Outline

1. Motivation
2. What is a reward machine (RM)?
3. How to exploit a reward machine’s structure
4. Results
5. Related work
6. Concluding remarks
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
“To summarize, a nice simple idea exposing more of the structure of an RL problem and the benefits thereof.”

— Third reviewer
Reinforcement learning

- Environment
  - Transition Probabilities
  - Reward Function

- RL Agent
  - Policy

The environment might be the real world.

Toro Icarte et al: Using RMs for Task Specification and Decomposition in RL
The environment might be the real world.
Running example
## Running example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>△</td>
<td>Agent</td>
</tr>
<tr>
<td>*</td>
<td>Furniture</td>
</tr>
<tr>
<td>☕</td>
<td>Coffee machine</td>
</tr>
<tr>
<td>✉️</td>
<td>Mail room</td>
</tr>
<tr>
<td>o</td>
<td>Office</td>
</tr>
<tr>
<td>A,B,C,D</td>
<td>Marked locations</td>
</tr>
</tbody>
</table>

![Grid with symbols and meanings](image-url)
Running example

<table>
<thead>
<tr>
<th>B</th>
<th>*</th>
<th>*</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>o</td>
<td>D</td>
<td>*</td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>△</td>
</tr>
</tbody>
</table>

Symbol | Meaning
---|---
△ | Agent
* | Furniture
☕️ | Coffee machine
✉️ | Mail room
o | Office
A,B,C,D | Marked locations

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

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

Someone has to program a reward function
Running example

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>△</td>
<td>Agent</td>
</tr>
<tr>
<td>*</td>
<td>Furniture</td>
</tr>
<tr>
<td>☕</td>
<td>Coffee machine</td>
</tr>
<tr>
<td>⌂</td>
<td>Mail room</td>
</tr>
<tr>
<td>o</td>
<td>Office</td>
</tr>
<tr>
<td>A,B,C,D</td>
<td>Marked locations</td>
</tr>
</tbody>
</table>

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

Someone has to program a reward function (even if the environment is the real world).
**Running example**

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

Someone has to program a reward function (even if the environment is the real world).
**Running example**

