policy (Q-value function) per state in the Reward Machine.
2. Select actions using the policy of the current RM state.
3. Reuse experience to update all Q-value functions on every transition via off-policy reinforcement learning.



This is a form of **Counterfactual Reasoning**


Methods for Exploiting RM Structure

Our Approaches:

1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
2. Hierarchical RL based on options (HRL) **Baseline**
3. HRL with RM-based pruning (HRL-RM)
4. Q-learning for Reward Machines (QRM)
-  5. QRM + Reward Shaping for Reward Machine (QRM + RS)
6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
7. CRM+ Reward Shaping for Reward Machine (CRM + RS)

Methods for Exploiting RM Structure

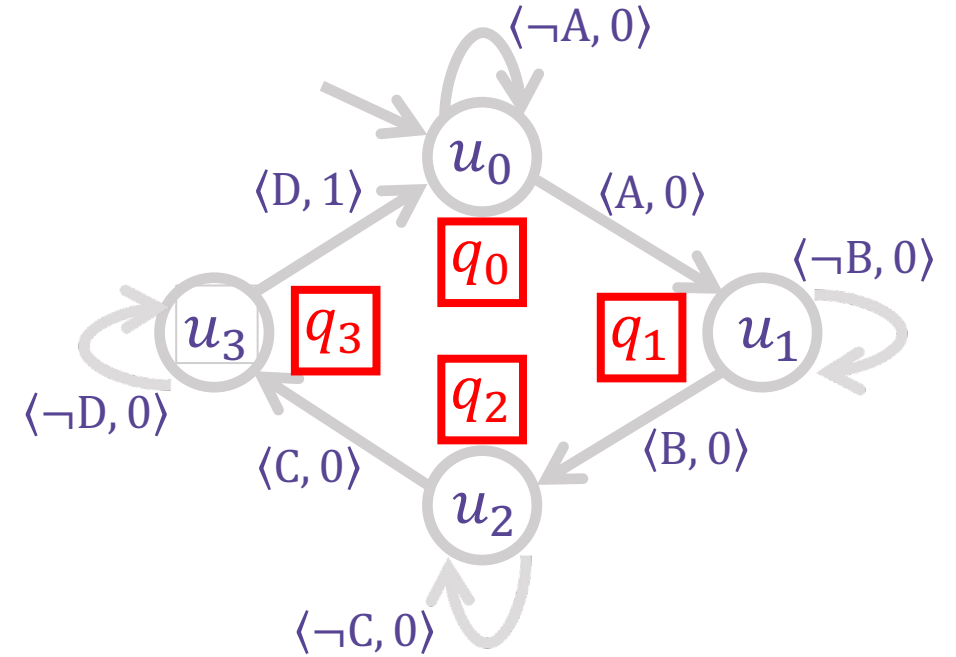
Our Approaches:

1. Q-learning over Cross-Product State + RM (Q-learning) **Baseline**
2. Hierarchical RL based on options (HRL) **Baseline**
3. HRL with RM-based pruning (HRL-RM)
4. Q-learning for Reward Machines (QRM)
5. QRM + Reward Shaping for Reward Machine (QRM + RS)
-  6. Q-learning with Counterfactual Experiences for Reward Machines (CRM)
7. CRM+ Reward Shaping for Reward Machine (CRM + RS)

6. Q-Learning w/ Counterfactual Experiences for RMs (CRM)

QRM

- learns **multiple Q-functions**
- one Q-function $q(s,a)$ per reward machine state



6. Q-Learning w/ Counterfactual Experiences for RMs (CRM)

QRM

- learns **multiple Q-functions**
- one Q-function $q(s,a)$ per reward machine state

