Learning Reward Machines for Partially Observable Reinforcement Learning

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

AAAI SSS-23

What is a

Reward Machine (RM)?

*Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning by Toro Icarte et al. (ICML, 2018)

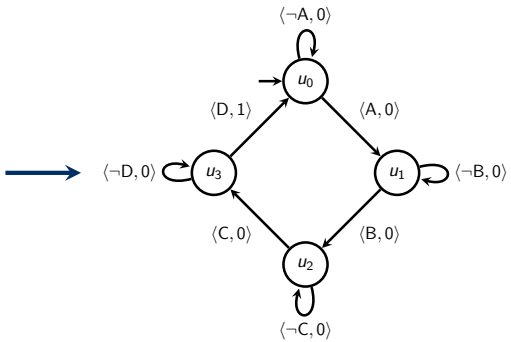
Reward Machines (RMs)

RMs are automata-based reward functions:

Reward Machines (RMs)

RMs are automata-based reward functions:

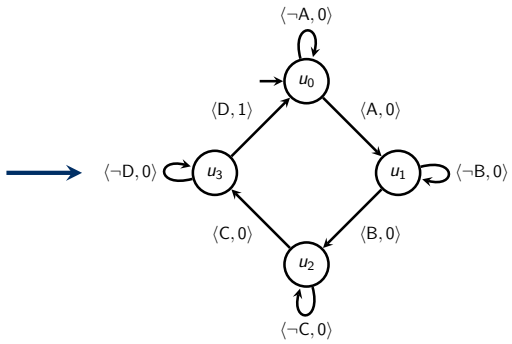
```
m = 0 # global variable
def get_reward(s):
  if m == 0 and s.at("A"):
    m = 1
  if m == 1 and s.at("B"):
    m = 2
  if m == 2 and s.at("C"):
    m = 3
  if m == 3 and s.at("D"):
    m = 0
    return 1
  return 0
```



Reward Machines (RMs)

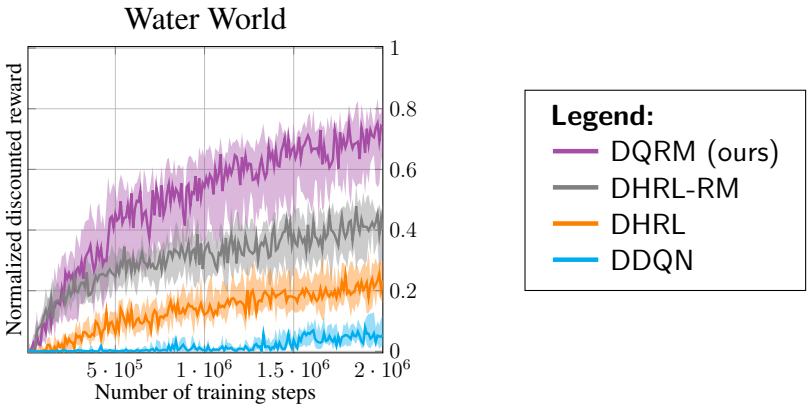
RMs are automata-based reward functions:

```
1  m = 0 # global variable
2  def get_reward(s):
3    if m == 0 and s.at("A"):
4      m = 1
5    if m == 1 and s.at("B"):
6      m = 2
7    if m == 2 and s.at("C"):
8      m = 3
9    if m == 3 and s.at("D"):
10     m = 0
11     return 1
12   return 0
```

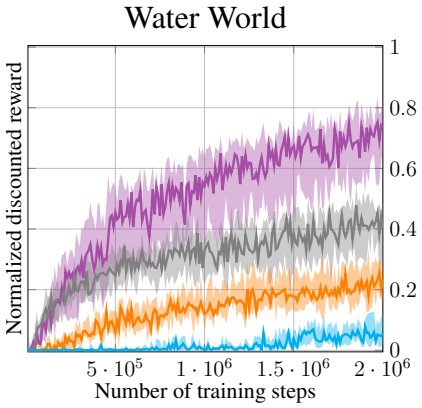


... that allow for **learning policies faster**.

Reward Machines (RMs)



Reward Machines (RMs)



Legend:
- DQRM (ours)
- DHRL-RM
- DHRL
- DDQN

... but the RMs were **handcrafted**.

Learning Reward Machines for Partially Observable RL

This work:

1. Shows how to learn RMs from experiences (LRM).

Learning Reward Machines for Partially Observable RL

This work:

1. Shows how to learn RMs from experiences (LRM).
2. Uses RMs as memory for partially observable RL.

Learning Reward Machines for Partially Observable RL

This work:

1. Shows how to learn RMs from experiences (LRM).
2. Uses RMs as memory for partially observable RL.
3. Extends QRM to work under partial observability.

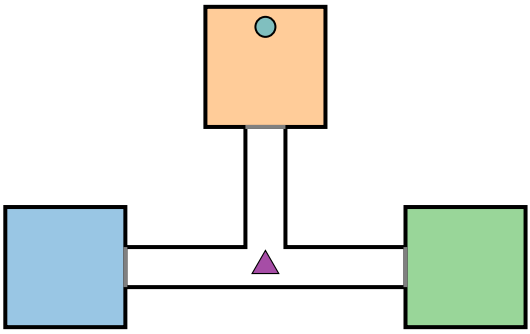
Learning Reward Machines for Partially Observable RL

This work:

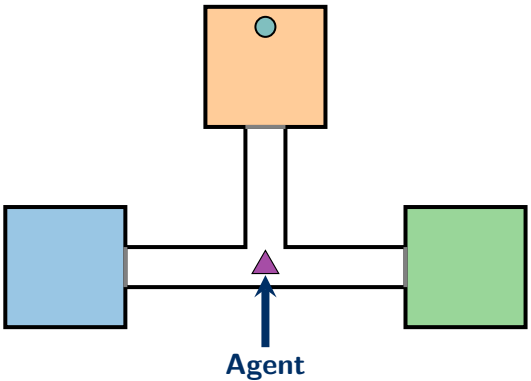
1. Shows how to learn RMs from experiences (LRM).
2. Uses RMs as memory for partially observable RL.
3. Extends QRM to work under partial observability.
4. Provides a theoretical and empirical analysis of LRM.

