Learning Reward Machines for Partially Observable Reinforcement Learning

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

UNIVERSITY OF TORONTO

VECTOR INSTITUTE

ELEMENT AI

NeurIPS 2019
December 11

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)

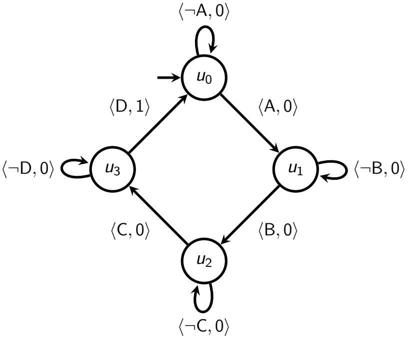
Reward Machines (RMs)

RMs are automata-based reward functions:

Reward Machines (RMs)

RMs are automata-based reward functions:

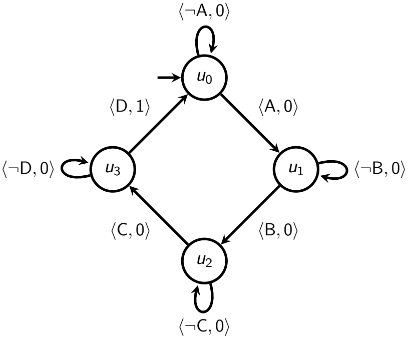
```python
m = 0 # global variable
def get_reward(s):
  if m == 0 and s.at("A"):
    m = 1
  if m == 1 and s.at("B"):
    m = 2
  if m == 2 and s.at("C"):
    m = 3
  if m == 3 and s.at("D"):
    m = 0
    return 1
  return 0
```



Reward Machines (RMs)

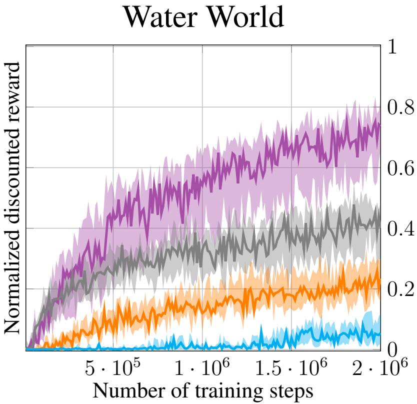
RMs are automata-based reward functions:

```
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
```



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

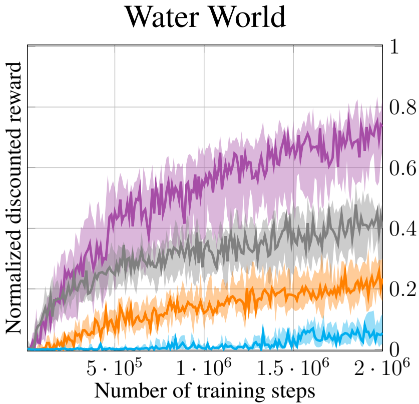
Reward Machines (RMs)



Legend:

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

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.

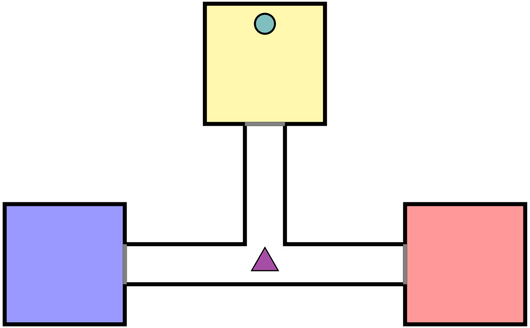
Learning Reward Machines for Partially Observable RL

This work:

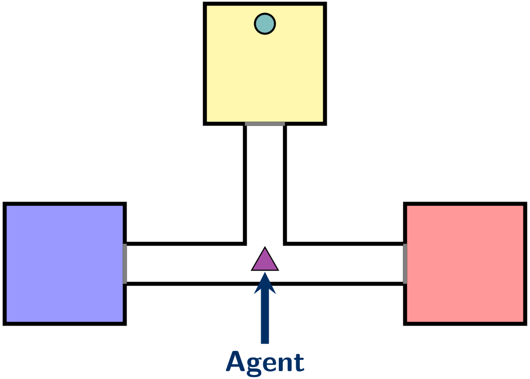
1. Shows how to learn RMs from experiences (LRM).
2. Uses RMs as memory for partially observable RL.
3. Extends QRM to work under partial observability.
4. Provides a theoretical and empirical analysis of LRM.

The Cookie Domain

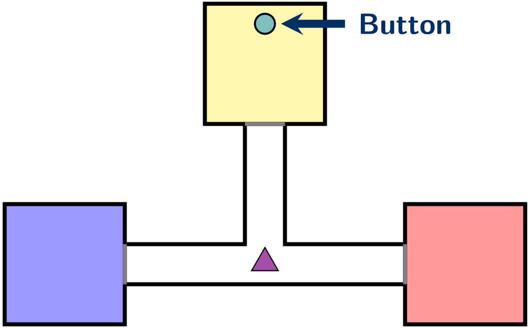
The cookie domain



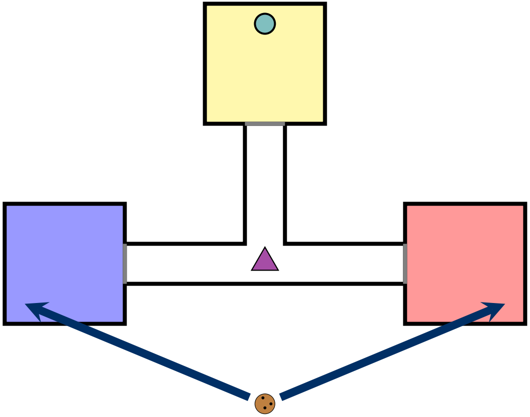
The cookie domain



The cookie domain

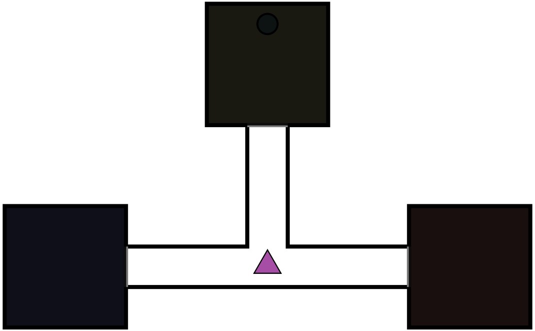


The cookie domain

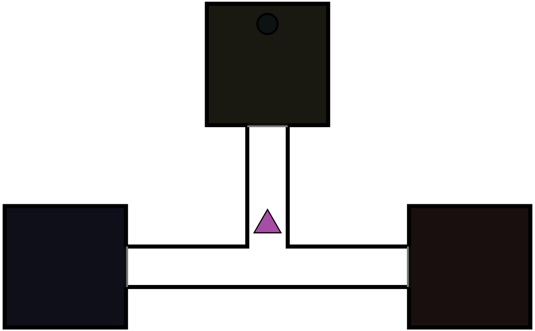


(Cookie)