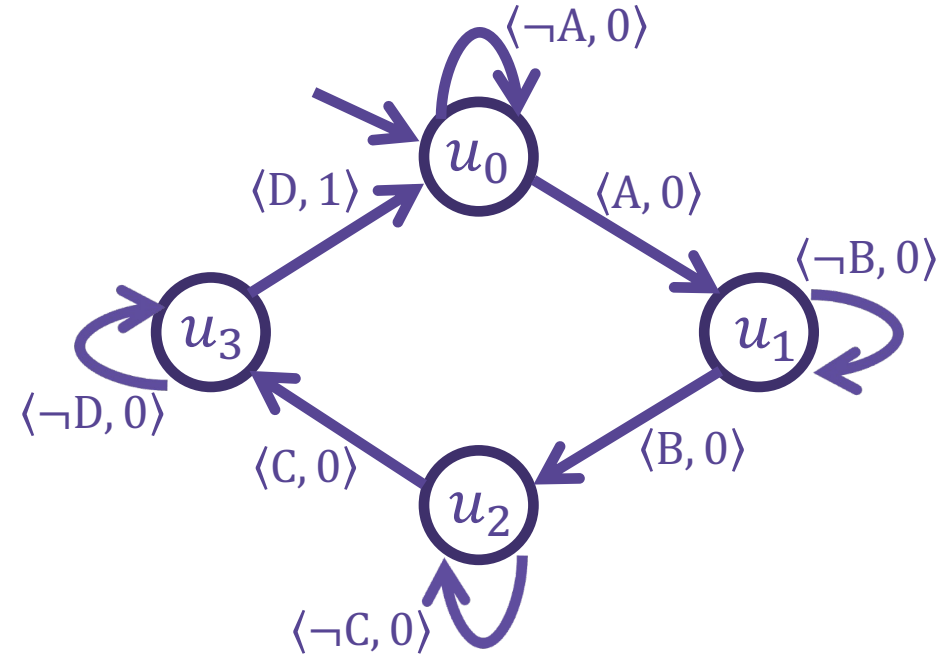
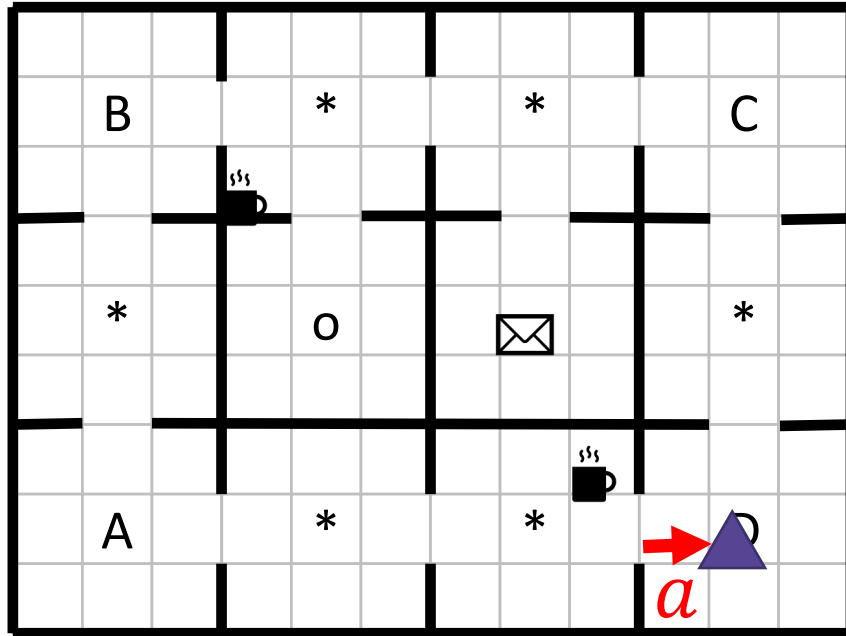
CRM

- learns **one Q-function** $q(s,u,a)$ for the cross-product state $\langle s,u \rangle$
- synthesizing **counterfactual experiences** for all the other RM states $\langle s,u_i \rangle$

CRM behaves identically to QRM in tabular domains

[Toro Icarte, Klassen, Valenzano, M., JAIR22]

Optimality of QRM and CRM (with reward shaping)

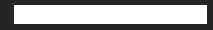


Theorem (informally stated):

QRM & CRM (+RS) preserve optimality guarantees of Q-learning & reward shaping. They converge to optimal policies when there is no function approximation.