The Cookie Domain

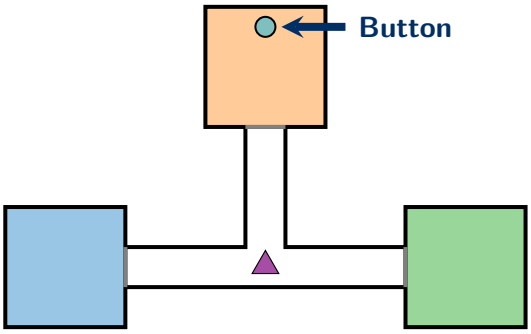
The cookie domain



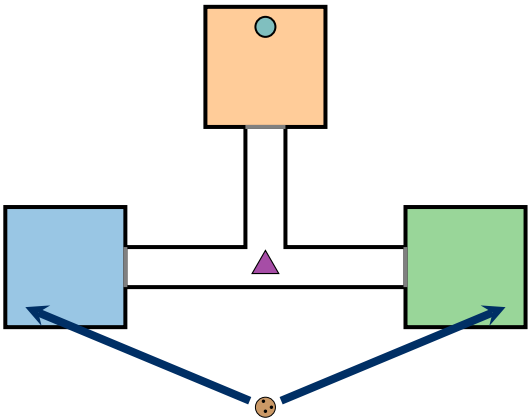
The cookie domain



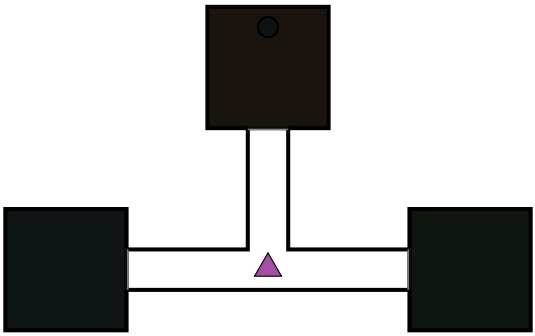
The cookie domain



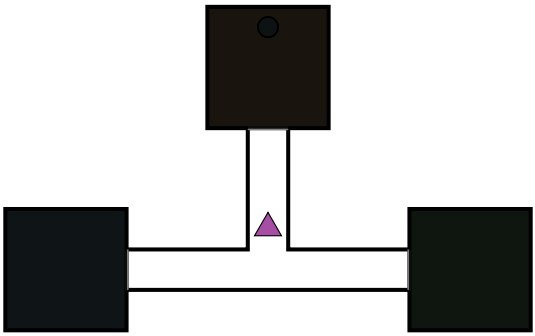
The cookie domain



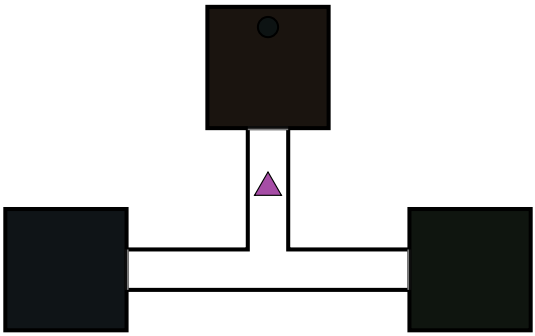
The cookie domain



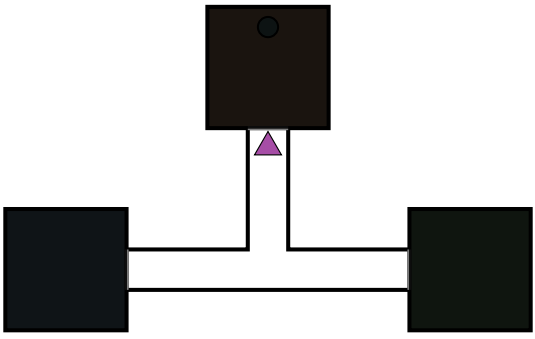
The cookie domain



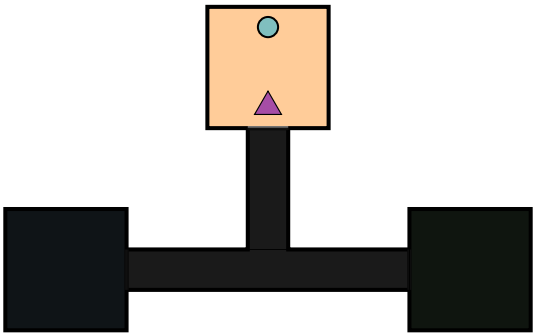
The cookie domain



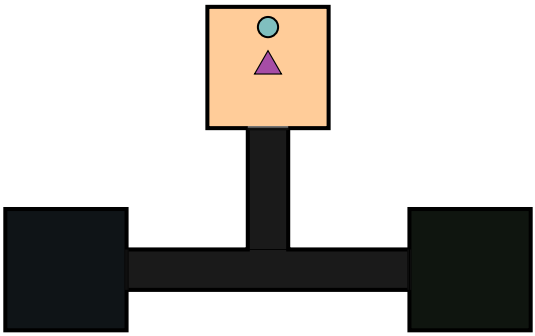
The cookie domain



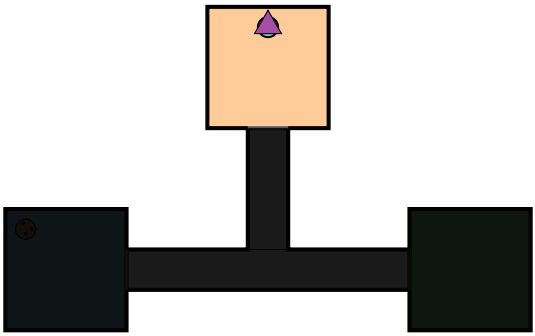
The cookie domain



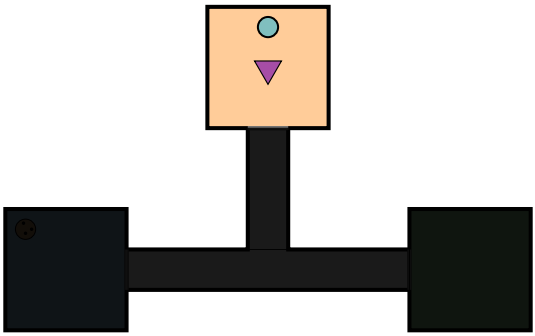
The cookie domain



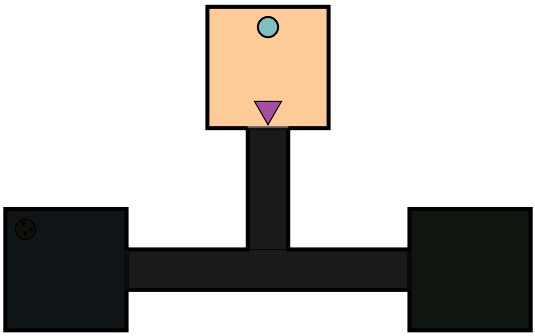
The cookie domain



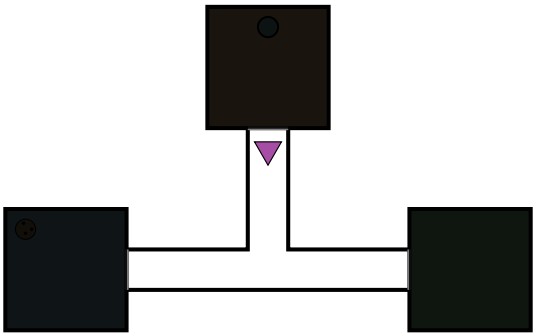
The cookie domain



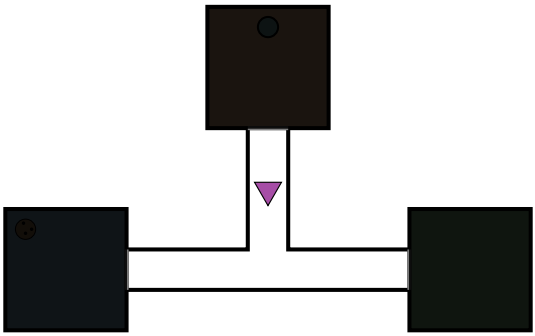
The cookie domain



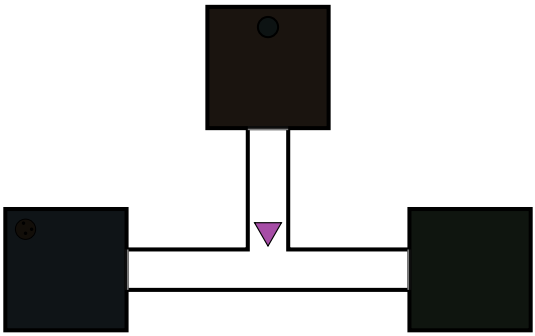
The cookie domain



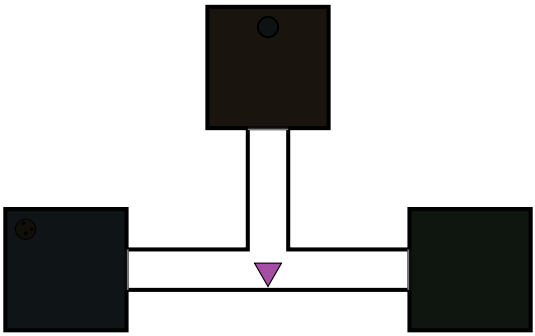
The cookie domain



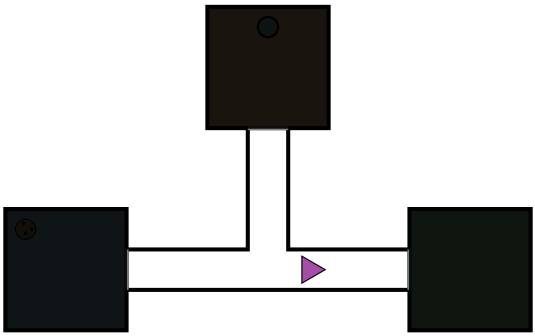
The cookie domain



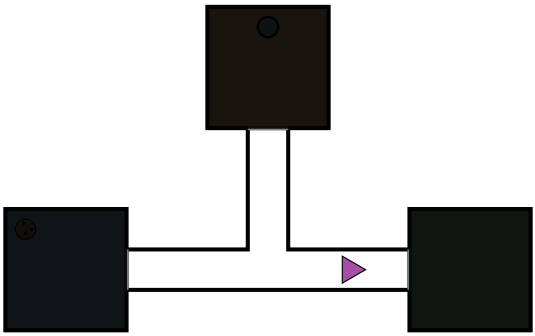
The cookie domain



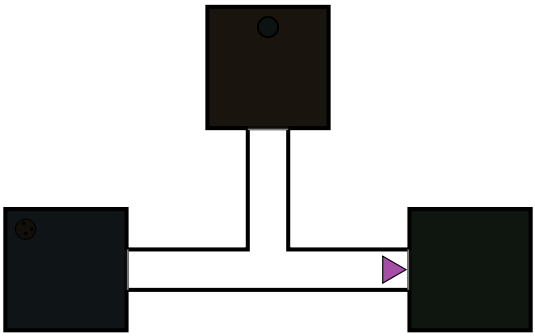
The cookie domain



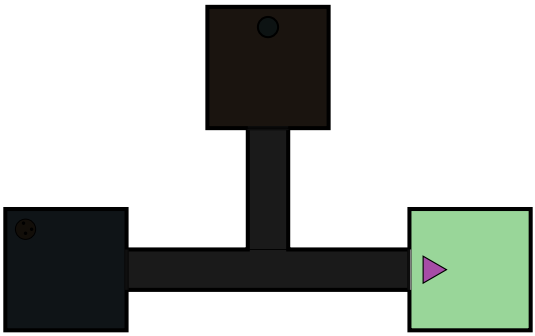
The cookie domain



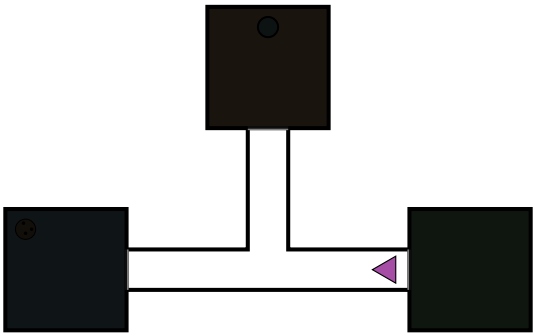
The cookie domain



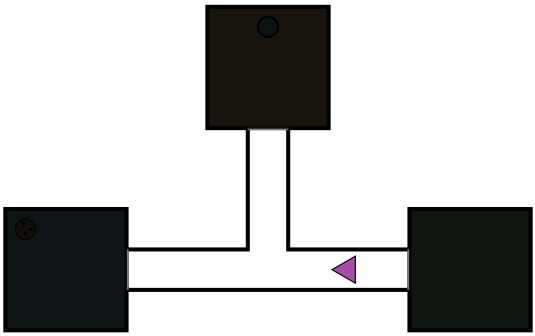
The cookie domain



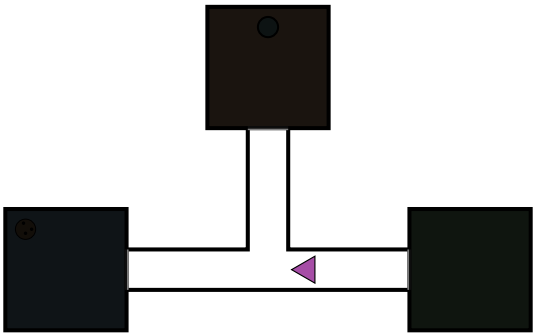
The cookie domain



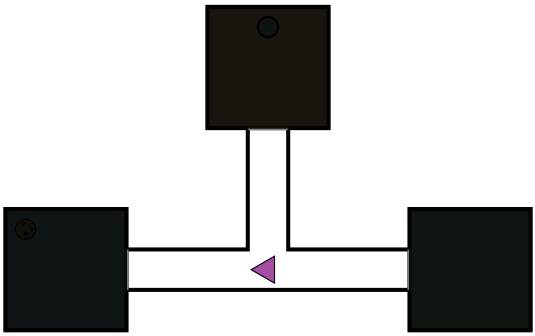
The cookie domain



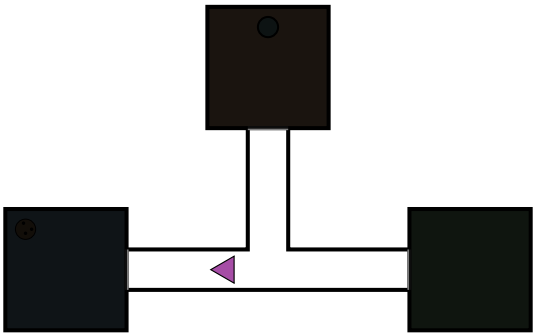
The cookie domain



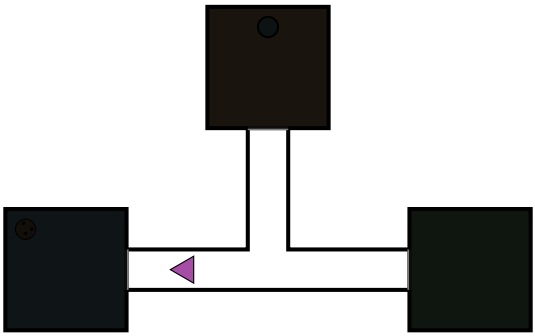
The cookie domain



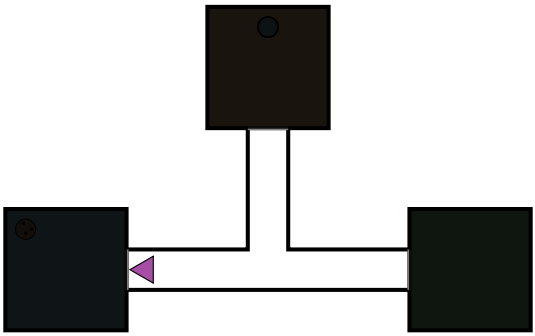
The cookie domain



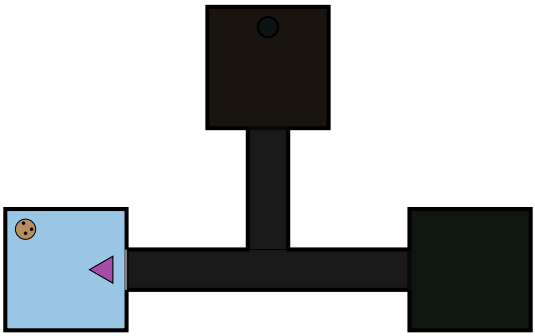
The cookie domain



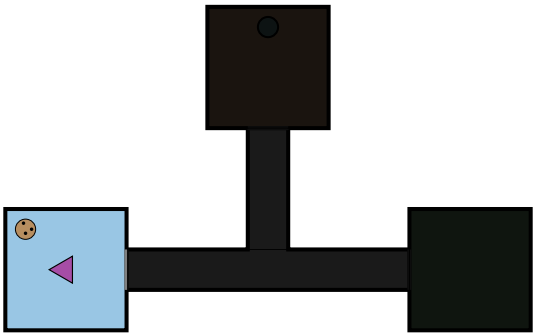
The cookie domain



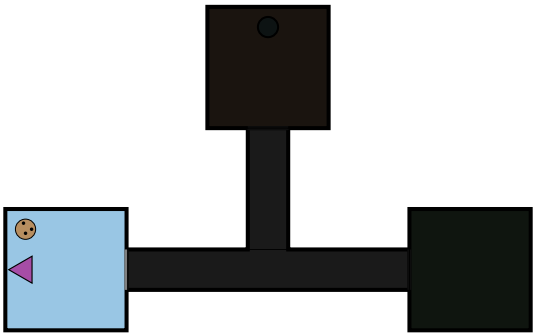
The cookie domain



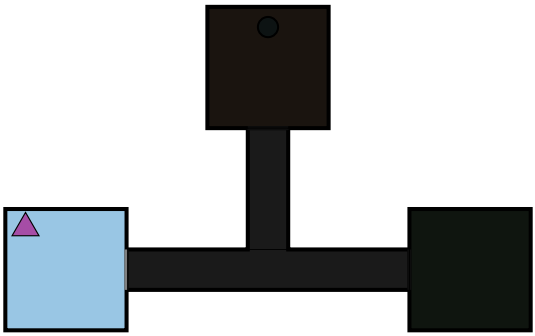
The cookie domain