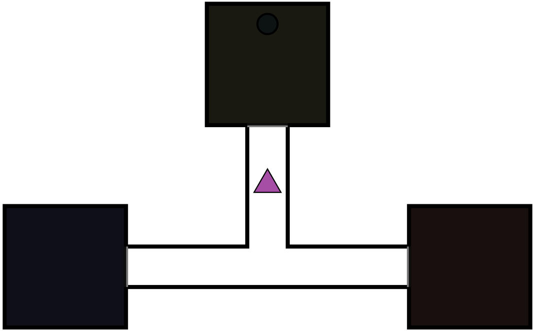
The cookie domain



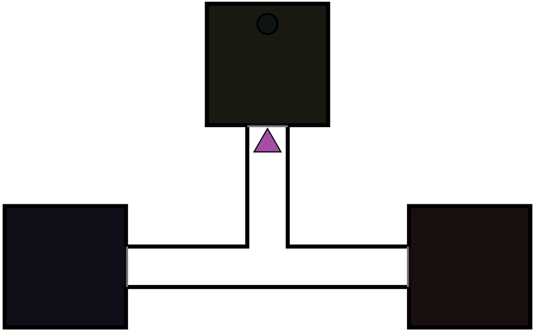
The cookie domain



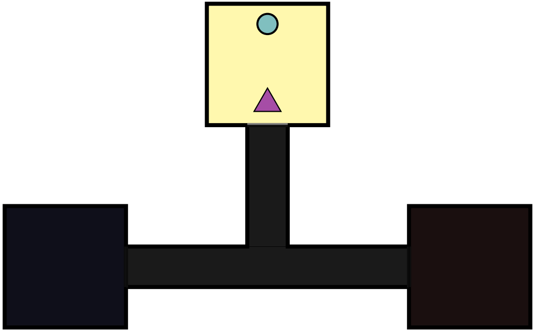
The cookie domain



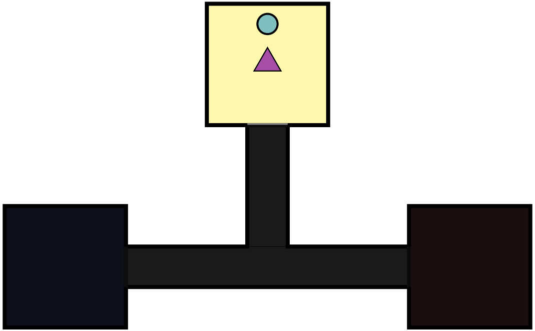
The cookie domain



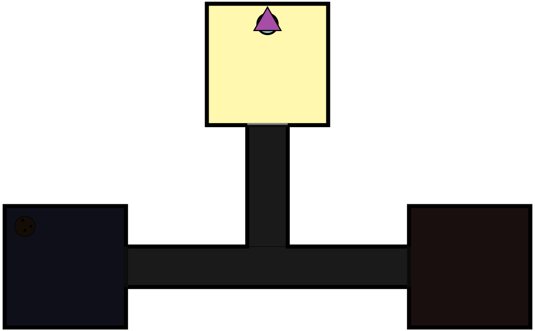
The cookie domain



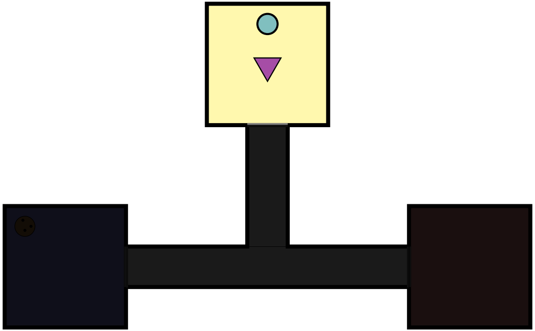
The cookie domain



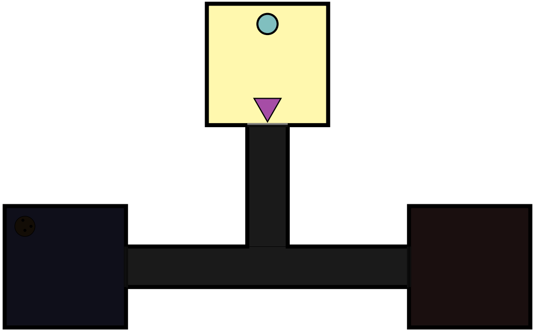
The cookie domain



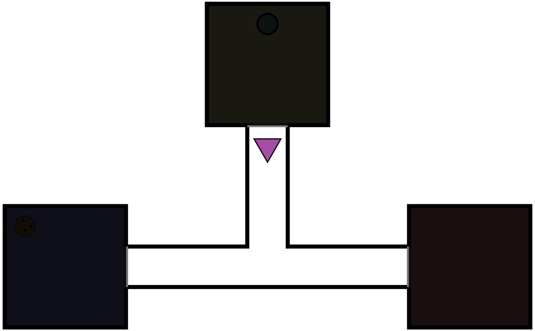
The cookie domain



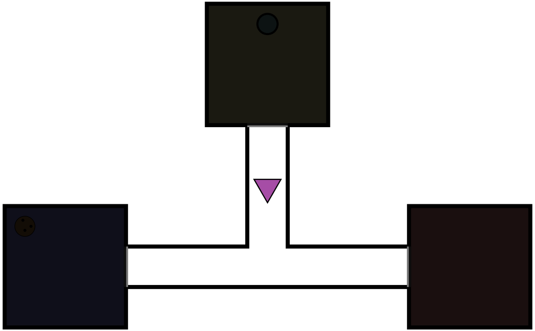
The cookie domain



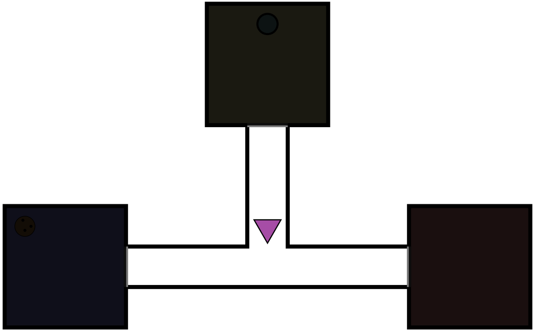
The cookie domain



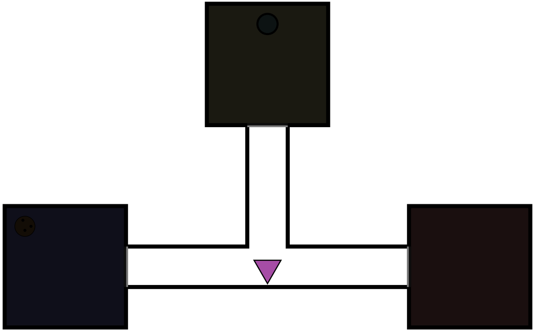
The cookie domain



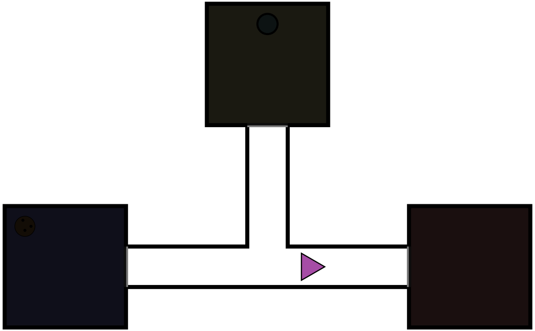
The cookie domain



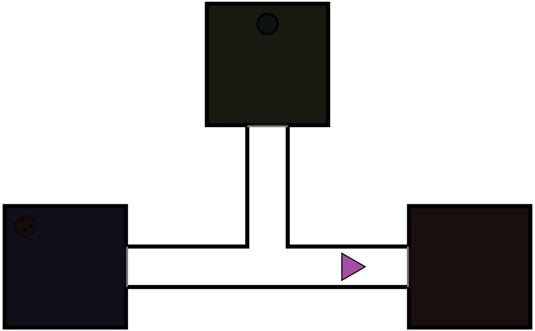
The cookie domain



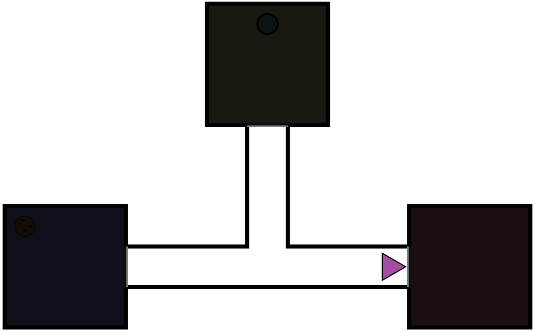
The cookie domain



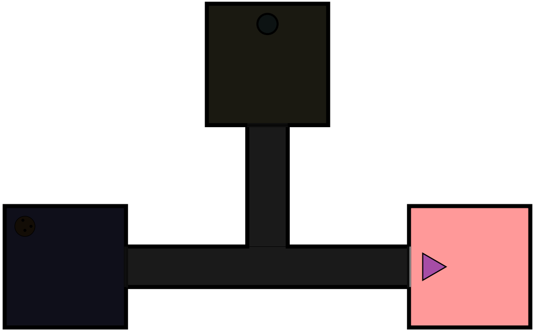
The cookie domain



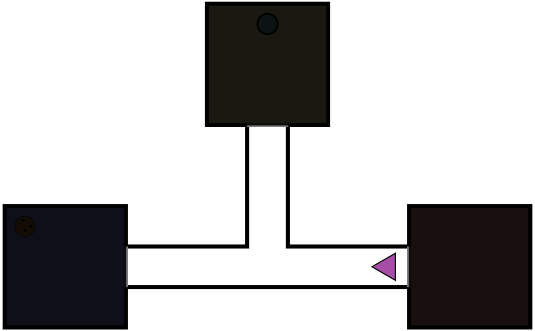
The cookie domain



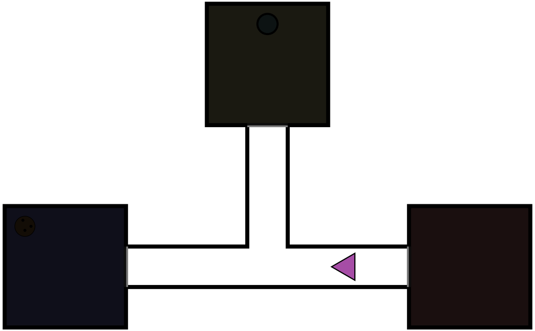
The cookie domain



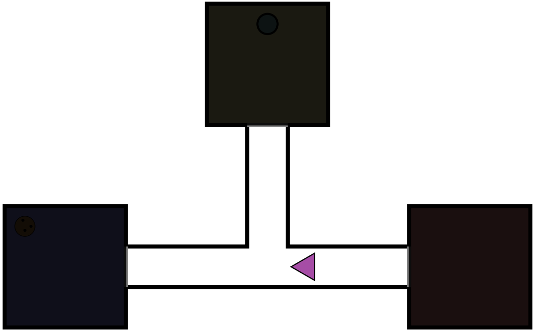
The cookie domain



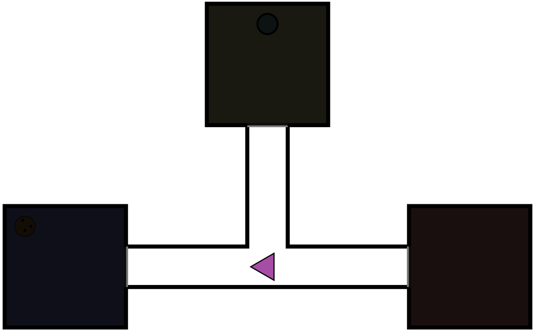
The cookie domain



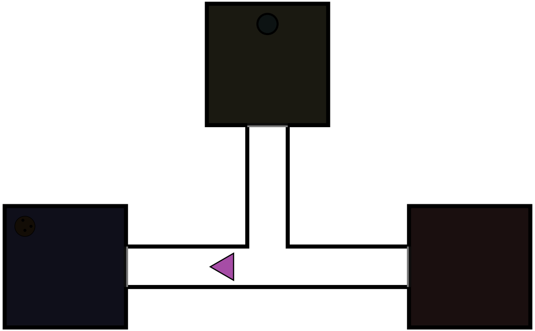
The cookie domain



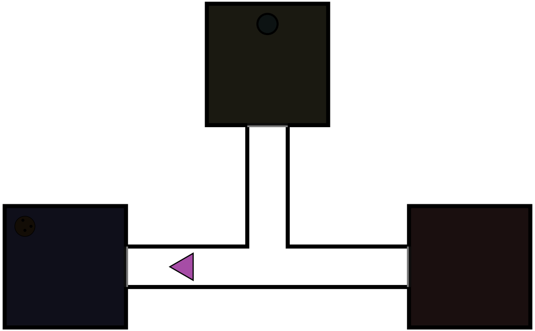