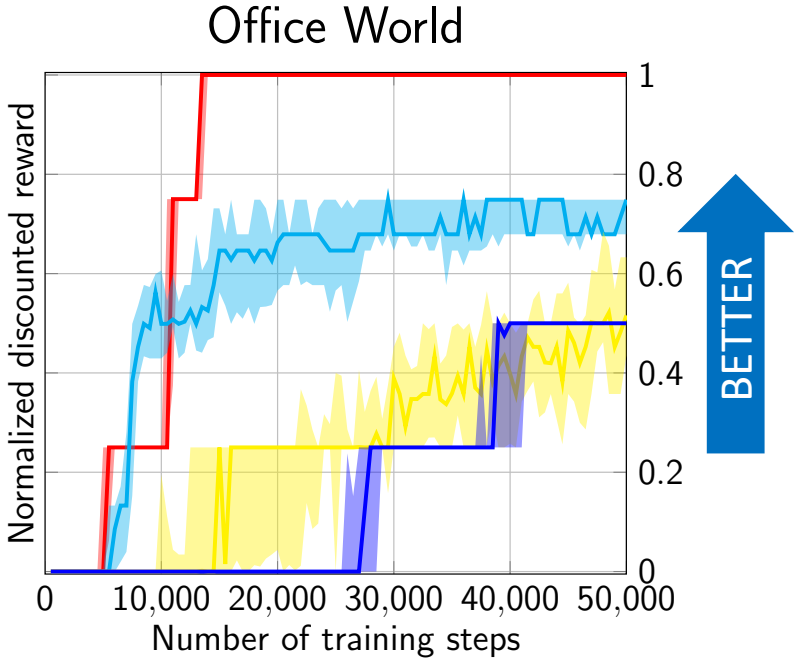
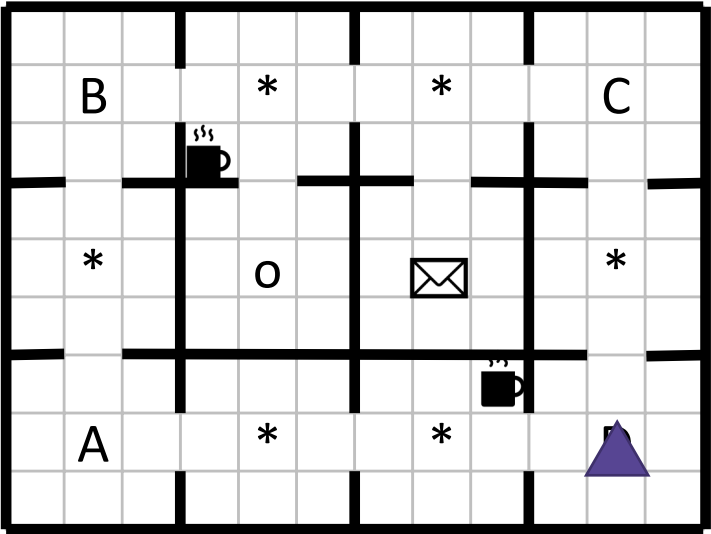
The Rest of the Talk

- **Reward Machines (RM)**
- **Exploiting RM Structure in Learning**
- ▶ **Experiments**
- **Creating Reward Machines**
- **Concluding Remarks**



EXPERIMENTS

Office World Experiments

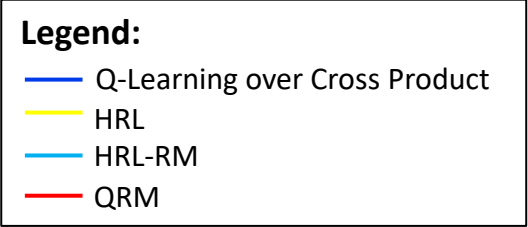
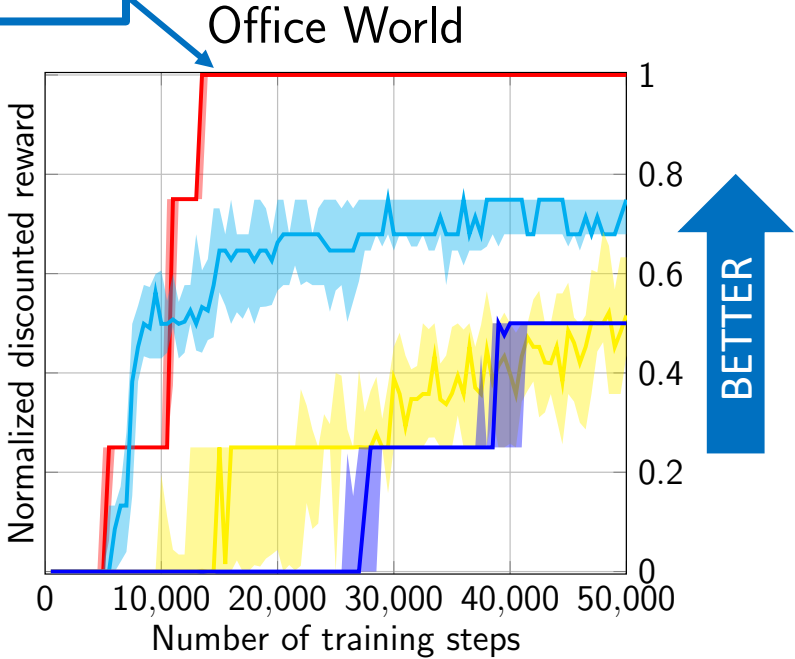
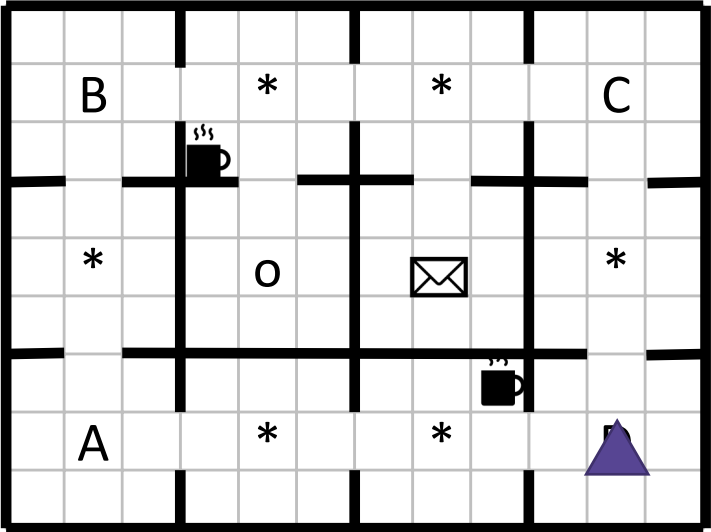