The cookie domain

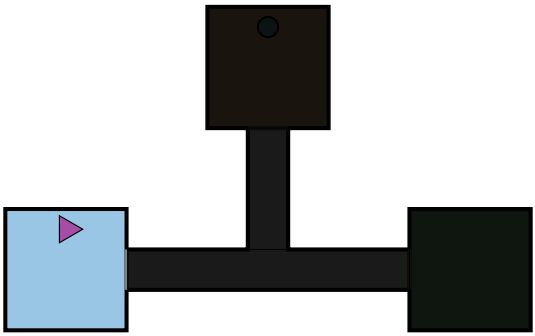


The cookie domain

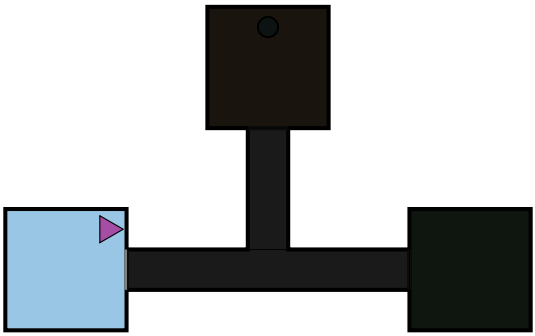


(+1 Reward)

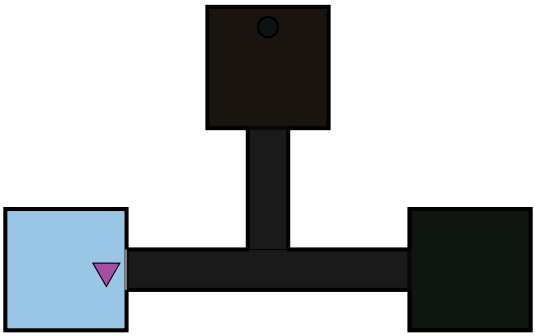
The cookie domain



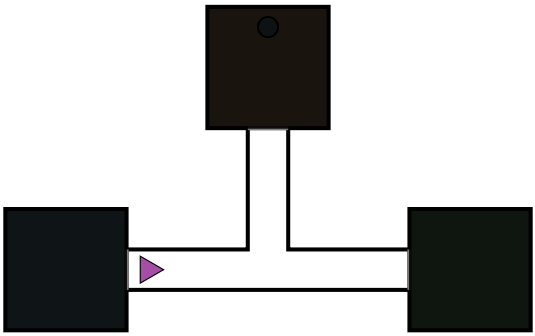
The cookie domain



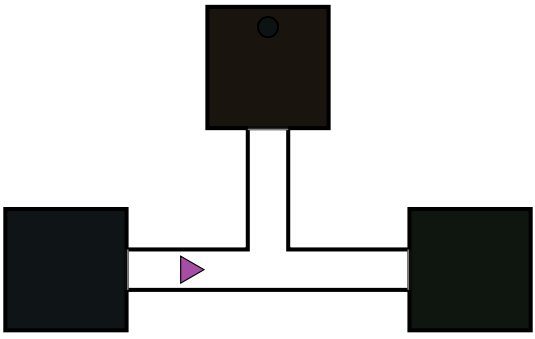
The cookie domain



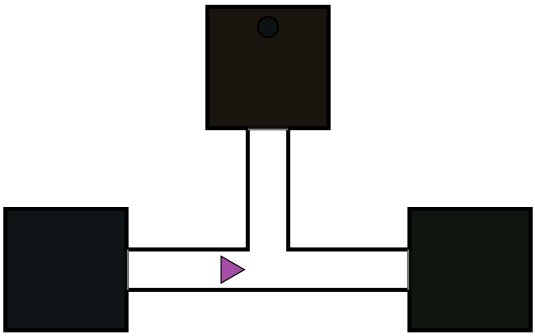
The cookie domain



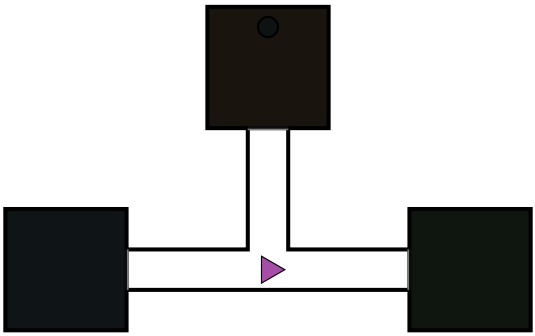
The cookie domain



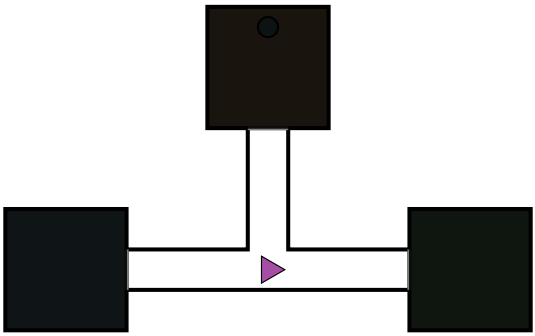
The cookie domain



The cookie domain

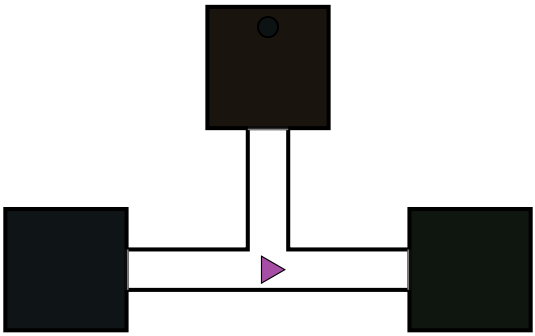


The cookie domain



Solving the cookie domain requires **memory**!

The cookie domain



Solving the cookie domain requires **memory**!

$$\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$$

Partially Observable RL

The most popular approach:

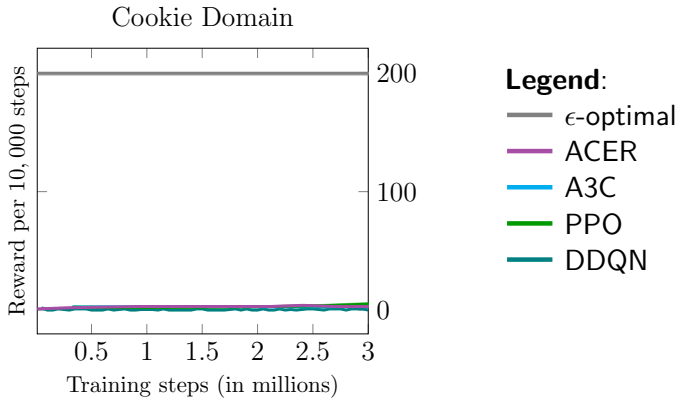
Training LSTMs policies using a policy gradient method.

Partially Observable RL

The most popular approach:

Training LSTMs policies using a policy gradient method.

... starves in the cookie domain.



RMs as memory

Reward Machines as memory

If the agent can detect the color of the rooms (🟩, ⬜, 🟦, 🟧),

Reward Machines as memory

If the agent can detect the color of the rooms (■, □, ■, ■), and when it presses the button (●),

Reward Machines as memory

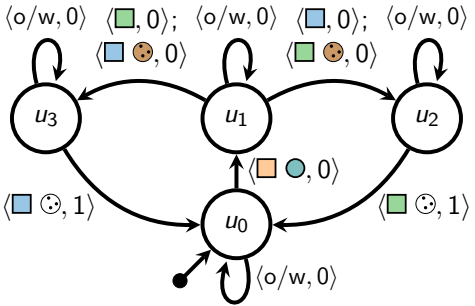
If the agent can detect the color of the rooms (🟩, ⬜, 🟦, 🟧), and when it presses the button (🔵), eats a cookie (🍪),

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 (🍪),

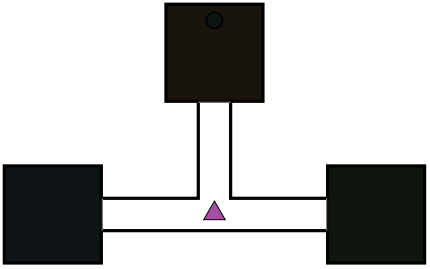
Reward Machines as memory

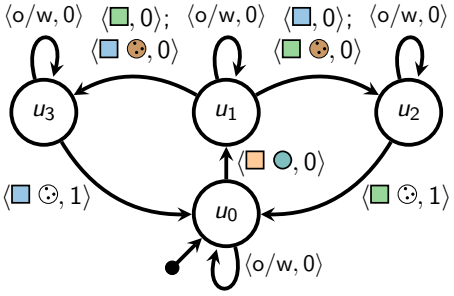
If the agent can detect the color of the rooms (■, □, ■, ■), and when it presses the button (●), eats a cookie (🍪), and sees a cookie (🍪), then:



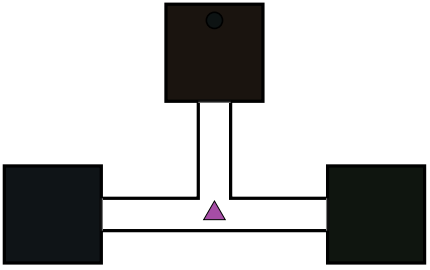
... becomes a **"perfect" memory** for the cookie domain.