The cookie domain



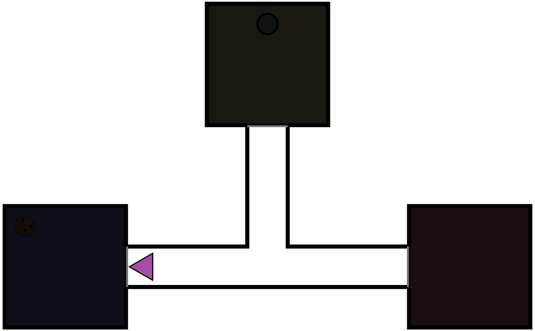
The cookie domain



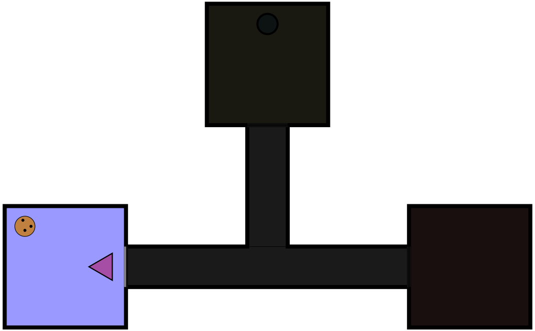
The cookie domain



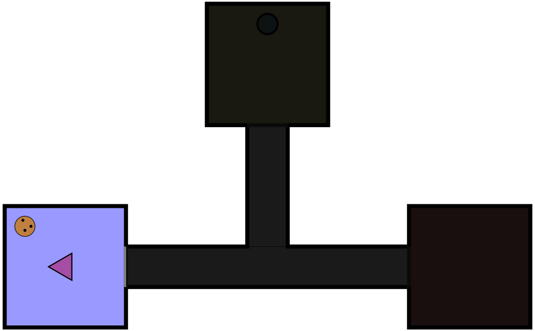
The cookie domain



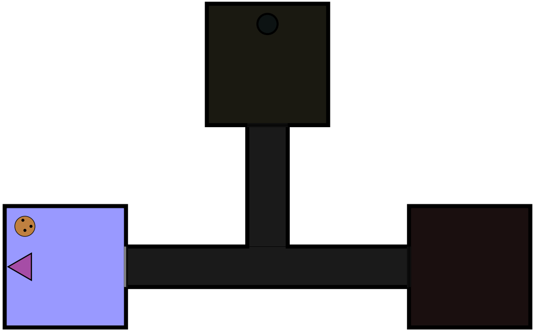
The cookie domain



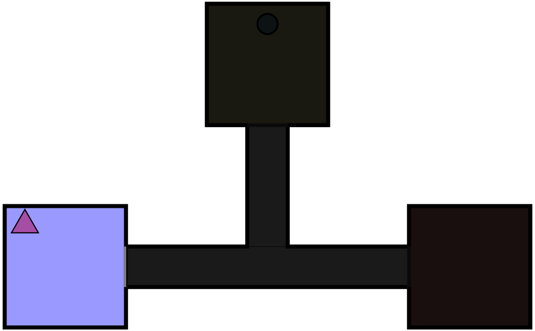
The cookie domain



The cookie domain

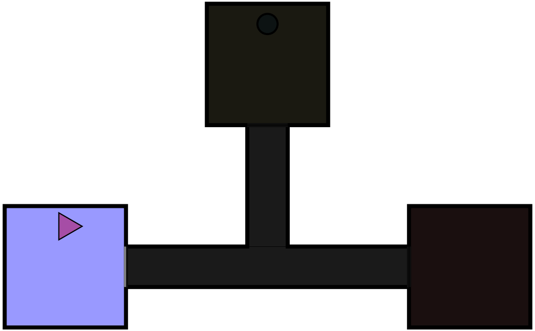


The cookie domain

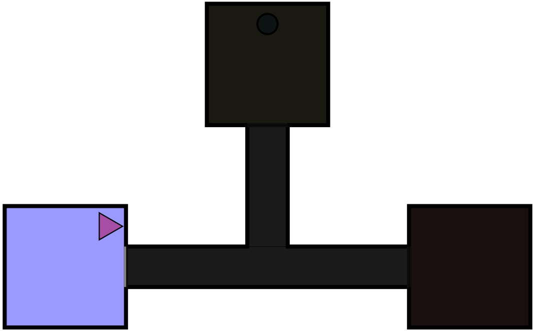


(+1 Reward)

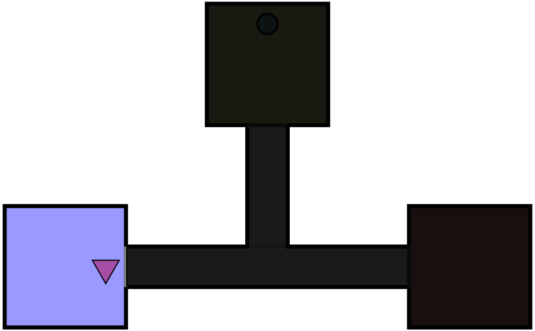
The cookie domain



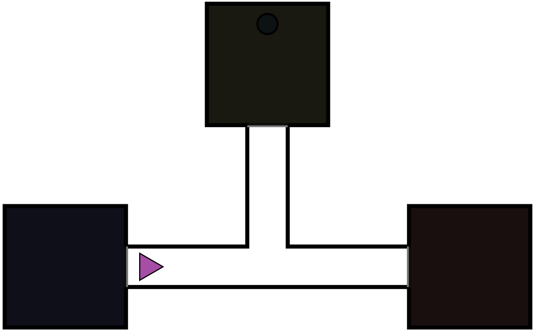
The cookie domain



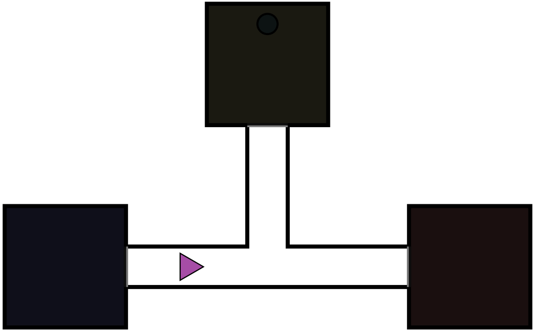
The cookie domain



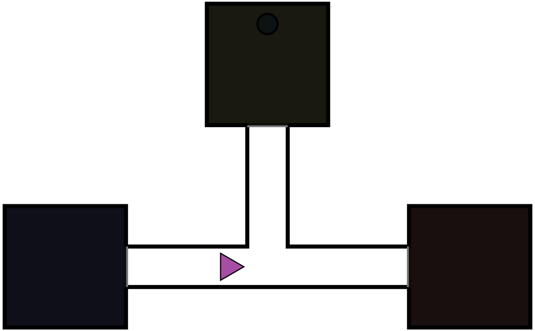
The cookie domain



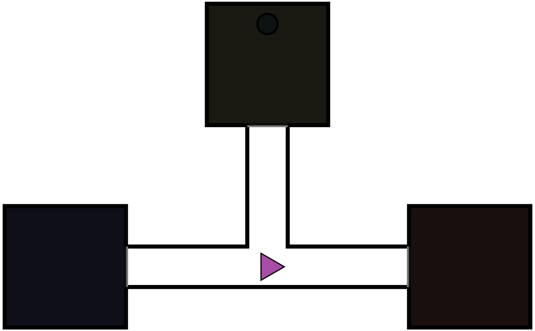
The cookie domain



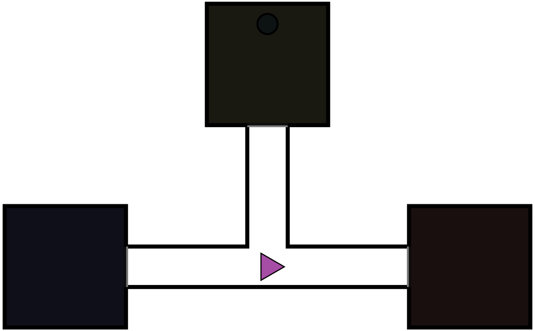
The cookie domain



The cookie domain

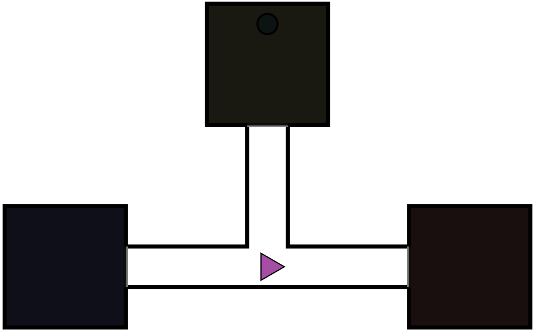


The cookie domain



Solving the cookie domain requires **memory**!

The cookie domain



Solving the cookie domain requires **memory**!

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

Partially Observable RL

The most popular approach:

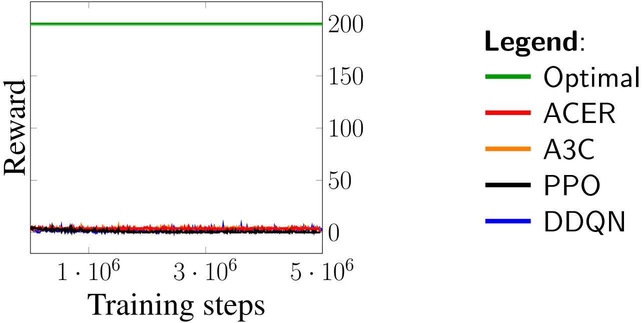
Training LSTMs policies using a policy gradient method.