4 tasks, 30 independent trials per task

[Toro Icarte, Klassen, Valenzano, M., ICML18]

Office World Experiments

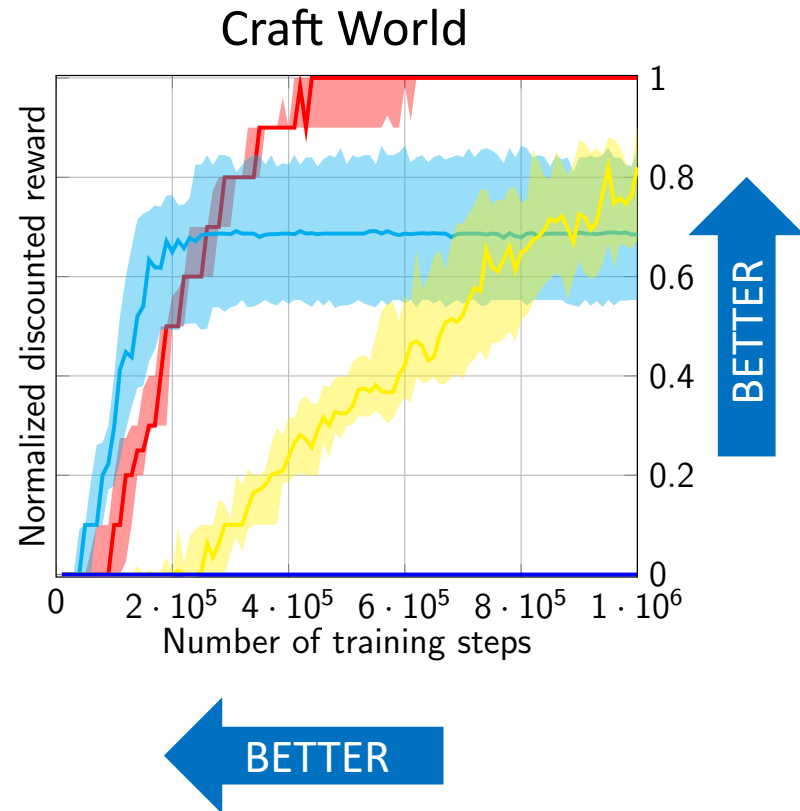
(Hugely) More Sample Efficiency (x axis)
Greater Reward (y-axis)



4 tasks, 30 independent trials per task

[Toro Icarte, Klassen, Valenzano, M., ICML18]