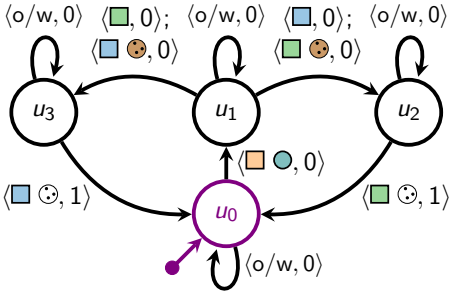
Reward Machines as memory



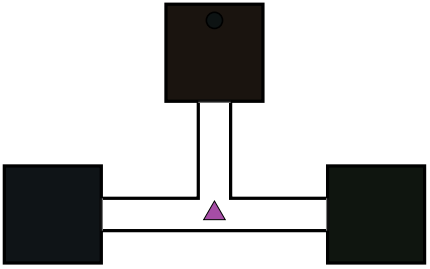


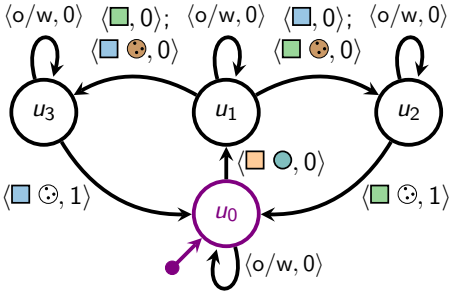
Reward Machines as memory





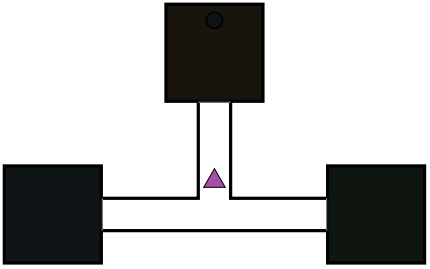
Reward Machines as memory

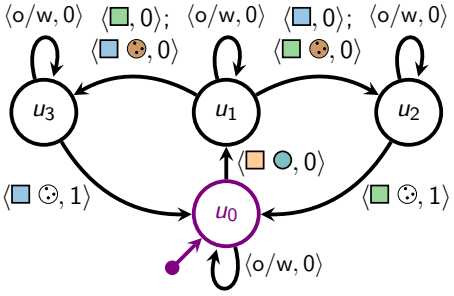




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

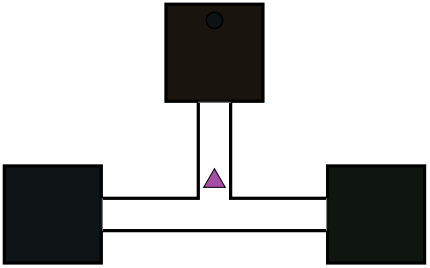
Reward Machines as memory

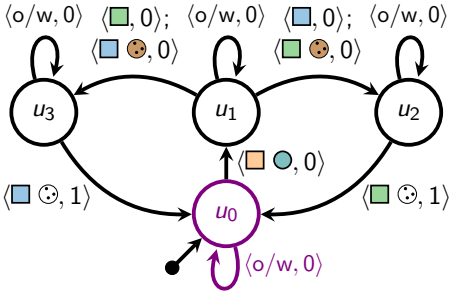




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

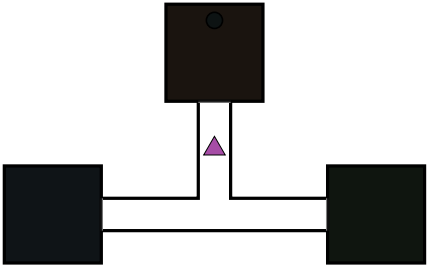
Reward Machines as memory

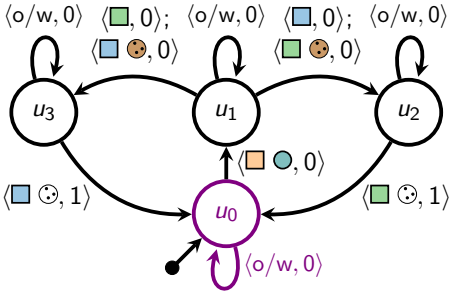




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

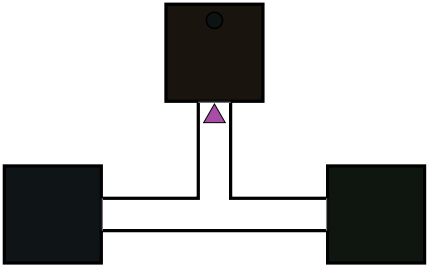
Reward Machines as memory

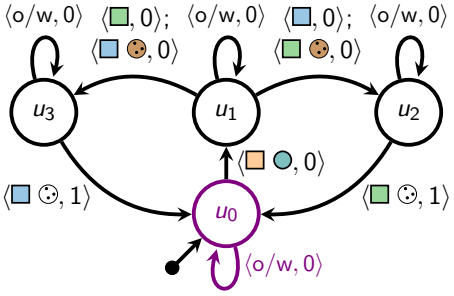




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

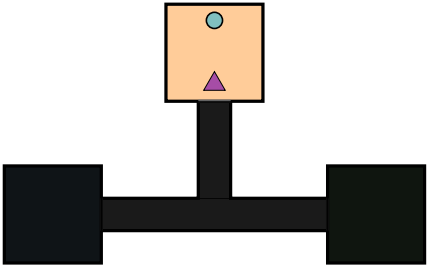
Reward Machines as memory

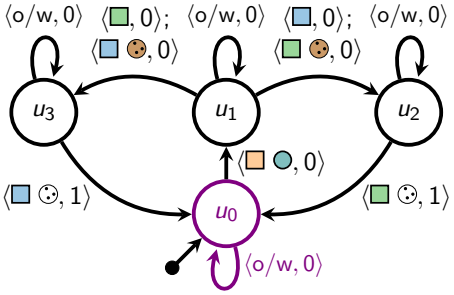




conditions at state u_0		
if ( )	$\rightarrow$	**goto** u_1
else	$\rightarrow$	**goto** u_0

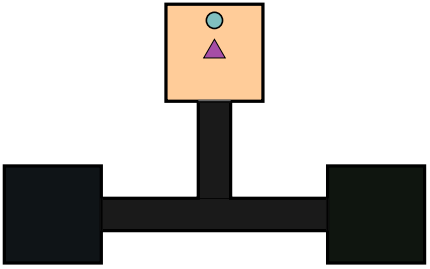
Reward Machines as memory

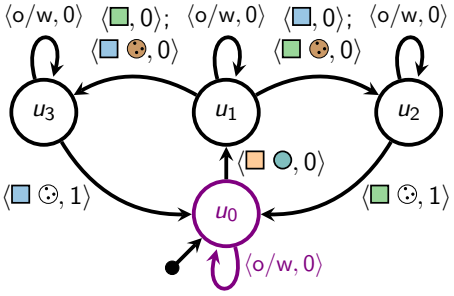




conditions at state u_0		
if ( )	$\rightarrow$	**goto** u_1
else	$\rightarrow$	**goto** u_0