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

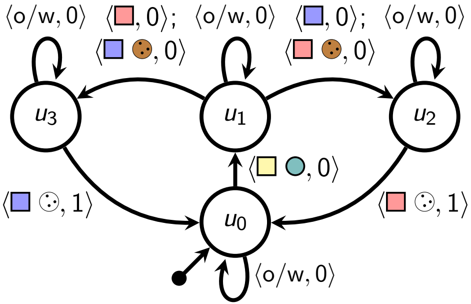
If the agent can detect the color of the rooms (🟥, □, 🟦, 🟨), and when it presses the button (●), eats a cookie (⚆),

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 (🍪),

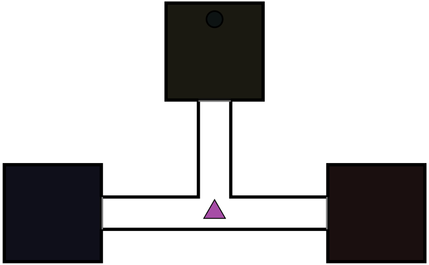
Reward Machines as memory

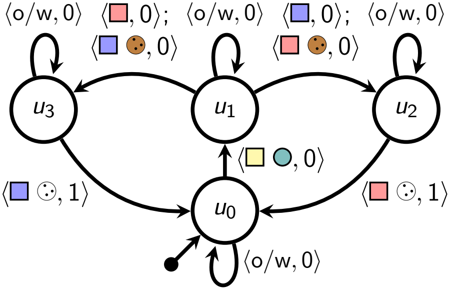
If the agent can detect the color of the rooms (🟥, ⬜, 🟦, 🟨), and when it presses the button (●), eats a cookie (◌), and sees a cookie (🍪), then:



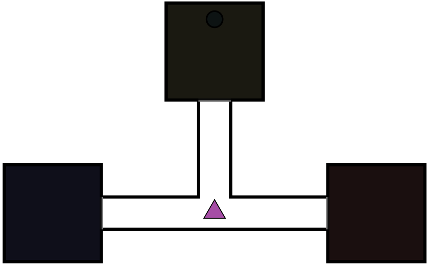
... becomes a **“perfect” memory** for the cookie domain.

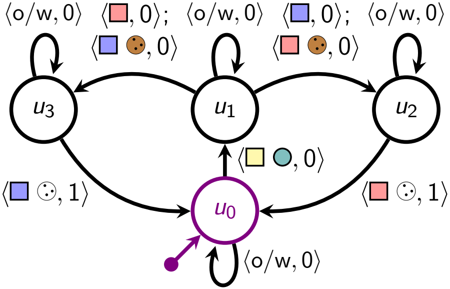
Reward Machines as memory



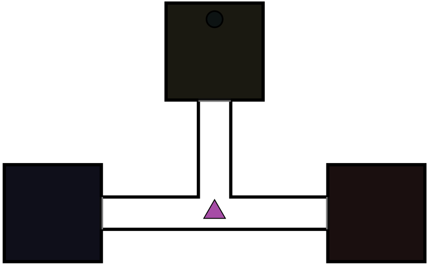


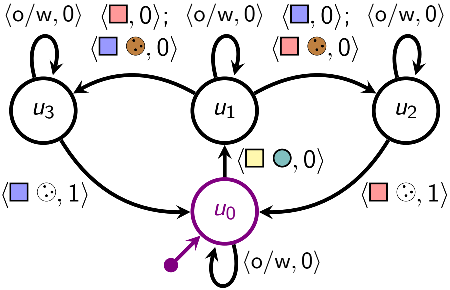
Reward Machines as memory





Reward Machines as memory



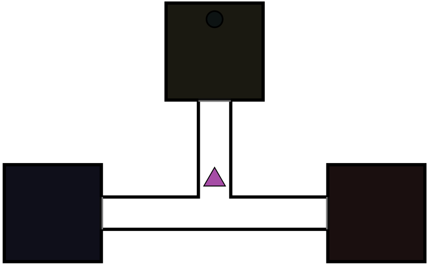


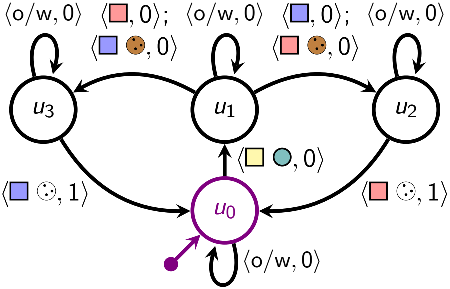
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



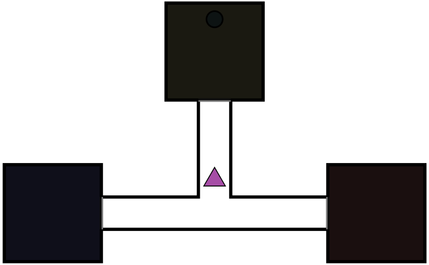


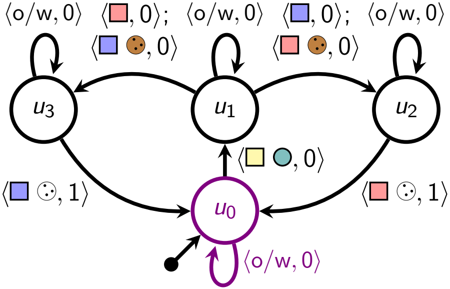
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



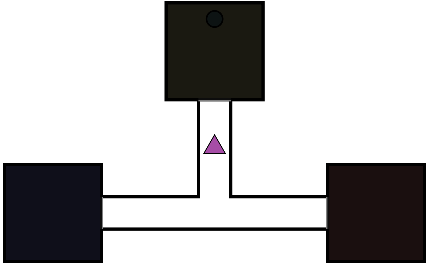


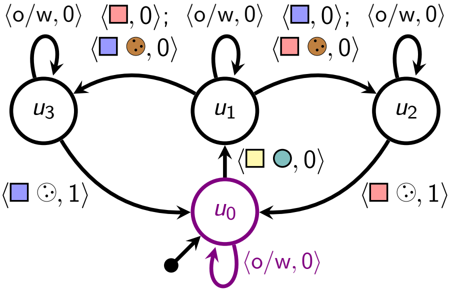
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory

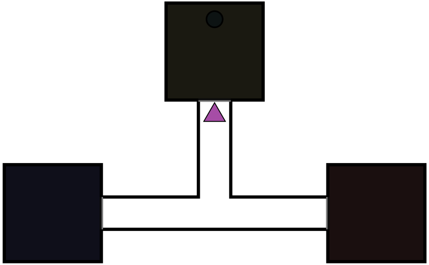


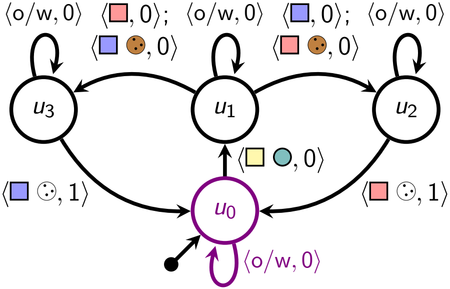


conditions at state u_0

if (🟨 ●)	→	**goto** u_1
else	→	**goto** u_0

Reward Machines as memory



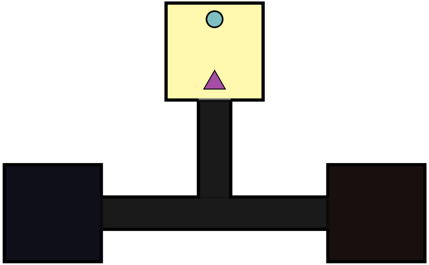


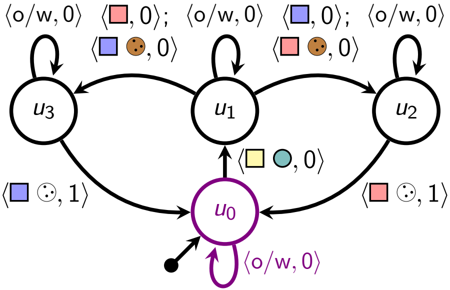
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



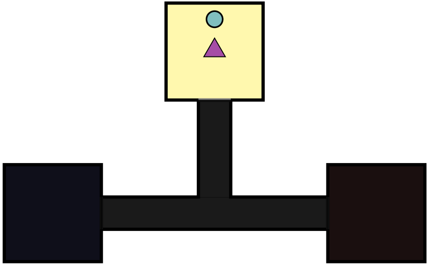


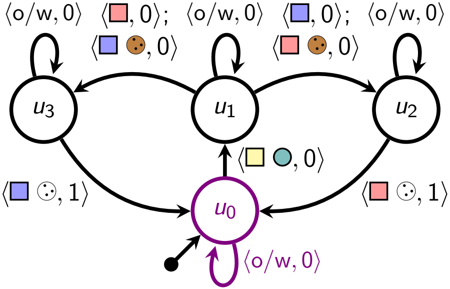
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory

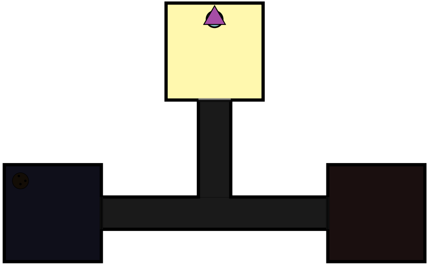


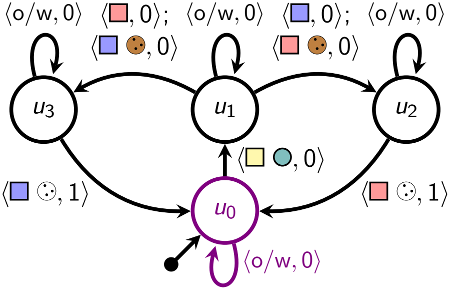


conditions at state u_0

if (🟨 ●)	→	goto u_1
else	→	goto u_0

Reward Machines as memory



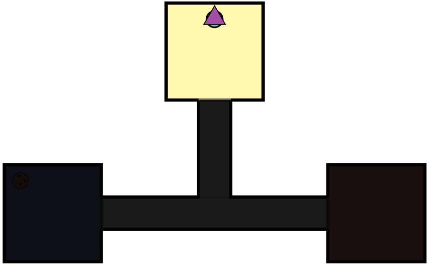


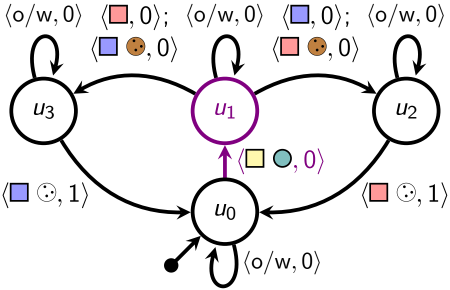
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