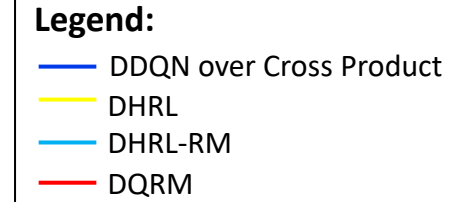
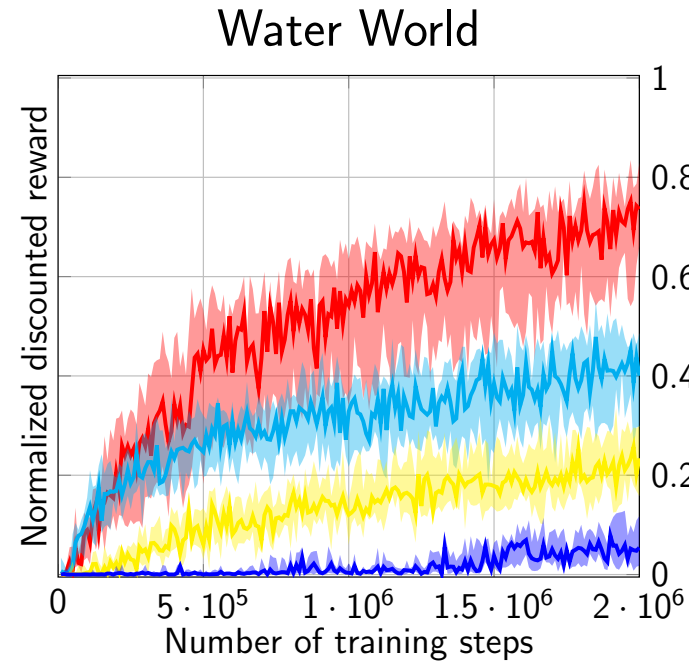
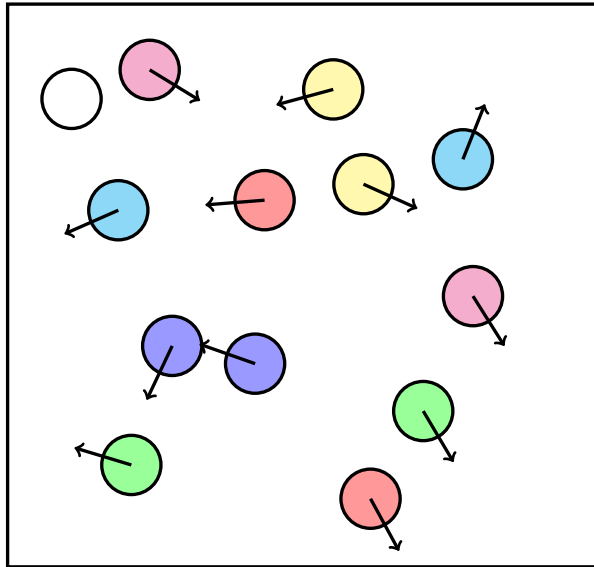
Craft World Experiments



10 tasks over 10 random maps, 3 independent trials per combination

[Toro Icarte, Klassen, Valenzano, M., ICML18]

Water World Experiments



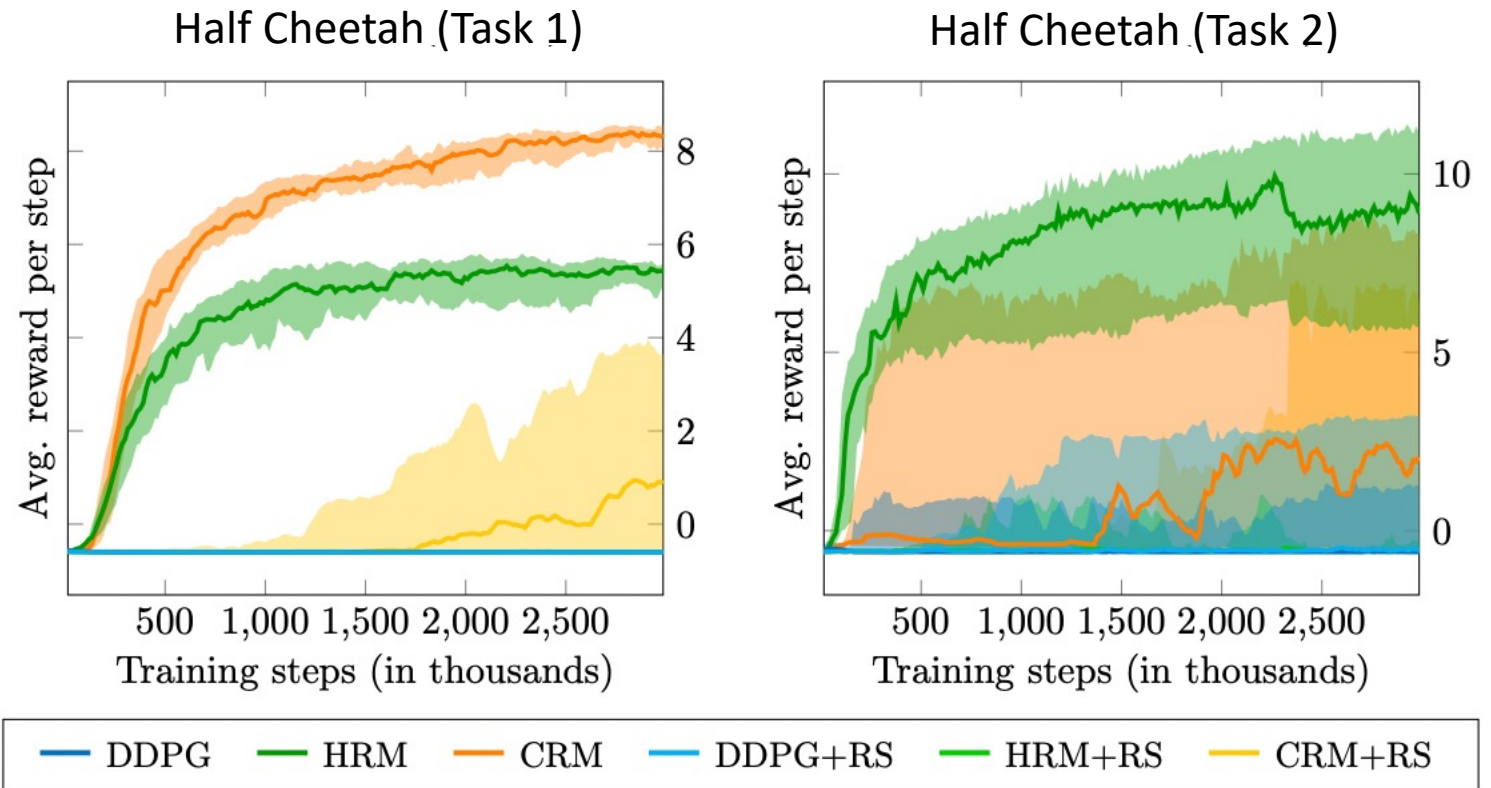
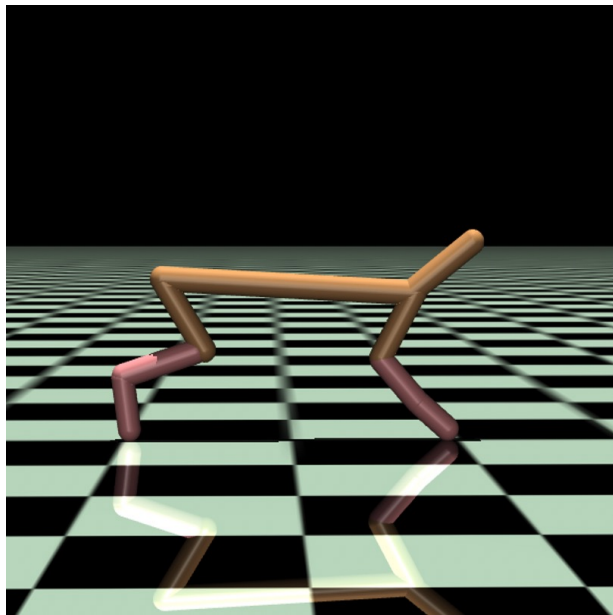
← BETTER