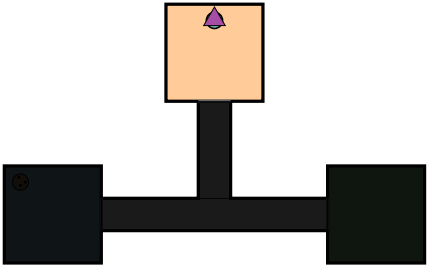
Reward Machines as memory

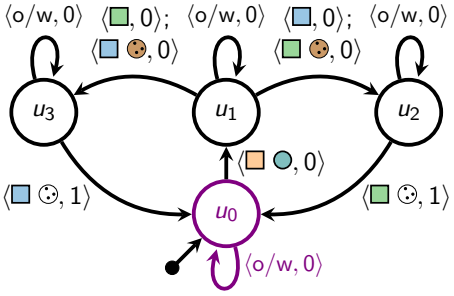




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

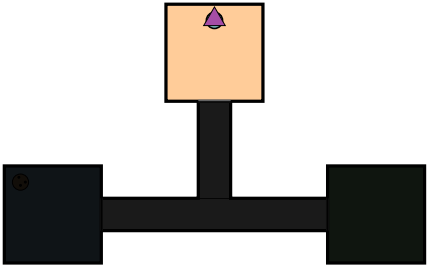
Reward Machines as memory

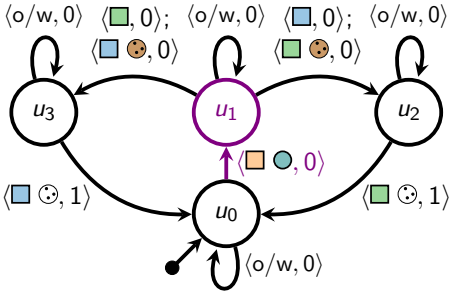




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

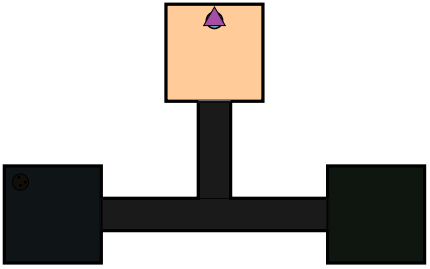
Reward Machines as memory

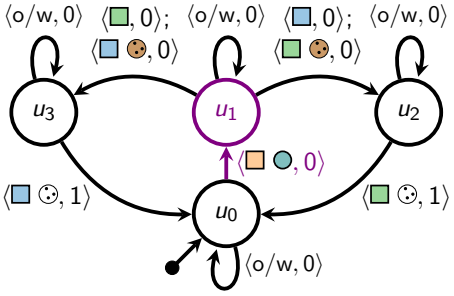




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

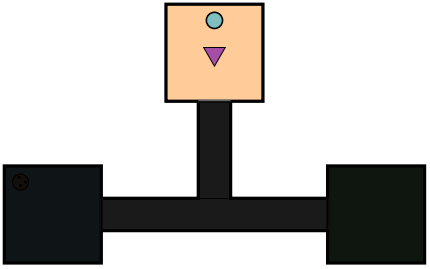
Reward Machines as memory

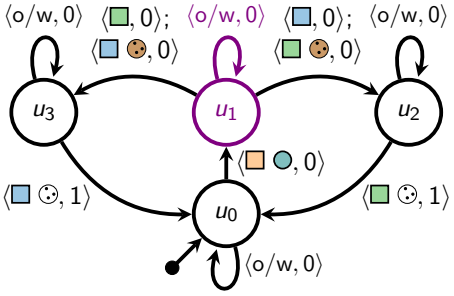




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

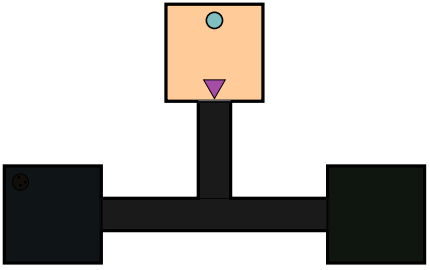
Reward Machines as memory

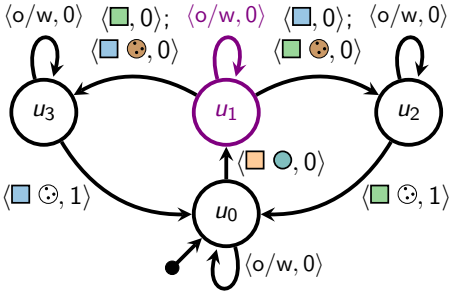




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

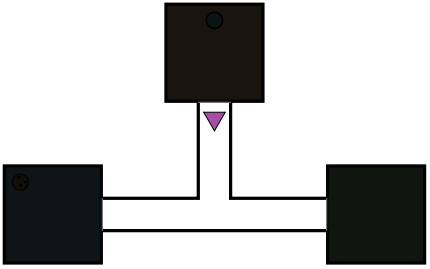
Reward Machines as memory

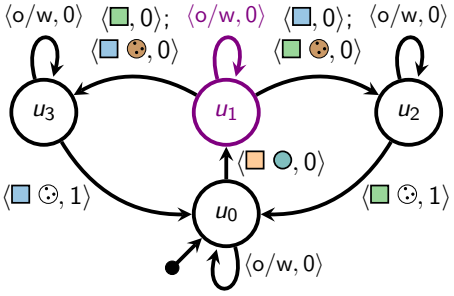




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

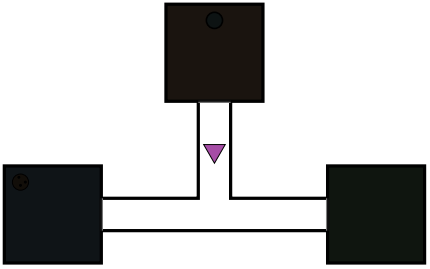
Reward Machines as memory

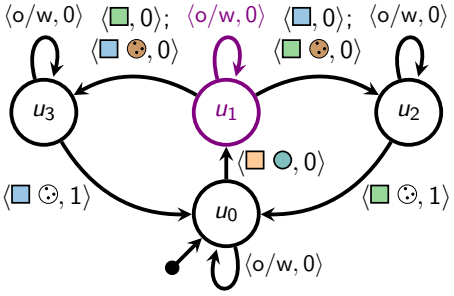




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

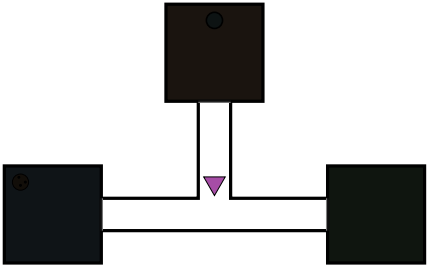
Reward Machines as memory

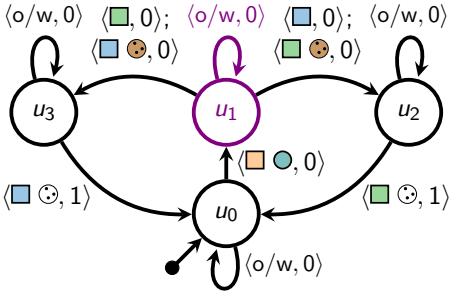




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

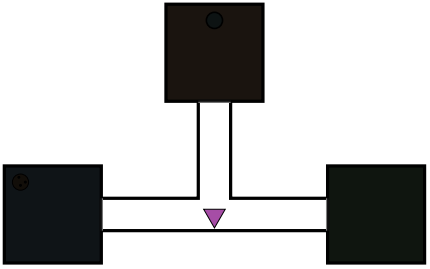
Reward Machines as memory

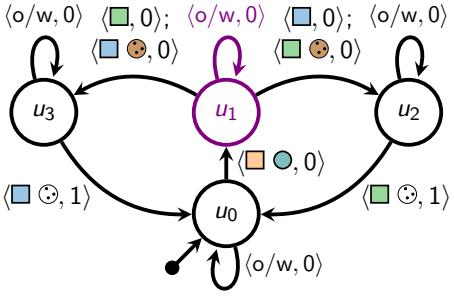




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

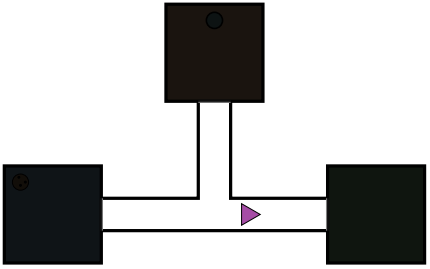
Reward Machines as memory

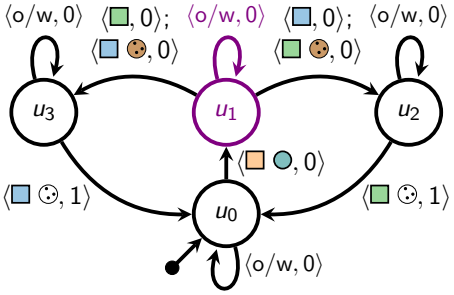




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

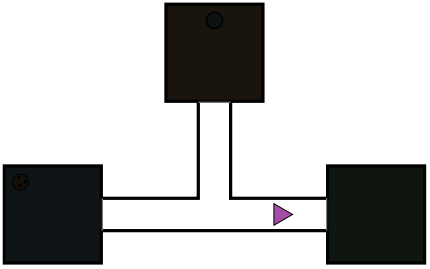
Reward Machines as memory

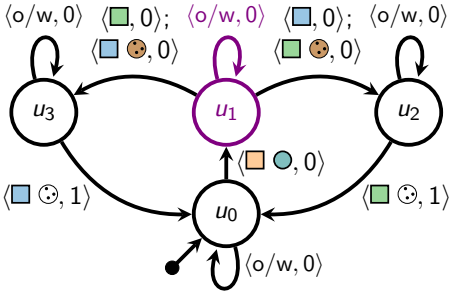




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

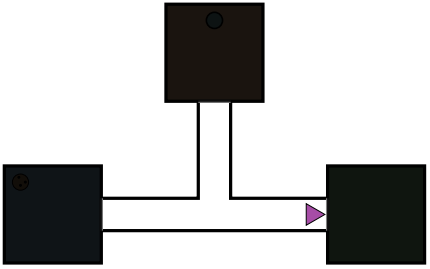
Reward Machines as memory

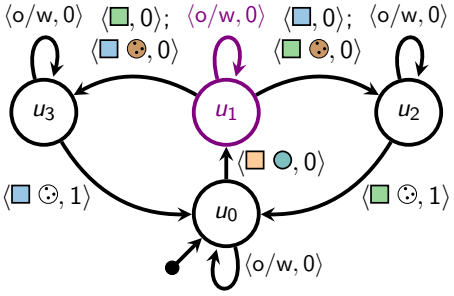




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

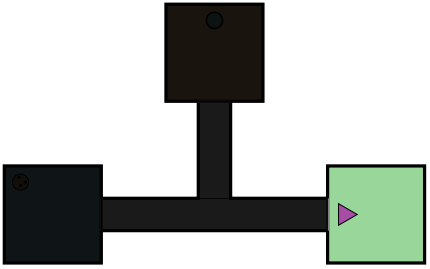
Reward Machines as memory

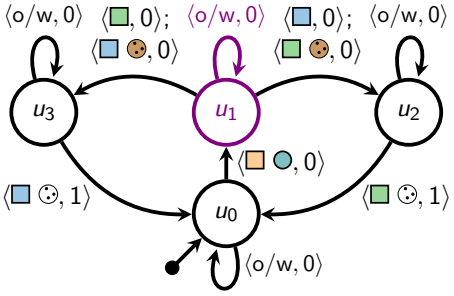




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

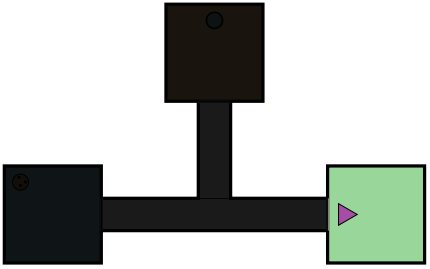
Reward Machines as memory

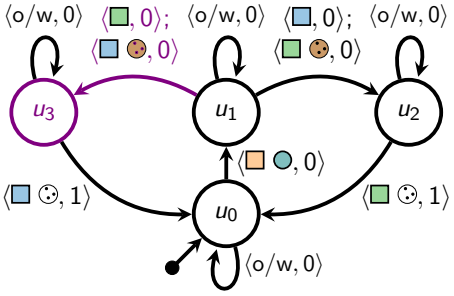




conditions at state u_1		
if (□ **or** □ ●)	→	**goto** u_2
if (□ **or** □ ●)	→	**goto** u_3
else	→	**goto** u_1