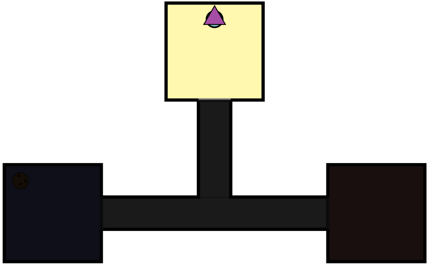


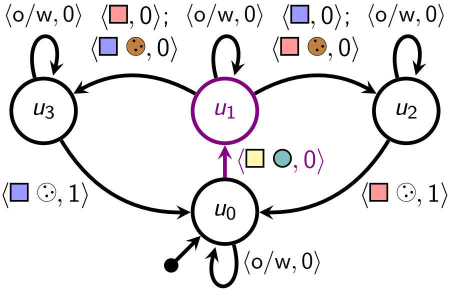
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory

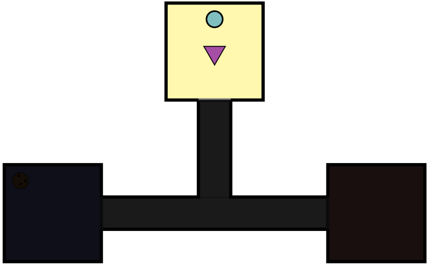


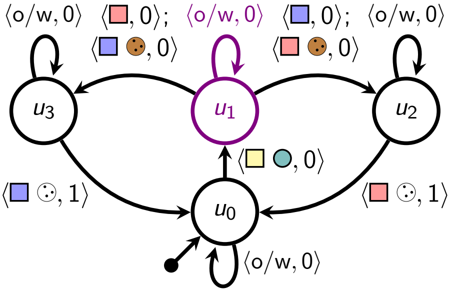


conditions at state u_1

if (🟦 **or** 🟥 🍪) → **goto** u_2
if (🟥 **or** 🟦 🍪) → **goto** u_3
else → **goto** u_1

Reward Machines as memory





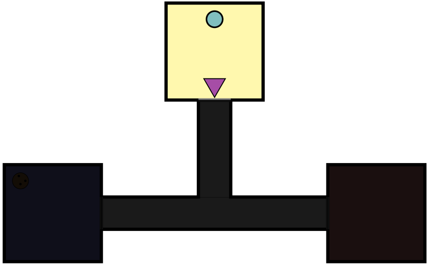
conditions at state u_1

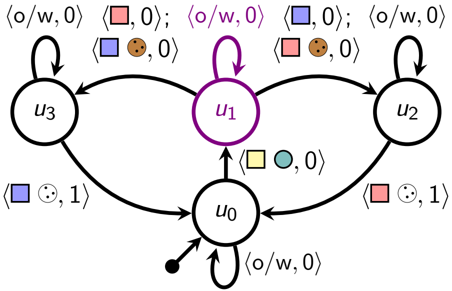
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → goto u_1

Reward Machines as memory

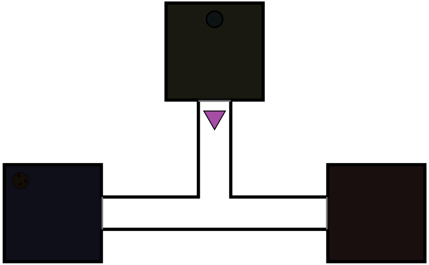


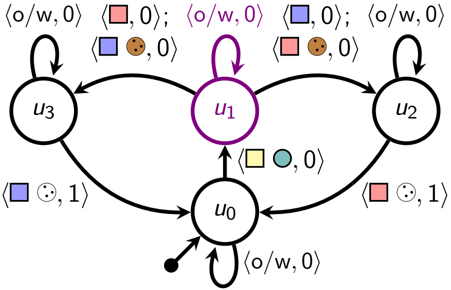


conditions at state u_1

if (🟦 or 🟥 🍪)	→	goto u_2
if (🟥 or 🟦 🍪)	→	goto u_3
else	→	**goto u_1**

Reward Machines as memory

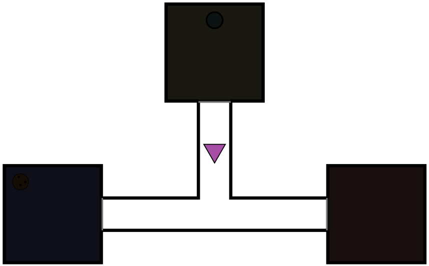


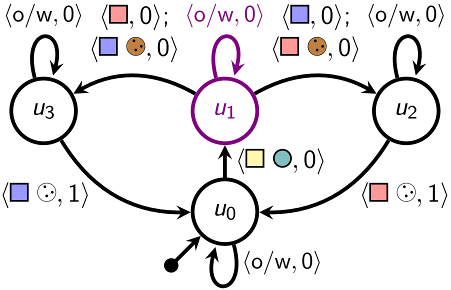


conditions at state u_1

if (🟦 or 🟥 🍪)	→	goto u_2
if (🟥 or 🟦 🍪)	→	goto u_3
else	→	**goto** u_1

Reward Machines as memory





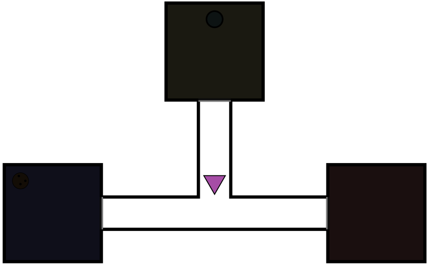
conditions at state u_1

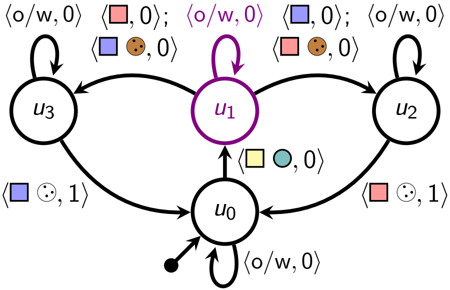
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → **goto** u_1

Reward Machines as memory





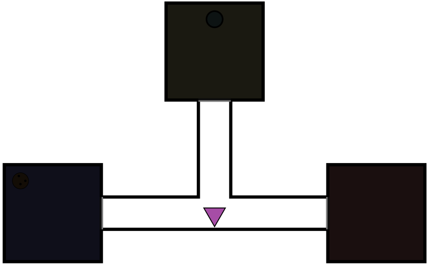
conditions at state u_1

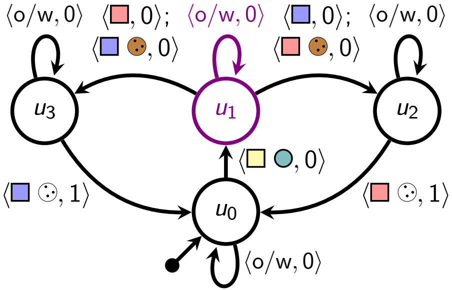
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → **goto** u_1

Reward Machines as memory





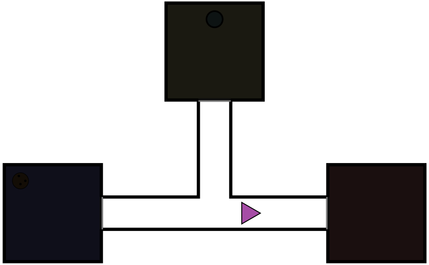
conditions at state u_1

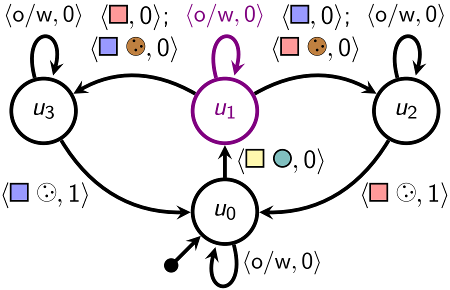
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → **goto** u_1

Reward Machines as memory





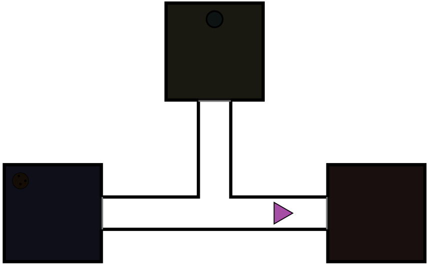
conditions at state u_1

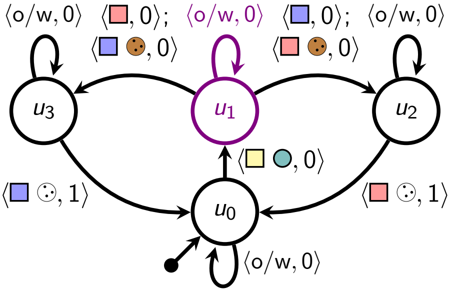
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → **goto** u_1

Reward Machines as memory





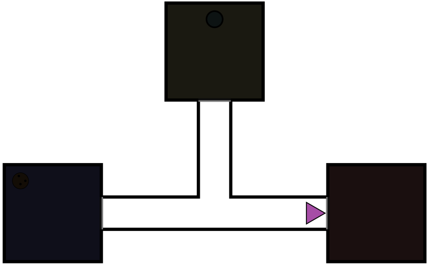
conditions at state u_1

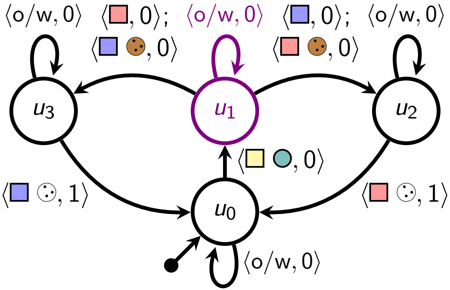
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → goto u_1