↑ BETTER

10 tasks over 10 random maps, 3 independent trials per combination

[Toro Icarte, Klassen, Valenzano, M., ICML18]

Half Cheetah – Continuous State and Action Domains



[Toro Icarte, Klassen, Valenzano, M., JAIR22]

The Rest of the Talk

- **Reward Machines (RM)**
- **Exploiting RM Structure in Learning**
- **Experiments**
- ▶ **Creating Reward Machines**
- **Concluding Remarks**



CREATING REWARD MACHINES

Creating Reward Machines

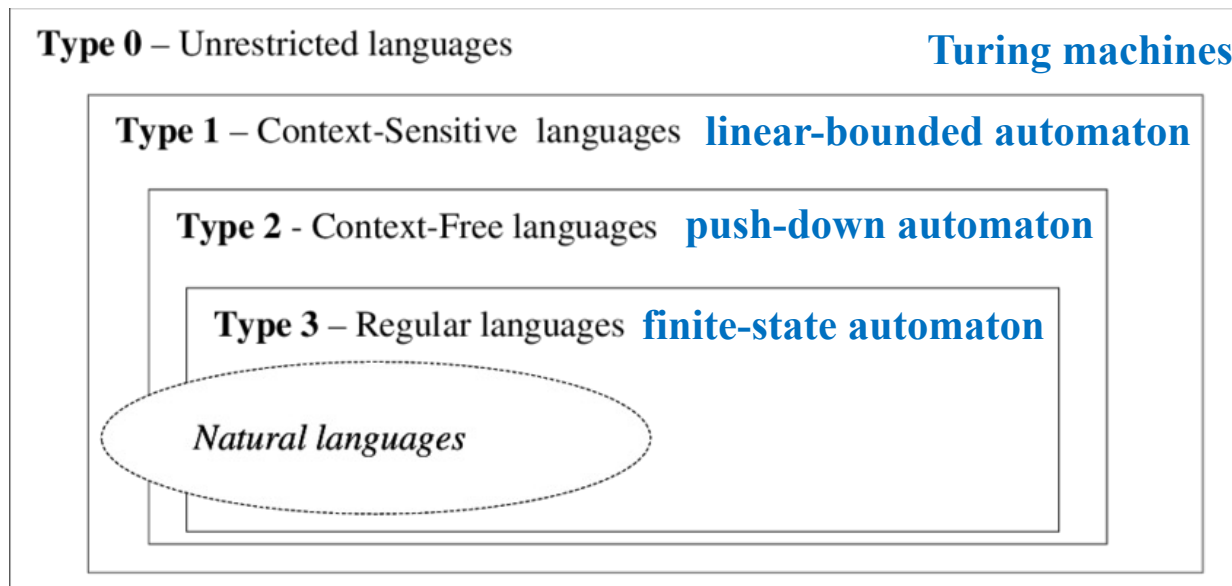
Where do Reward Machines come from?