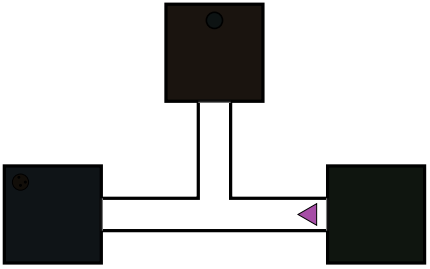
Reward Machines as memory

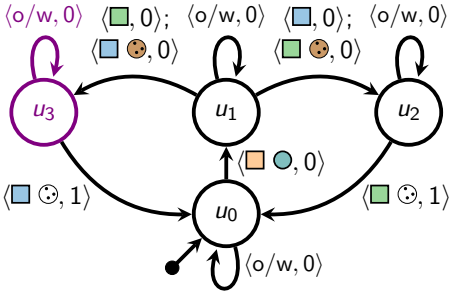




conditions at state u_3		
if (□ ☺)	→	**goto** u_0
else	→	**goto** u_3

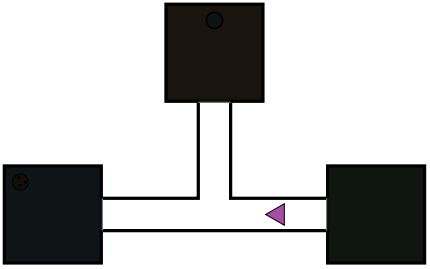
Reward Machines as memory

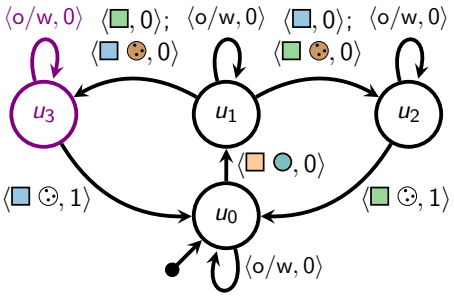




conditions at state u_3		
if (□ ⚉)	→	**goto** u_0
else	→	**goto** u_3

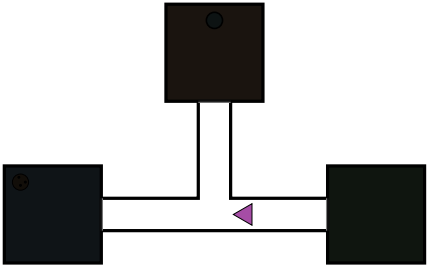
Reward Machines as memory

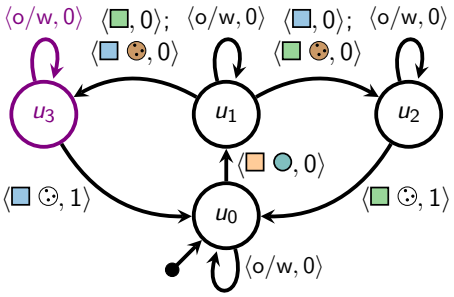




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

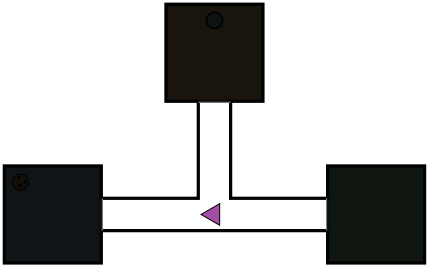
Reward Machines as memory

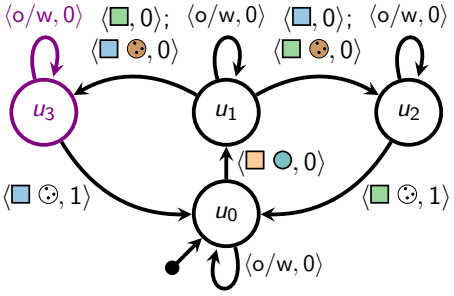




conditions at state u_3		
if (□ ⚇)	→	**goto** u_0
else	→	**goto** u_3

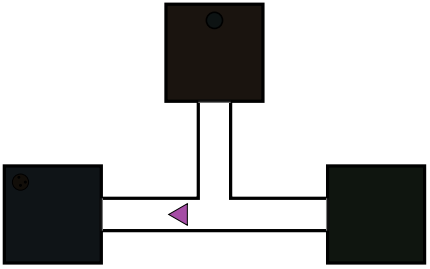
Reward Machines as memory

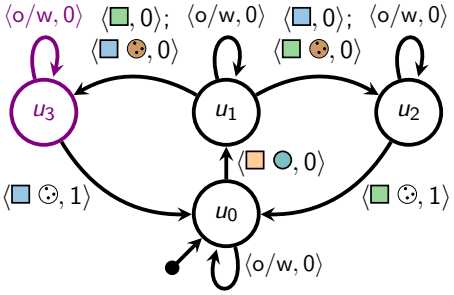




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

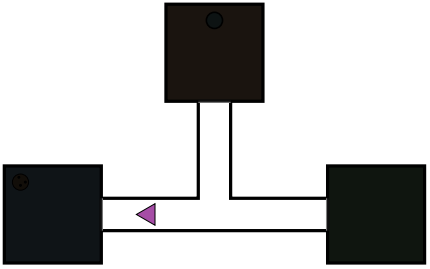
Reward Machines as memory

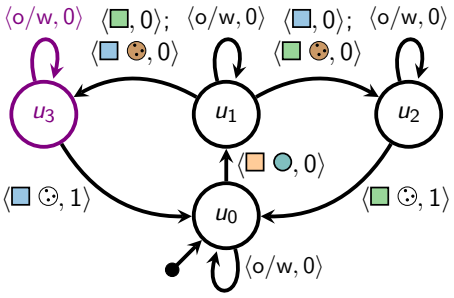




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

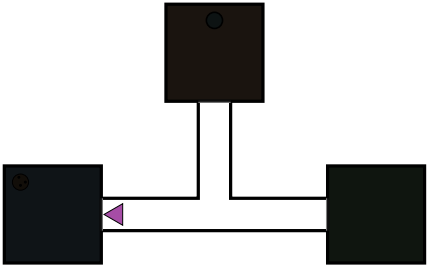
Reward Machines as memory

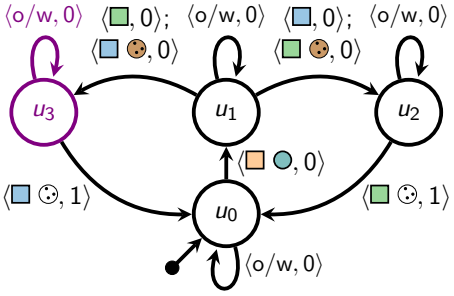




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

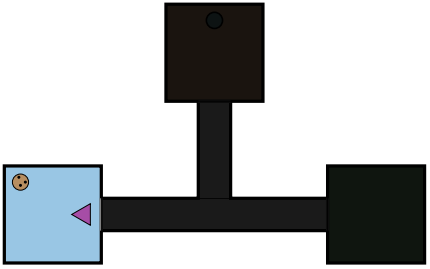
Reward Machines as memory

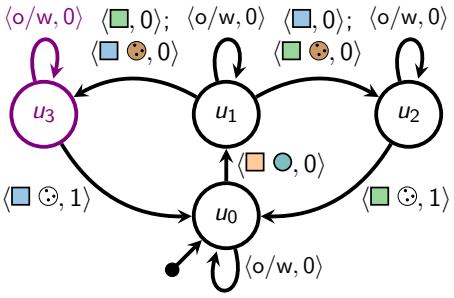




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

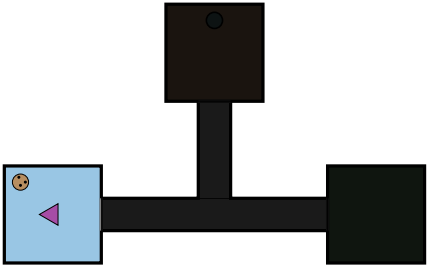
Reward Machines as memory

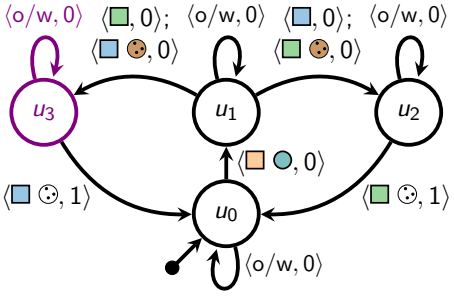




conditions at state u_3		
if (□ ⚇)	→	**goto** u_0
else	→	**goto** u_3

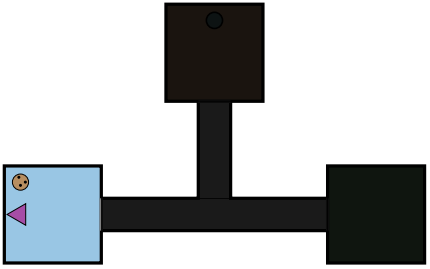
Reward Machines as memory

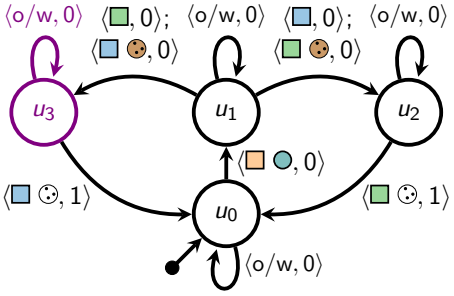




conditions at state u_3		
if (□ ⚇)	→	**goto** u_0
else	→	**goto** u_3

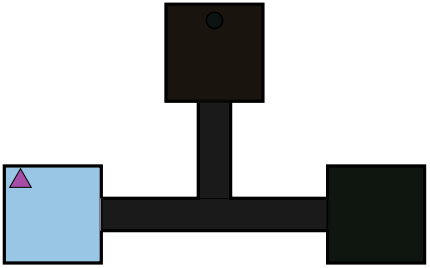
Reward Machines as memory

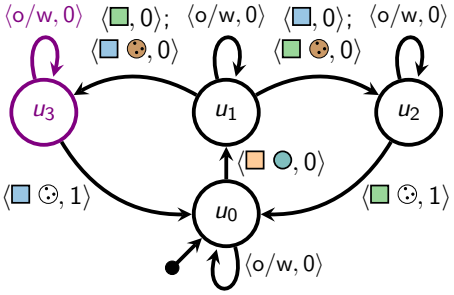




conditions at state u_3		
if (□ ⚽)	→	**goto** u_0
else	→	**goto** u_3

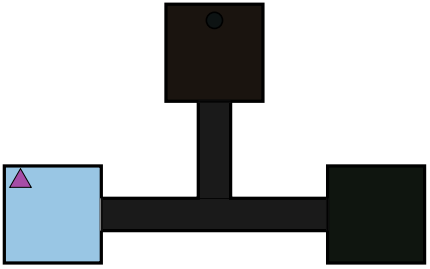
Reward Machines as memory

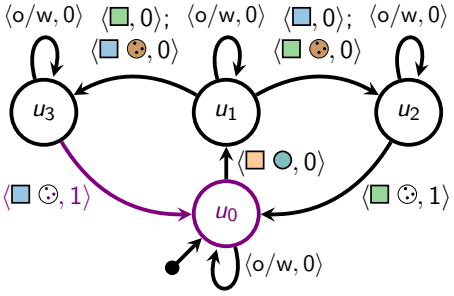




conditions at state u_3		
if (□ ⚇)	→	**goto** u_0
else	→	**goto** u_3

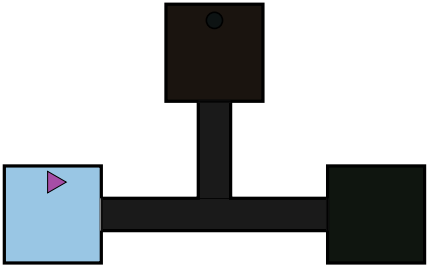
Reward Machines as memory

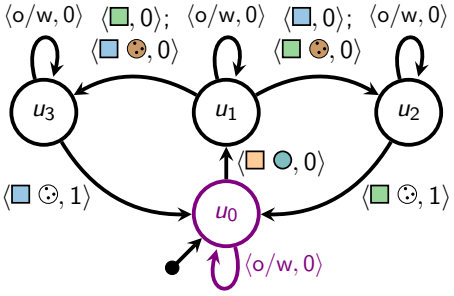




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

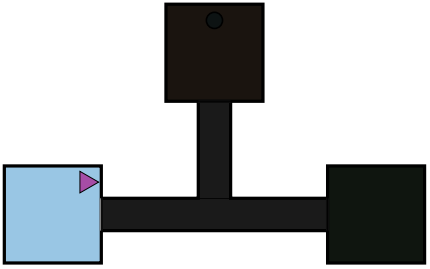
Reward Machines as memory

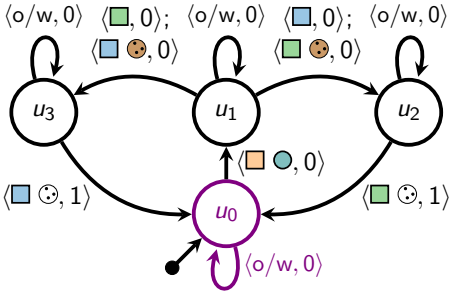




conditions at state u_0		
if ( )	$\rightarrow$	**goto** u_1
else	$\rightarrow$	**goto** u_0

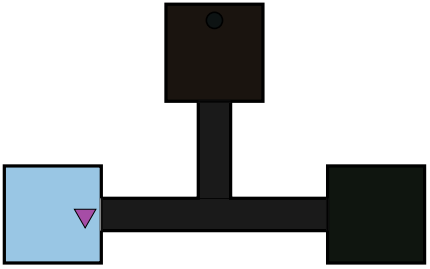
Reward Machines as memory

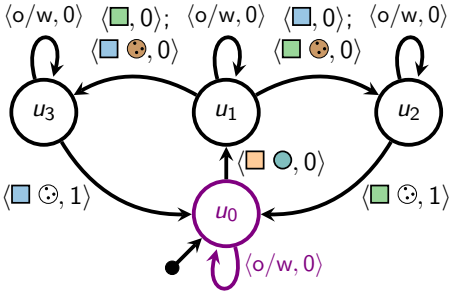




conditions at state u_0		
if ( )	$\rightarrow$	**goto** u_1
else	$\rightarrow$	**goto** u_0

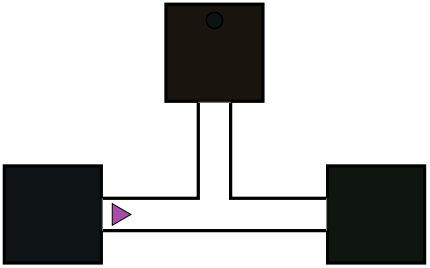
Reward Machines as memory

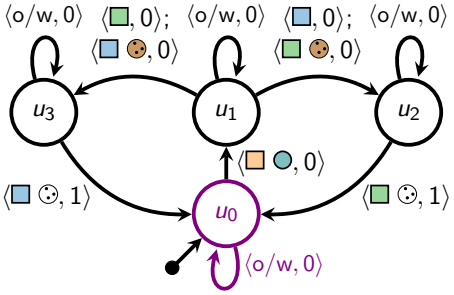




conditions at state u_0		
if ( )	→	**goto** u_1
else	→	**goto** u_0

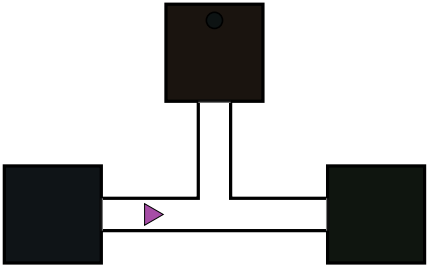
Reward Machines as memory

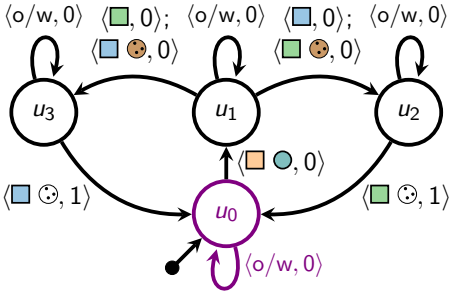




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

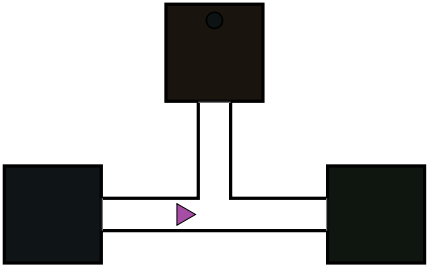
Reward Machines as memory

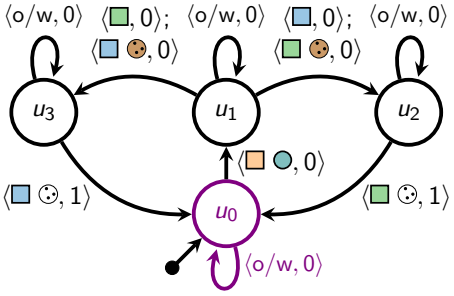




conditions at state u_0		
if (□ ○)	→	**goto** u_1
else	→	**goto** u_0

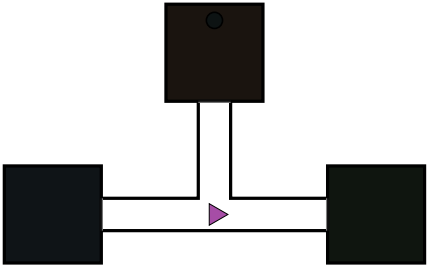
Reward Machines as memory

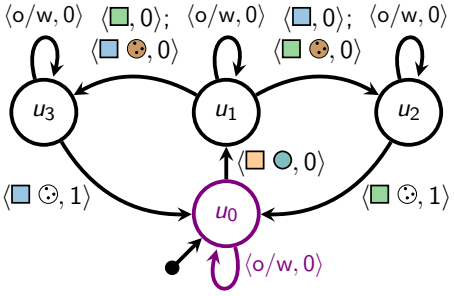




conditions at state u_0		
if ( )	→	**goto** u_1
else	→	**goto** u_0

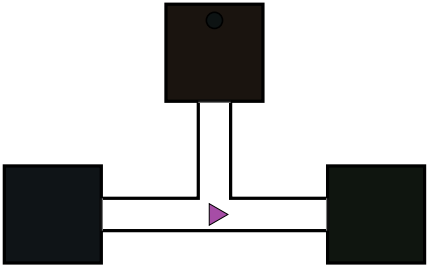
Reward Machines as memory

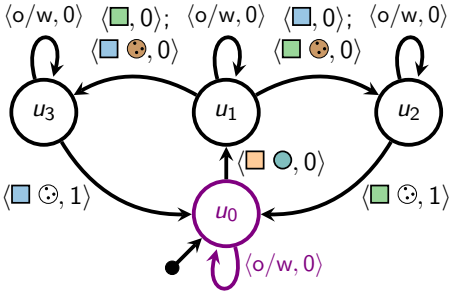




conditions at state u_0		
if ( )	$\rightarrow$	**goto** u_1
else	$\rightarrow$	**goto** u_0