Reward Machines as memory





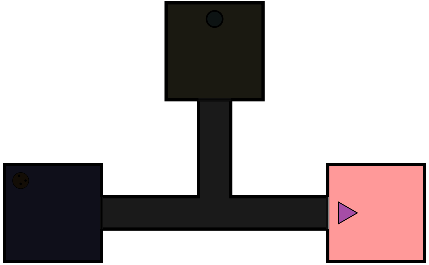
conditions at state u_1

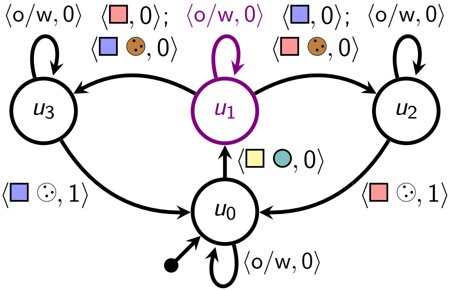
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → **goto** u_1

Reward Machines as memory





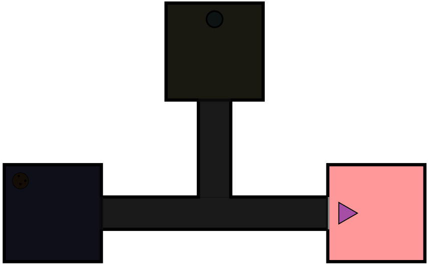
conditions at state u_1

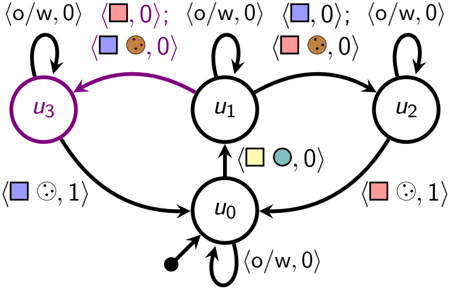
if (🟦 or 🟥 🍪) → goto u_2

if (🟥 or 🟦 🍪) → goto u_3

else → goto u_1

Reward Machines as memory



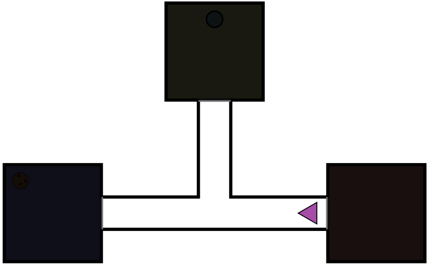


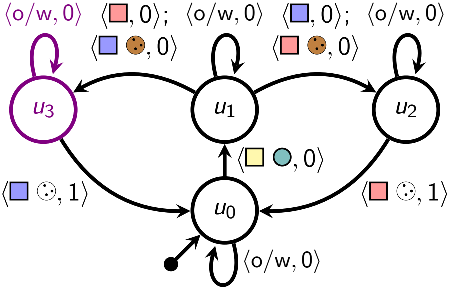
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



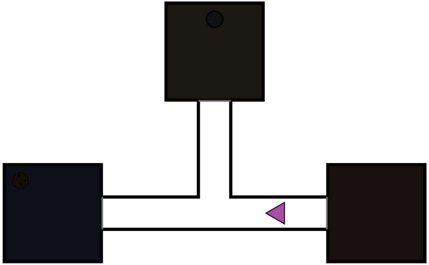


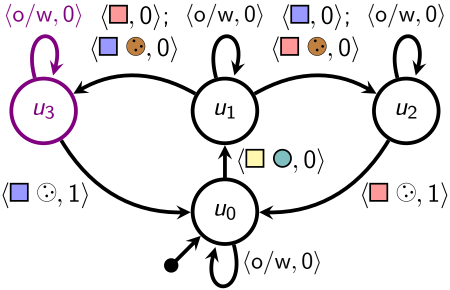
conditions at state u_3

if (🟦 ○) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



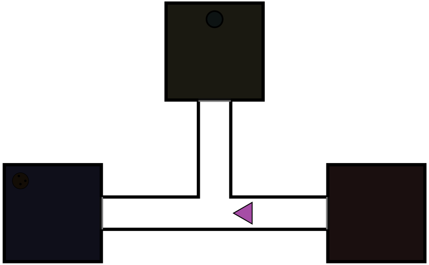


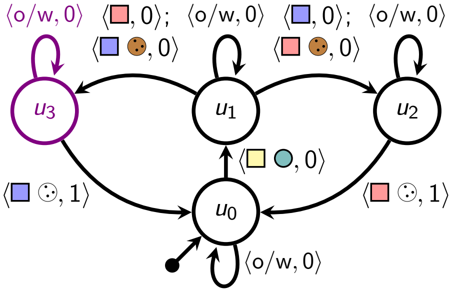
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory

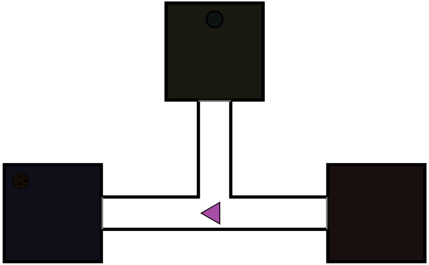


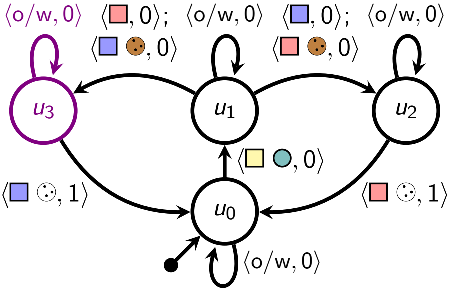


conditions at state u_3

if (🟦 ◌)	→	goto u_0
else	→	goto u_3

Reward Machines as memory



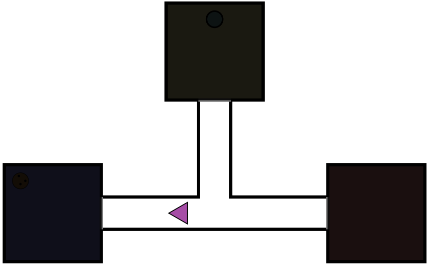


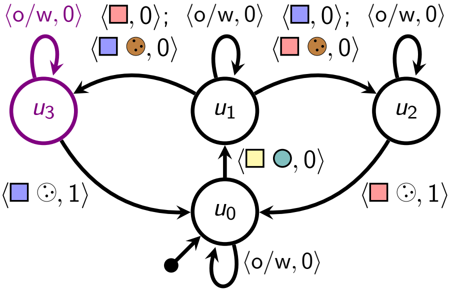
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



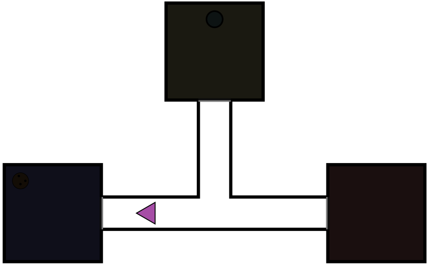


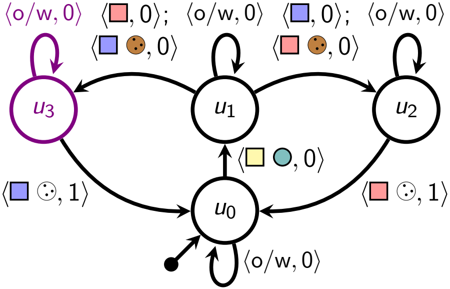
conditions at state u_3

if (🟦 ⛬) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



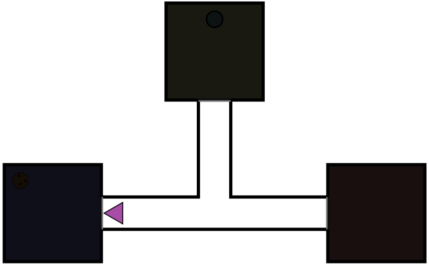


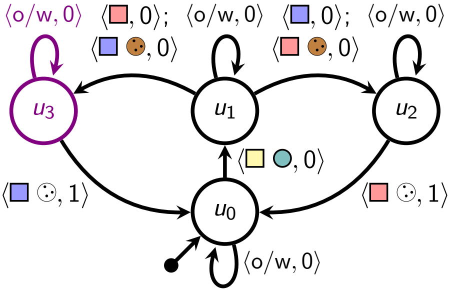
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



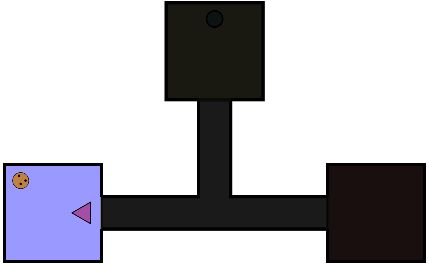


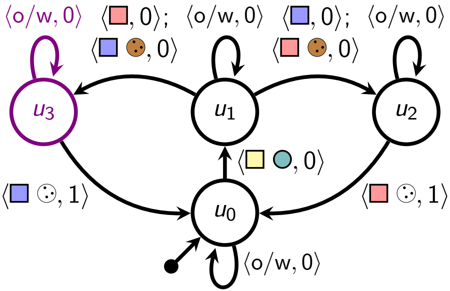
conditions at state u_3

if (🟦 ◌) → goto u_0

else → **goto** u_3

Reward Machines as memory



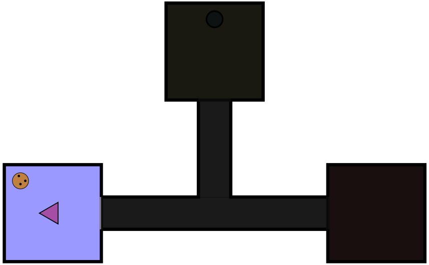


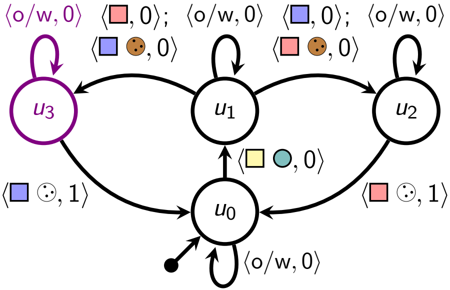
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