- 1. Specify**
- 2. Generate**
- 3. Learn**

1. Reward Specification: one size does *not* fit all

Do not need to specify Reward Machines directly.

Specify reward-worthy behavior in **any formal language that is translatable to finite-state automata**.

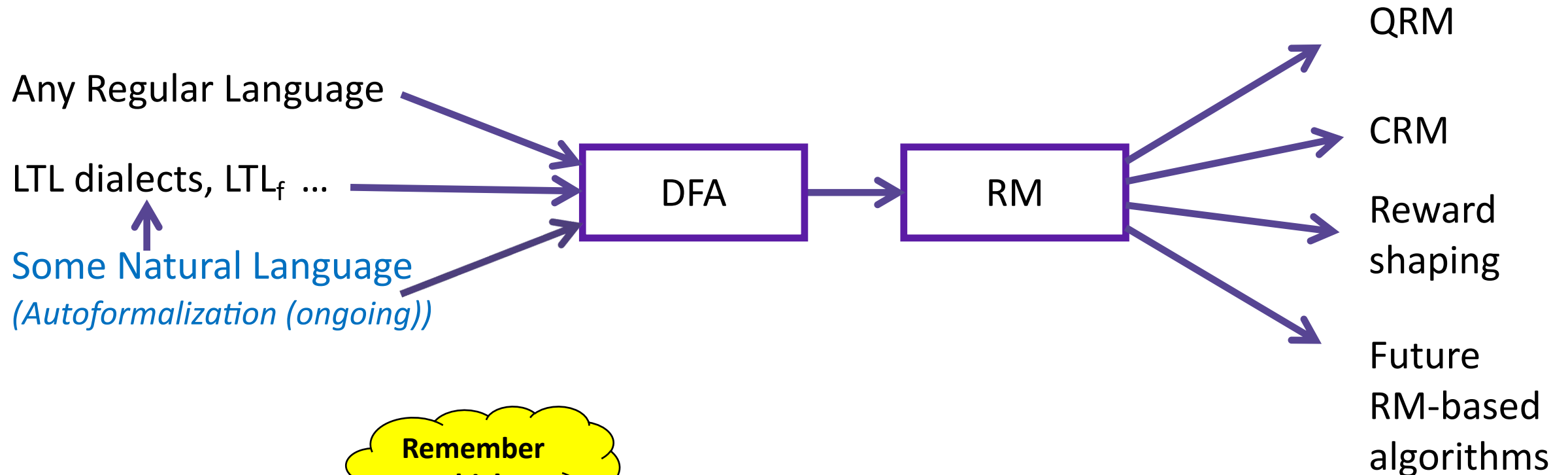


Noam Chomsky

The Chomsky Hierarchy

1. Construct Reward Machine from Formal Languages

Reward Machines serve as a **lingua franca** and provide a **normal form representation** for the reward function that **supports reward-function-tailored learning**.



Remember this!

[Camacho, Toro Icarte, Klassen, Valenzano, M., IJCAI19]
[Middleton, Klassen, Baier, M, ICAPS2020 Systems Demo]

Creating Reward Machines

Where do Reward Machines come from?

1. Specify

2. Generate

- High-level planner

- LLMs

- ...

3. Learn

Creating Reward Machines

Where do Reward Machines come from?

1. Specify

2. Generate

3. Learn

- Use RM as Memory – Partially Observable Settings
- Full Observable Settings
- ...

[Toro Icarte; Waldie; Klassen; Valenzano; Castro; M, NeurIPS 2019]

[Toro Icarte; Waldie; Klassen; Valenzano; Castro; M, AIJ23]



CONCLUDING REMARKS

TAKEAWAY

The alphabet, compositional syntax, and semantics of (formal) language can help RL agents learn, learn what to remember, plan, and reason.

*

Formal languages are powerful and effective tools to advise, instruct, task and impart knowledge to an AI that we have only begun to exploit.