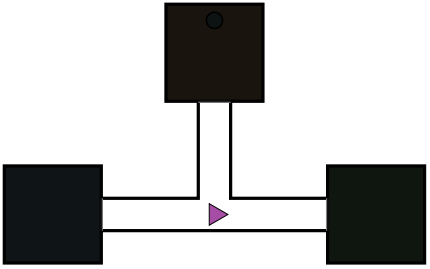
Reward Machines as memory

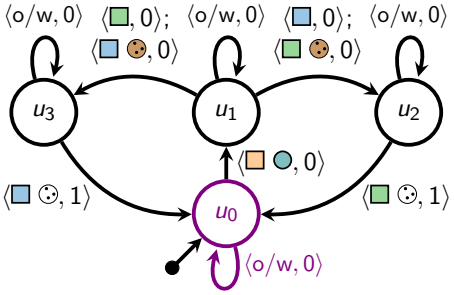




Why is this a perfect memory?

Reward Machines as memory

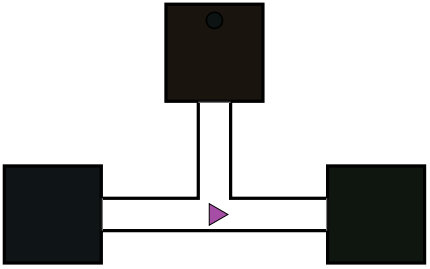


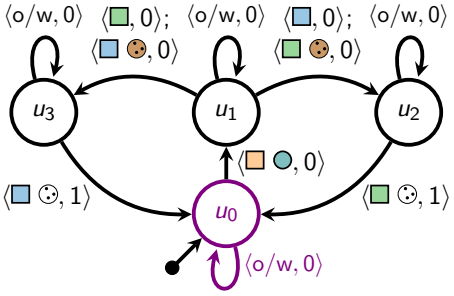


Why is this a perfect memory?

$$\pi^*(a|o_0, \cdots, o_t) = \pi^*(a|o_t, u_t)$$

Reward Machines as memory





Why is this a perfect memory?

$$\pi^*(a|o_0, \cdots, o_t) = \pi^*(a|o_t, u_t)$$

Hard problem $\xrightarrow{\text{RM}}$ Easy problem

How to learn such RMs?

Learning Reward Machines

Given a set of detectors (e.g., $\{$  ,  ,  ,  ,  ,  ,  $\}$) and traces $\mathcal{T}$,

Learning Reward Machines

Given a set of detectors (e.g., $\{$  $\}$) and traces $\mathcal{T}$, learning RMs is a **discrete optimization** problem:

$$\underset{\langle U, u_0, \delta_u, \delta_r \rangle}{\text{minimize}} \sum_{i \in I} \sum_{t \in T_i} \log(|N_{x_{i,t}, L(e_{i,t})}|) \tag{LRM}$$

$$s.t.\ \langle U, u_0, \delta_u, \delta_r \rangle \in \mathcal{R}_{\mathcal{P}} \tag{1}$$

$$|U| \leq u_{\max} \tag{2}$$

$$x_{i,t} \in U \qquad \forall i \in I, t \in T_i \cup \{t_i\} \tag{3}$$

$$x_{i,0} = u_0 \qquad \forall i \in I \tag{4}$$

$$x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) \qquad \forall i \in I, t \in T_i \tag{5}$$

$$N_{u,l} \subseteq 2^{2^{\mathcal{P}}} \qquad \forall u \in U, l \in 2^{\mathcal{P}} \tag{6}$$

$$L(e_{i,t+1}) \in N_{x_{i,t}, L(e_{i,t})} \qquad \forall i \in I, t \in T_i \tag{7}$$

Learning Reward Machines

Given a set of detectors (e.g., $\{$  ,  ,  ,  ,  ,  ,  $\}$) and traces $\mathcal{T}$, learning RMs is a **discrete optimization** problem:

$$\underset{\langle U,u_0,\delta_u,\delta_r\rangle}{\text{minimize}} \sum_{i\in I}\sum_{t\in T_i} \log(|N_{x_{i,t},L(e_{i,t})}|) \qquad \text{(LRM)}$$

$$s.t.\ \langle U,u_0,\delta_u,\delta_r\rangle \in \mathcal{R}_{\mathcal{P}} \qquad (1)$$

$$|U| \leq u_{\max} \qquad (2)$$

$$x_{i,t} \in U \qquad \forall i \in I, t \in T_i \cup \{t_i\} \qquad (3)$$

$$x_{i,0} = u_0 \qquad \forall i \in I \qquad (4)$$

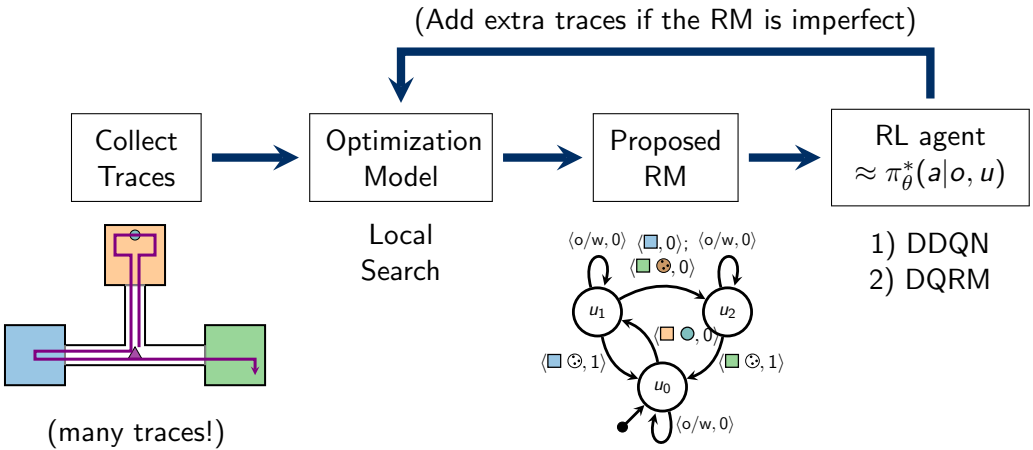
$$x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) \qquad \forall i \in I, t \in T_i \qquad (5)$$

$$N_{u,l} \subseteq 2^{2^{\mathcal{P}}} \qquad \forall u \in U, l \in 2^{\mathcal{P}} \qquad (6)$$

$$L(e_{i,t+1}) \in N_{x_{i,t},L(e_{i,t})} \qquad \forall i \in I, t \in T_i \qquad (7)$$

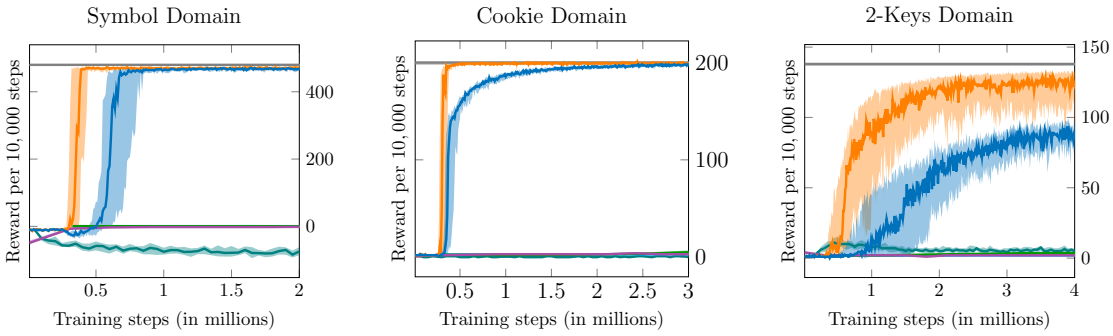
... that we solved using **local search**.

Overall approach



Results

Results



*__Note__: The detectors were also given to the baselines.

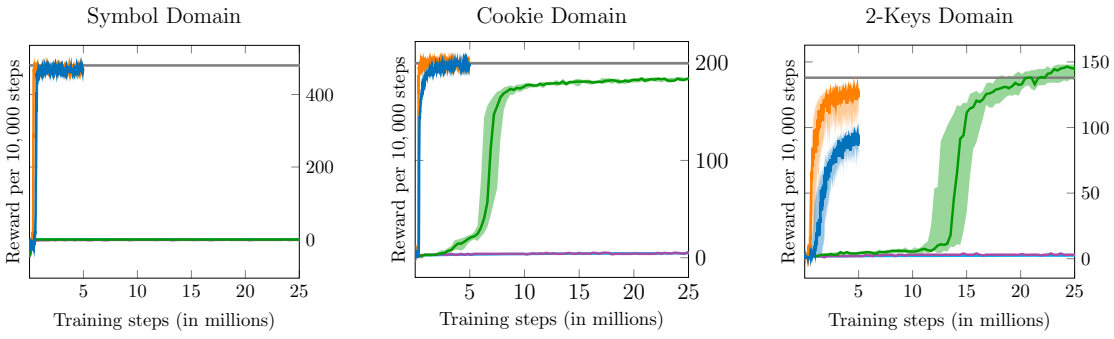
Discussion

Frequently asked questions

1) Would the LSTM-based baselines eventually learn to solve these domains?

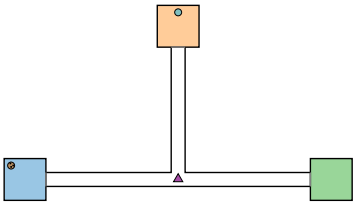
Frequently asked questions

1) Would the LSTM-based baselines eventually learn to solve these domains?



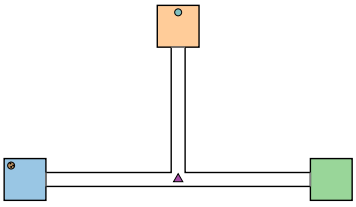
Frequently asked questions

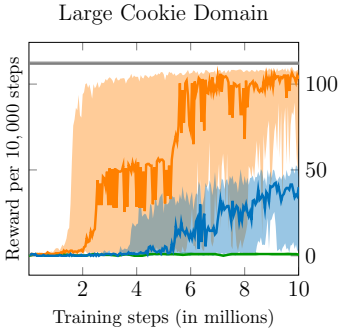
1) Would the LSTM-based baselines eventually learn to solve these domains?



Frequently asked questions

1) Would the LSTM-based baselines eventually learn to solve these domains?

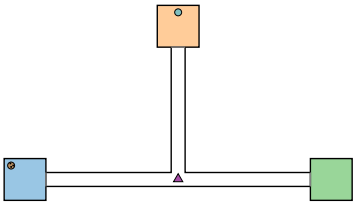


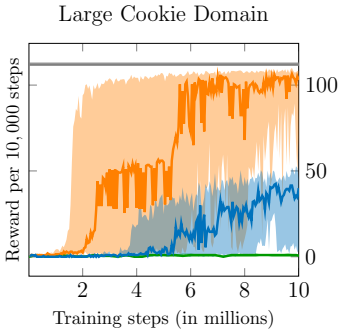


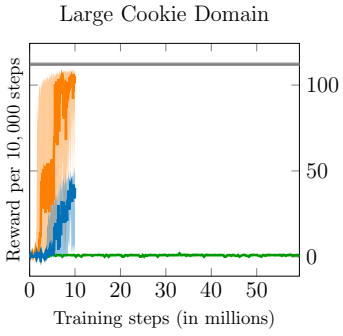
Legend: ϵ-optimal, A3C, ACER, PPO, DDQN, LRM+DDQN, LRM+DQRM

Frequently asked questions

1) Would the LSTM-based baselines eventually learn to solve these domains?