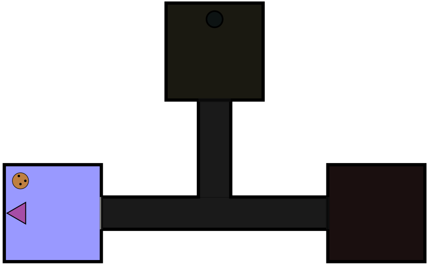


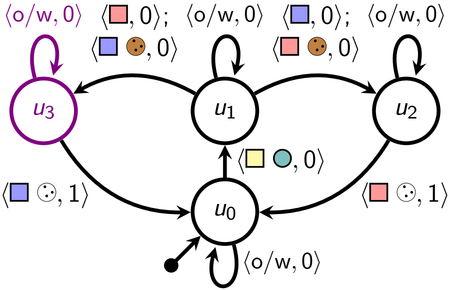
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory

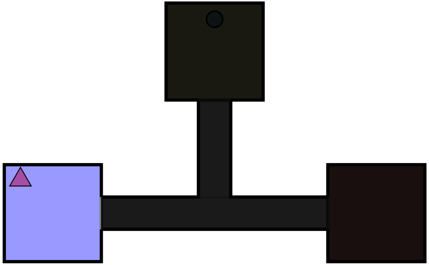


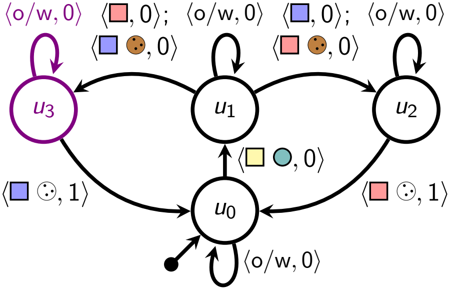


conditions at state u_3

if (🟦 ◌)	→	**goto** u_0
else	→	**goto** u_3

Reward Machines as memory



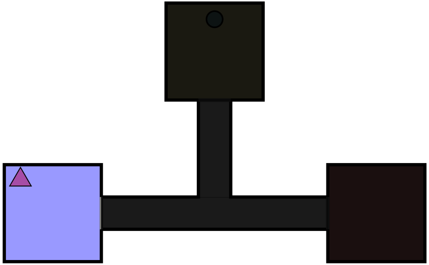


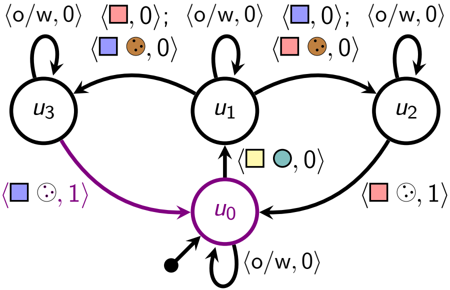
conditions at state u_3

if (🟦 ◌) → **goto** u_0

else → **goto** u_3

Reward Machines as memory



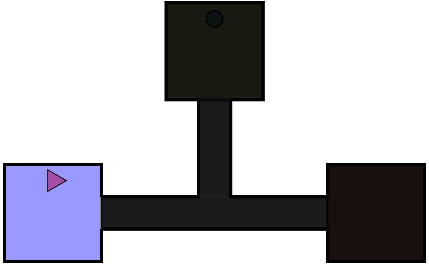


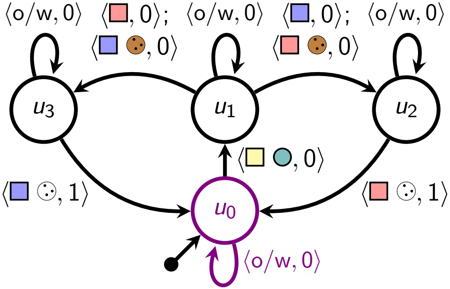
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



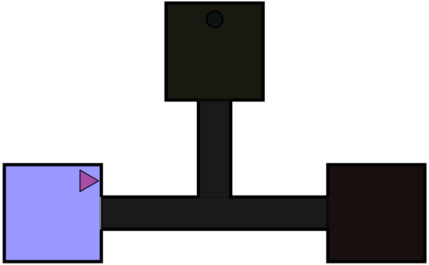


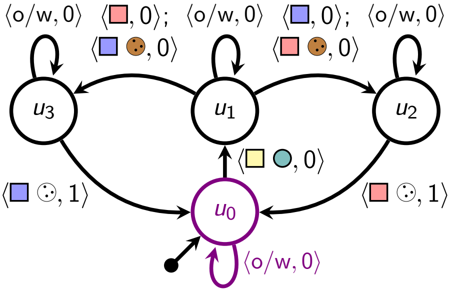
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory

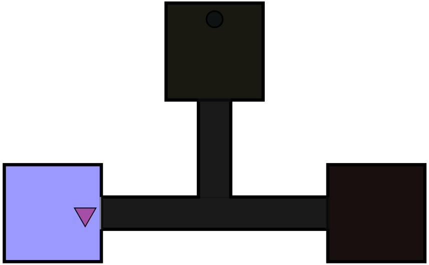


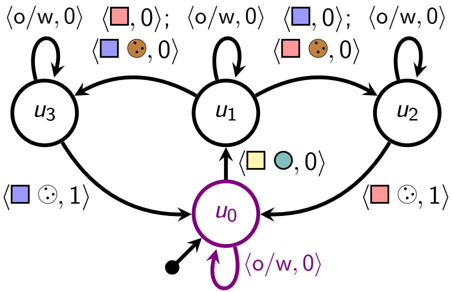


conditions at state u_0

if (🟨 ●)	→	**goto** u_1
else	→	**goto** u_0

Reward Machines as memory



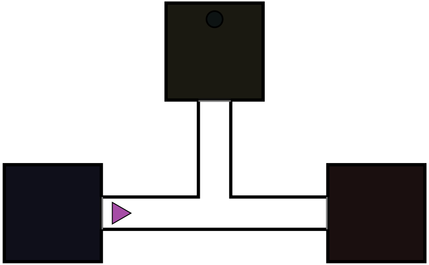


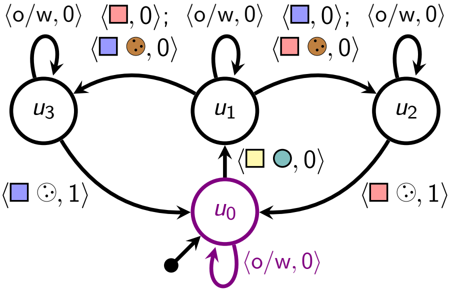
conditions at state u_0

if (🟨 ●) → goto u_1

else → goto u_0

Reward Machines as memory

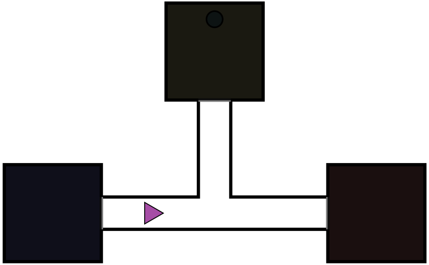


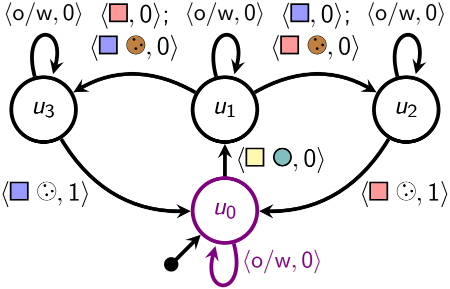


conditions at state u_0

if (🟨 ●)	→	**goto** u_1
else	→	**goto** u_0

Reward Machines as memory



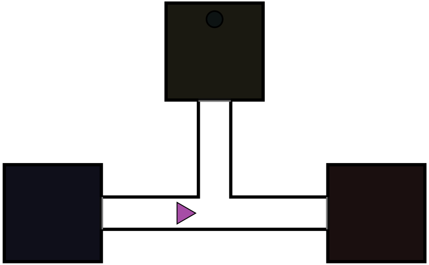


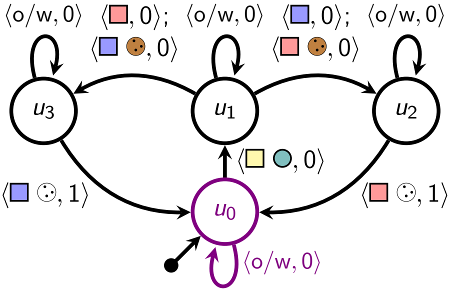
conditions at state u_0

if (🟨 ●) → goto u_1

else → goto u_0

Reward Machines as memory



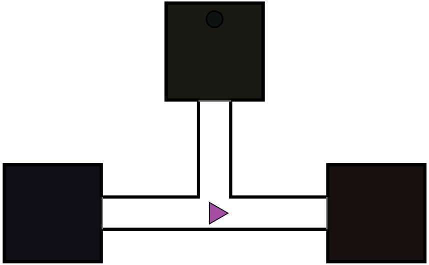


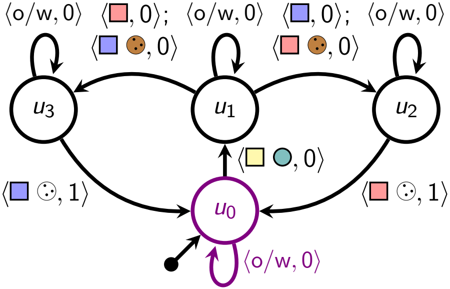
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