Acknowledgements



Rodrigo Toro Icarte



Toryn Klassen



Richard Valenzano



Alberto Camacho



Léon Illanes



Ethan Waldie



Margarita Castro



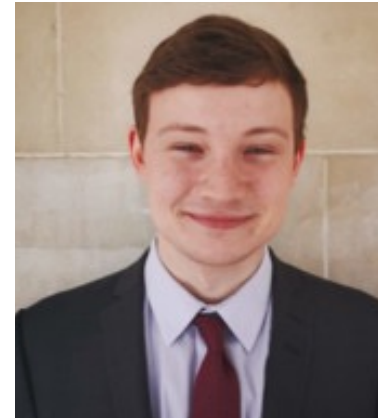
Andrew Li



Pashootan Vaezipoor



Maayan Shvo



Phillip Christoffersen



Xi Yan

Play with the code, read the papers, ...

Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning

Toro Icarte, Klassen, Valenzano, McIlraith

Journal of Artificial Intelligence Research (JAIR), 73: 173 – 208 (2022). (IJCAI-JAIR Best Paper Prize)

Code: https://bitbucket.org/RodrigoToroIcarte/reward_machines (CRM) *integrated w OpenAI Gym API*

Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning

Toro Icarte, Klassen, Valenzano, McIlraith

ICML 2018

Code: <https://bitbucket.org/RToroIcarte/qrm> (QRM) *Integrated w/ OpenAI Gym API*

Teaching Multiple Tasks to an RL Agent using LTL

Toro Icarte, Klassen, Valenzano, McIlraith

AAMAS 2018 & NeurIPS 2018 Workshop (Learning by Instructions)

Code: <https://bitbucket.org/RToroIcarte/lpopl>

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

Camacho, Toro Icarte, Klassen, Valenzano, McIlraith

IJCAI 2019

Learning Reward Machines for Partially Observable Reinforcement Learning

Toro Icarte, Waldie, Klassen, Valenzano, Castro, McIlraith

NeurIPS 2019

Symbolic Plans as High-Level Instructions for Reinforcement Learning

Illanes, Yan, Toro Icarte, McIlraith

ICAPS 2020/RLDM 2019/KR2ML@NeurIPS 2020

... and more

Other/More Recent

Reward Machines for Deep Learning in Noisy and Uncertain Environments

Li, Chen, Klassen, Vaezipoor, Toro Icarte, McIlraith
NeurIPS 2024 (to appear).

Learning Reward Machines: A Study in Partially Observable Reinforcement Learning

Toro Icarte, Klassen, Valenzano, McIlraith
AIJ, 2023.

LTL2Action: Generalizing LTL Instructions for Multi-Task RL

Vaezipoor, Li, Toro Icarte, McIlraith
ICML 2021.

Learning to Follow Instructions in Text-Based Games.

Tuli, Li, Vaezipoor, Klassen, Sanner, McIlraith.
NeurIPS 2022.



Generalization to 10^{39} tasks

Other related work

Advice-Based Exploration in Model-Based Reinforcement Learning.

Toro Icarte, Klassen, Valenzano, McIlraith

Canadian AI 2018.

Linear temporal logic (LTL) formulas and a heuristic were used to guide exploration during reinforcement learning.

Non-Markovian Rewards Expressed in LTL: Guiding Search Via Reward Shaping (Extended Version)

Camacho, Chen, Sanner, McIlraith

Extended Abstract: SoCS 2017, RLDM 2017

Full Paper: First Workshop on Goal Specifications for Reinforcement Learning, collocated with ICML/IJCAI/AAMAS, 2018.

Linear temporal logic (LTL) formulas are used to express non-Markovian reward in fully specified MDPs. LTL is translated to automata and reward shaping is used over the automata to help solve the MDP.

Learning Interpretable Models in Linear Temporal Logic

Camacho, McIlraith

ICAPS, 2019

FL-AT: A Formal Language–Automaton Transmogrifier.

Middleton, Klassen, Baier, McIlraith

ICAPS 2020 Systems Demo

Past work on Planning with Formal Languages & Automata

Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces

Camacho, Triantafillou, Muise, Baier and McIlraith

AAAI 2017

Planning with First-Order Temporally Extended Goals Using Heuristic Search

Baier and McIlraith AAI 2006

Planning with Temporally Extended Goals Using Heuristic Search

Baier and McIlraith, ICAPS 2006

Exploiting Procedural Domain Control Knowledge in State-of-the-Art Planners

Baier Fritz and McIlraith, ICAPS 2007

Beyond Classical Planning: Procedural Control Knowledge and Preferences in State-of-the-Art Planners

Baier Fritz Bienvenu and McIlraith, AAI 2008

A Heuristic Search Approach to Planning with Temporally Extended Preference

Baier, Bacchus and McIlraith Artificial Intelligence Journal, 2009

Specifying and Computing Preferred Plans

Fritz, Bienvenu and McIlraith, Artificial Intelligence Journal, 2011 (See also KR2006 paper)

•••

Questions?