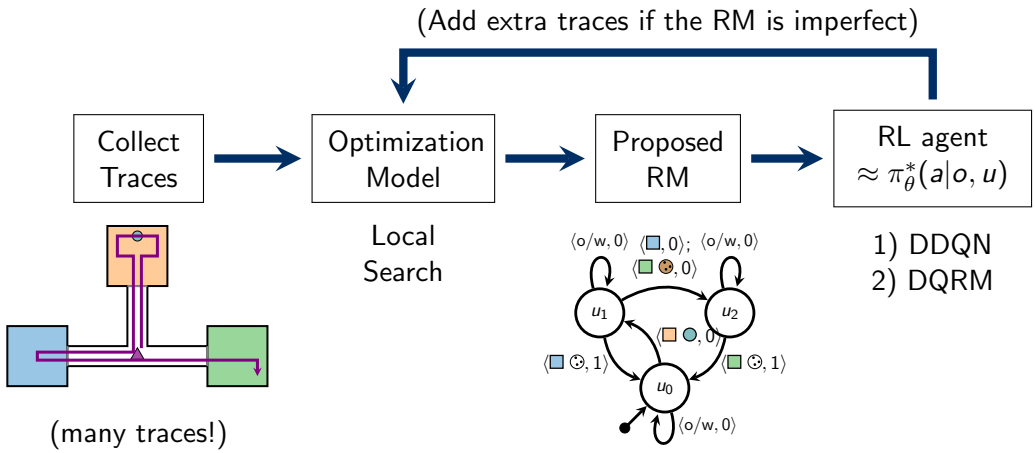
Legend:
ϵ-optimal
A3C
ACER
PPO
DDQN
LRM+DDQN
LRM+DQRM

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

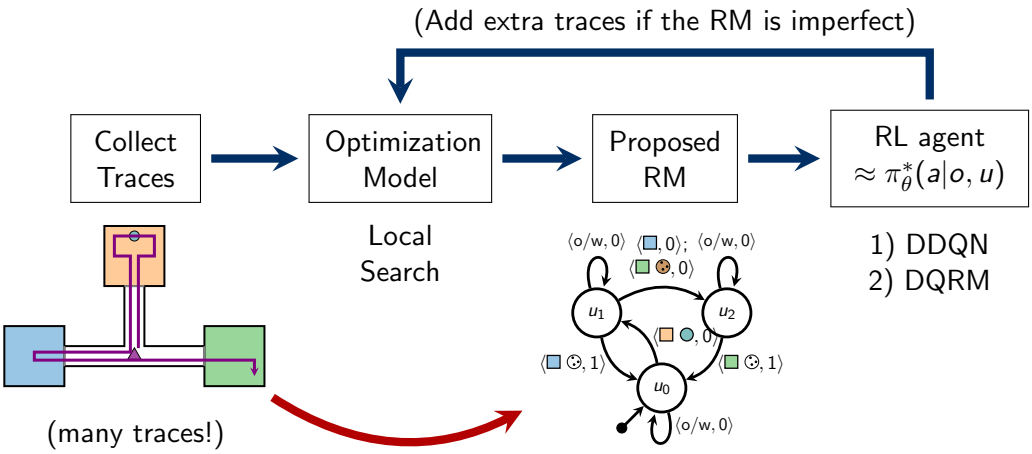
Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?



Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?



Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

A. LRM tries to solve POMDPs

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

A. LRM tries to solve POMDPs

POMDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

A. LRM tries to solve POMDPs

POMDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Thus, LRM's learning objective is to find a machine such that:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, u_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, u_t, a_t)$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

A. LRM tries to solve POMDPs

POMDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Thus, LRM's learning objective is to find a machine such that:

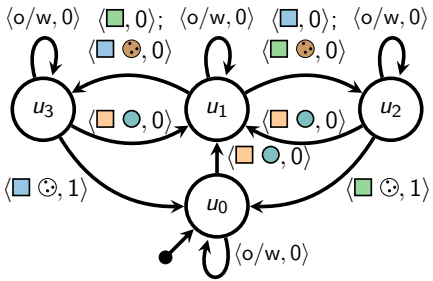
- $P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, u_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, u_t, a_t)$

Result: $\pi^*(a_t|o_t, u_t)$ optimally solves the POMDP.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

A. LRM tries to solve POMDPs



In the cookie domain, LRM learns this RM because it holds that

$$P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, u_t, a_t)$$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

NMRDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

NMRDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Thus, their learning objective is to find the smallest machine such that:

- $P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, u_t, a_t)$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

NMRDPs are hard because:

- $P(o_{t+1}|o_0, \cdots, o_t, a_t) = P(o_{t+1}|o_t, a_t)$
- $P(r_{t+1}|o_0, \cdots, o_t, a_t) \neq P(r_{t+1}|o_t, a_t)$
- As a result, $\pi^*(a|o_t) \ll \pi^*(a|o_0, \cdots, o_t)$

Thus, their learning objective is to find the smallest machine such that:

- $P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, u_t, a_t)$

Result: $\pi^*(a_t|o_t, u_t)$ optimally solves the NMRDP.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

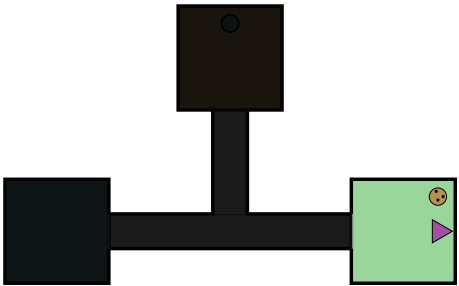
These methods do not work in our domains because our domains are not NMRDPs.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

These methods do not work in our domains because our domains are not NMRDPs.



In the cookie domain:

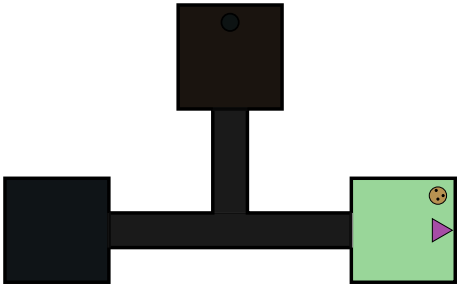
$$P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

These methods do not work in our domains because our domains are not NMRDPs.



In the cookie domain:

$$P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$$

And

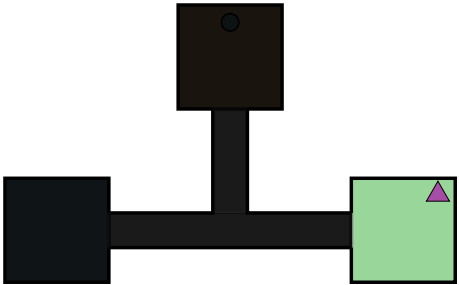
$$P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, a_t)$$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

These methods do not work in our domains because our domains are not NMRDPs.



In the cookie domain:

$$P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$$

And

$$P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, a_t)$$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

B. Methods that solve NMRDPs (e.g., JIRP, ISA, SRMI)

These methods do not work in our domains because our domains are not NMRDPs.

In the cookie domain:

$$P(o_{t+1}|o_0, \cdots, o_t, a_t) \neq P(o_{t+1}|o_t, a_t)$$

And

$$P(r_{t+1}|o_0, \cdots, o_t, a_t) = P(r_{t+1}|o_t, a_t)$$

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

MDPs are usually easy:

- $P(o_{t+1}, r_{t+1} | o_0, \cdots, o_t, a_t) = P(o_{t+1}, r_{t+1} | o_t, a_t)$

... but MDPs with sparse rewards are hard.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

MDPs are usually easy:

- $P(o_{t+1}, r_{t+1} | o_0, \cdots, o_t, a_t) = P(o_{t+1}, r_{t+1} | o_t, a_t)$

... but MDPs with sparse rewards are hard.

Thus, their learning objective is to find the smallest machine such that:

- It accepts traces that **can** be generated by interacting with the environment.
- It rejects traces that **cannot** be generated by interacting with the environment.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

MDPs are usually easy:

- $P(o_{t+1}, r_{t+1} | o_0, \cdots, o_t, a_t) = P(o_{t+1}, r_{t+1} | o_t, a_t)$

... but MDPs with sparse rewards are hard.

Thus, their learning objective is to find the smallest machine such that:

- It accepts traces that **can** be generated by interacting with the environment.
- It rejects traces that **cannot** be generated by interacting with the environment.

Result: They learn a high-level model that is then used to encourage exploration.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

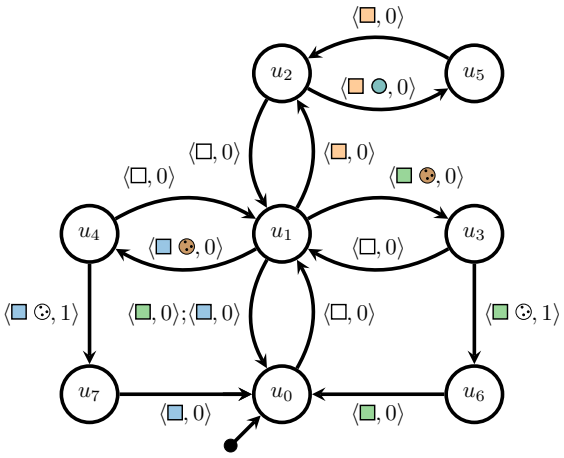
These methods do not work in our domains because our domains are not MDPs.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

C. Methods that solve MDPs with sparse rewards (e.g., DeepSynth)

These methods do not work in our domains because our domains are not MDPs.



Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

Summary

There are three learning objectives for combining automata learning with RL:

A. [LRM] Learn an RM that makes the whole problem Markovian.
B. [JIRP] Learn the smallest DFA that makes the reward function Markovian.
C. [DeepSynth] Learn a high-level model of the environment.

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

Summary

There are three learning objectives for combining automata learning with RL:

A. [LRM] Learn an RM that makes the whole problem Markovian.
B. [JIRP] Learn the smallest DFA that makes the reward function Markovian.
C. [DeepSynth] Learn a high-level model of the environment.

So, what's the right learning objective?

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

Summary

There are three learning objectives for combining automata learning with RL:

A. [LRM] Learn an RM that makes the whole problem Markovian.
B. [JIRP] Learn the smallest DFA that makes the reward function Markovian.
C. [DeepSynth] Learn a high-level model of the environment.

So, what's the right learning objective? ¯_(ツ)_/¯

Frequently asked questions

2) How does LRM relate to other methods that learn automata to aid RL agents?

Summary

There are three learning objectives for combining automata learning with RL:

A. [LRM] Learn an RM that makes the whole problem Markovian.
B. [JIRP] Learn the smallest DFA that makes the reward function Markovian.
C. [DeepSynth] Learn a high-level model of the environment.

So, what's the right learning objective? ¯_(ツ)_/¯

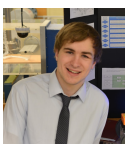
Method	Cookie	Symbol	2-Keys
JIRP	0.6 ± 0.8	-31.1 ± 13.5	2.0 ± 1.4
DeepSynth	0.4 ± 0.6	-30.4 ± 14.0	2.4 ± 1.5
LRM (ours)	**197.2 ± 2.1**	**460.4 ± 15.7**	**86.6 ± 9.4**

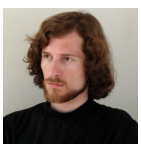
Concluding Remarks

https://bitbucket.org/RToroIcarte/lrm

Thanks! :)


Rodrigo


Ethan


Toryn


Rick


Margarita


Sheila