Reward Machines as memory



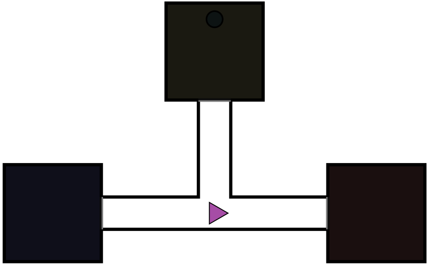


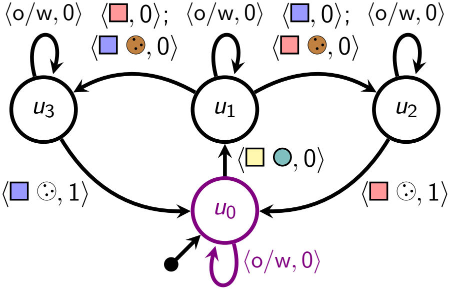
conditions at state u_0

if (🟨 ●) → **goto** u_1

else → **goto** u_0

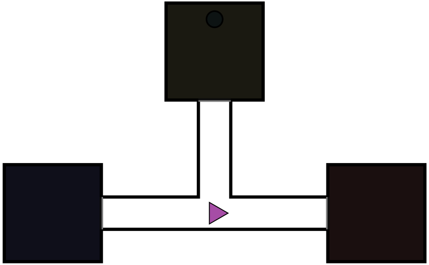
Reward Machines as memory

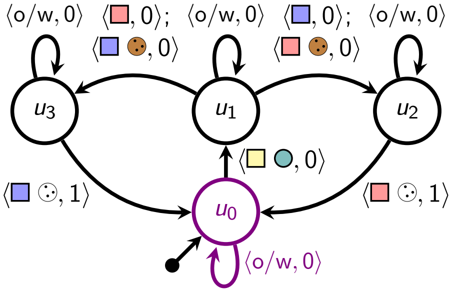




Why is this a perfect memory?

Reward Machines as memory

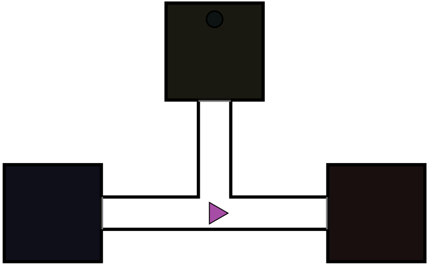


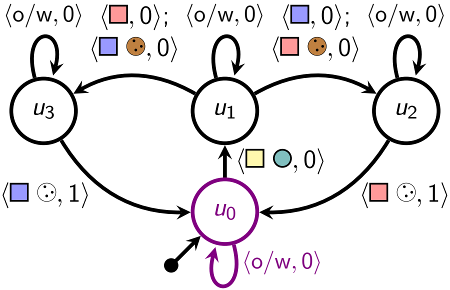


Why is this a perfect memory?

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

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:

Learning Reward Machines

Given a set of detectors (e.g., {🟥, □, 🟦, 🟨, 🔵, 🍪, ⚇}) and traces $\mathcal{T}$, learning RMs is a **discrete optimization** problem:

$$
\begin{aligned}
&\underset{\langle U, u_0, \delta_u, \delta_r \rangle}{\text{minimize}} \quad \sum_{i \in I} \sum_{t \in T_i} \log(|N_{x_{i,t}, L(e_{i,t})}|) && \texttt{(LRM)} \\
&s.t. \ \langle U, u_0, \delta_u, \delta_r \rangle \in \mathcal{R}_{\mathcal{P}} && (1) \\
&|U| \leq u_{\max} && (2) \\
&x_{i,t} \in U \qquad \forall i \in I, t \in T_i \cup \{t_i\} && (3) \\
&x_{i,0} = u_0 \qquad \forall i \in I && (4) \\
&x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) \qquad \forall i \in I, t \in T_i && (5) \\
&N_{u,l} \subseteq 2^{2^{\mathcal{P}}} \qquad \forall u \in U, l \in 2^{\mathcal{P}} && (6) \\
&L(e_{i,t+1}) \in N_{x_{i,t}, L(e_{i,t})} \qquad \forall i \in I, t \in T_i && (7)
\end{aligned}
$$

Learning Reward Machines

Given a set of detectors (e.g., {🟥, □, 🟦, 🟨, ●, 🍪, ○}) and traces $\mathcal{T}$, learning RMs is a **discrete optimization** problem:

$$
\begin{aligned}
&\underset{\langle U, u_0, \delta_u, \delta_r\rangle}{\text{minimize}} \quad \sum_{i\in I}\sum_{t\in T_i} \log(|N_{x_{i,t}, L(e_{i,t})}|) && && \texttt{(LRM)}\\
&s.t. \ \langle U, u_0, \delta_u, \delta_r\rangle \in \mathcal{R}_{\mathcal{P}} && && (1)\\
&|U| \leq u_{\text{max}} && && (2)\\
&x_{i,t} \in U && \forall i \in I, t \in T_i \cup \{t_i\} && (3)\\
&x_{i,0} = u_0 && \forall i \in I && (4)\\
&x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) && \forall i \in I, t \in T_i && (5)\\
&N_{u,l} \subseteq 2^{2^{\mathcal{P}}} && \forall u \in U, l \in 2^{\mathcal{P}} && (6)\\
&L(e_{i,t+1}) \in N_{x_{i,t}, L(e_{i,t})} && \forall i \in I, t \in T_i && (7)
\end{aligned}
$$

Learn a
“*simple*”
causal model

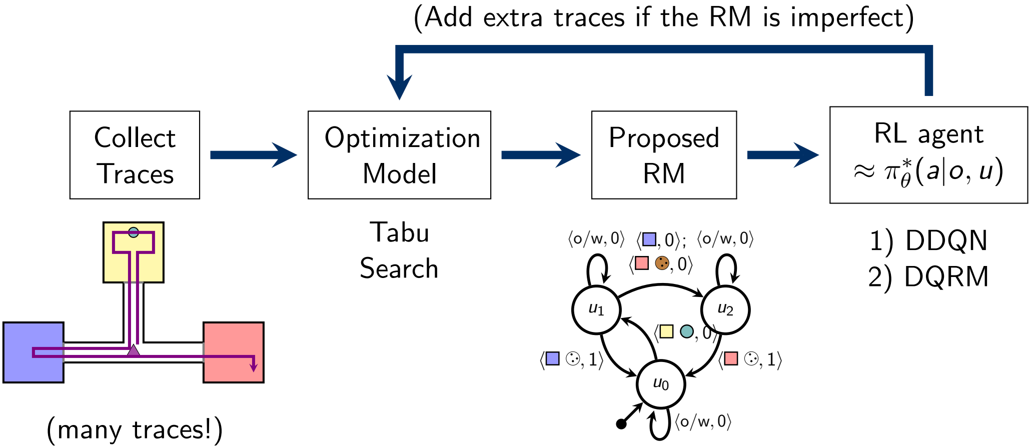
Learning Reward Machines

Given a set of detectors (e.g., {🟥, □, 🟦, 🟨, ●, 🍪, ⚇}) and traces $\mathcal{T}$, learning RMs is a **discrete optimization** problem:

$$
\begin{aligned}
&\underset{\langle U, u_0, \delta_u, \delta_r \rangle}{\text{minimize}} \;\; \sum_{i \in I} \sum_{t \in T_i} \log(|N_{x_{i,t}, L(e_{i,t})}|) && && \texttt{(LRM)} \\
&\quad s.t. \;\; \langle U, u_0, \delta_u, \delta_r \rangle \in \mathcal{R}_{\mathcal{P}} && && (1) \\
&\qquad\quad |U| \leq u_{\max} && && (2) \\
&\qquad\quad x_{i,t} \in U && \forall i \in I, t \in T_i \cup \{t_i\} && (3) \\
&\qquad\quad x_{i,0} = u_0 && \forall i \in I && (4) \\
&\qquad\quad x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) && \forall i \in I, t \in T_i && (5) \\
&\qquad\quad N_{u,l} \subseteq 2^{2^{\mathcal{P}}} && \forall u \in U, l \in 2^{\mathcal{P}} && (6) \\
&\qquad\quad L(e_{i,t+1}) \in N_{x_{i,t}, L(e_{i,t})} && \forall i \in I, t \in T_i && (7)
\end{aligned}
$$

... that we solved using **Tabu Search**.

Overall approach



(Add extra traces if the RM is imperfect)

Collect Traces → Optimization Model → Proposed RM → RL agent $\approx \pi^*_\theta(a|o,u)$

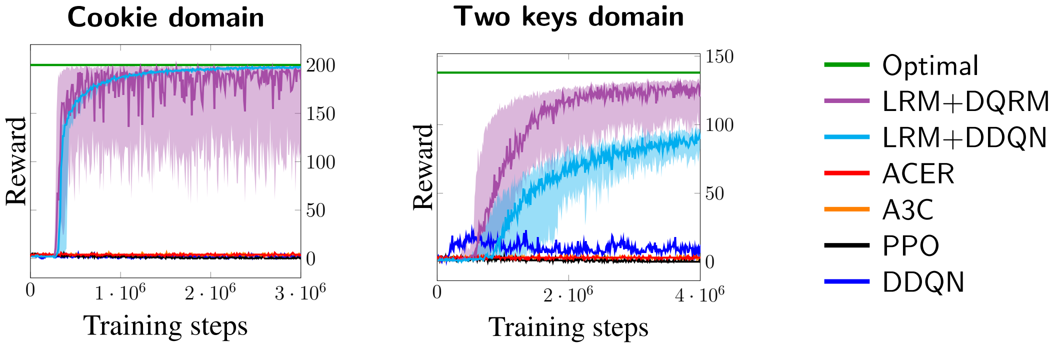
(many traces!)

Tabu Search

1) DDQN
2) DQRM

Results

Results



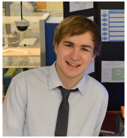
*__Note__: The detectors were also given to the baselines.

Discussion at poster #210

`https://bitbucket.org/RToroIcarte/lrm`

Thanks! :)

Rodrigo



Ethan

Toryn

Rick

Margarita

Sheila