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

Someone has to program a reward function (even if the environment is the real world).

```
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 Function**

- `m = 0` and `s.at("A")`:
  - `m = 1`
- `m = 1` and `s.at("B")`:
  - `m = 2`
- `m = 2` and `s.at("C")`:
  - `m = 3`
- `m = 3` and `s.at("D")`:
  - `m = 0`
  - return 1
- return 0

Running example

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

Someone has to program a reward function (even if the environment is the real world).
Running example

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

Someone has to program a reward function (even if the environment is the real world).

Reward Function:

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

Someone has to program a reward function (even if the environment is the real world).
Running example

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

Someone has to program a reward function (even if the environment is the real world).

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

Someone has to program a reward function (even if the environment is the real world).
**Task:** Patrol A, B, C, and D.

Someone has to program a reward function (even if the environment is the real world).
**Task**: Patrol A, B, C, and D.

Someone has to program a reward function (even if the environment is the real world).
What if we give the agent access to the reward function?
What if we give the agent access to the reward function?
What if we give the agent access to the reward function?

Is there any advantage of doing do?
What if we give the agent access to the reward function?

Is there any advantage of doing do?
The agent can exploit the reward structure!
The simple idea

Environment
- Transition Probabilities
- Reward Function

RL Agent
- Policy

action

reward

state
The simple idea

Environment

Transition Probabilities

RL Agent

Policy

Reward Function

Action

State

Reward
The simple idea

Environment

Transition Probabilities

RL Agent

Policy

Reward Function

How to exploit the reward function definition
The simple idea

How to exploit the reward function definition

1. **RMs**: A novel language to define reward functions.
2. **QRM**: An RL algorithm that exploits RM’s structure.
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
We encode reward functions using a finite state machine.
We encode reward functions using a finite state machine.

\begin{verbatim}
  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
\end{verbatim}
A reward machine

A finite set of states $U$

An initial state $u_0 \in U$

A set of transitions labelled by:
- A logical condition and
- A reward.

Graph:
- $u_0$ to $u_1$: $\langle \neg A, 0 \rangle$
- $u_0$ to $u_2$: $\langle D, 1 \rangle$
- $u_0$ to $u_3$: $\langle \neg D, 0 \rangle$
- $u_1$ to $u_0$: $\langle A, 0 \rangle$
- $u_1$ to $u_2$: $\langle B, 0 \rangle$
- $u_1$ to $u_3$: $\langle \neg B, 0 \rangle$
- $u_2$ to $u_0$: $\langle C, 0 \rangle$
- $u_2$ to $u_1$: $\langle \neg C, 0 \rangle$
- $u_3$ to $u_0$: $\langle \neg A, 0 \rangle$
- $u_3$ to $u_1$: $\langle A, 0 \rangle$
- $u_3$ to $u_2$: $\langle C, 0 \rangle$
- $u_3$ to $u_3$: $\langle \neg C, 0 \rangle$
A reward machine

A finite set of states $U$
A reward machine

A finite set of states $U$
A reward machine

A finite set of states $U$
An initial state $u_0 \in U$
A reward machine

A finite set of states $U$

An initial state $u_0 \in U$
A reward machine

A finite set of states \( U \)
An initial state \( u_0 \in U \)
A set of transitions labelled by:

\[
\begin{align*}
\langle A, 0 \rangle & \quad \langle -A, 0 \rangle \\
\langle B, 0 \rangle & \quad \langle -B, 0 \rangle \\
\langle C, 0 \rangle & \quad \langle -C, 0 \rangle \\
\langle D, 1 \rangle & \quad \langle -D, 0 \rangle
\end{align*}
\]
A reward machine

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

- $\langle A, 0 \rangle$
- $\langle -A, 0 \rangle$
- $\langle B, 0 \rangle$
- $\langle -B, 0 \rangle$
- $\langle C, 0 \rangle$
- $\langle -C, 0 \rangle$
- $\langle D, 1 \rangle$
- $\langle -D, 0 \rangle$
A reward machine

A finite set of states $U$

An initial state $u_0 \in U$

A set of transitions labelled by:
- a logical condition and
Reward machines

A reward machine

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

- a logical condition and
A reward machine

A finite set of states $U$
An initial state $u_0 \in U$
A set of transitions labelled by:
- a logical condition and

Conditions are over properties of the current state:

$$\mathcal{P} = \{\text{\ding{85}}, \text{\ding{86}}, o, *, A, B, C, D\}$$
A reward machine

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

Conditions are over properties of the current state:

$$\mathcal{P} = \{\text{☕️}, \text{📖}, o, \ast, A, B, C, D\}$$
A **simple** reward machine

A finite set of states $U$
An initial state $u_0 \in U$
A set of transitions labelled by:
- a logical condition and
- a reward (constant number).

Conditions are over properties of the current state:

$$\mathcal{P} = \{\text{☕, ☕, 0, ⋄, A, B, C, D}\}$$
A simple reward machine

A finite set of states $U$
An initial state $u_0 \in U$
A set of transitions labelled by:
- a logical condition and
- a reward (constant number).

Conditions are over properties of the current state:

$$\mathcal{P} = \{\text{☕, ☕, o, ⋆, A, B, C, D}\}$$
Reward machines in action

\[
\langle \neg A, 0 \rangle \quad \langle A, 0 \rangle
\]

\[
\langle \neg D, 0 \rangle \quad \langle D, 1 \rangle
\]

\[
\langle \neg C, 0 \rangle \quad \langle C, 0 \rangle
\]

\[
\langle \neg B, 0 \rangle \quad \langle B, 0 \rangle
\]
Reward machines in action

\[
\begin{array}{cccc}
B & * & * & C \\
* & O & \text{\ding{172}} & * \\
A & * & * & D \\
\end{array}
\]

\[
\langle \neg A, 0 \rangle \quad \langle D, 1 \rangle \\
\langle A, 0 \rangle \quad \langle \neg D, 0 \rangle \\
\langle C, 0 \rangle \quad \langle B, 0 \rangle \\
\langle \neg C, 0 \rangle \quad \langle \neg B, 0 \rangle
\]
Reward machines in action
Reward machines in action
Reward machines in action
Reward machines in action
## Reward machines in action

### Grid Layout

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>O</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

- Initial state: $u_0$
- Transitions:
  - $(D, 1) ightarrow u_1$
  - $(A, 0) ightarrow u_0$
  - $(−D, 0) ightarrow u_3$
  - $(B, 0) ightarrow u_2$
  - $(C, 0) ightarrow u_2$
  - $(−C, 0) ightarrow u_2$
  - $(−B, 0) ightarrow u_1$
  - $(A, 0) ightarrow u_0$
  - $(−B, 0) ightarrow u_1$
  - $(D, 1) ightarrow u_1$

- Reward transitions:
  - $(−A, 0) ightarrow u_0$ (coffee change)
  - $(−A, 0) ightarrow u_0$ (coffee change)
  - $(−A, 0) ightarrow u_0$ (coffee change)
  - $(−A, 0) ightarrow u_0$ (coffee change)
  - $(−A, 0) ightarrow u_0$ (coffee change)
  - $(−A, 0) ightarrow u_0$ (coffee change)
Reward machines in action
Reward machines in action
Reward machines in action
Reward machines in action
Reward machines in action

\[(\neg A, 0) \quad (A, 0) \quad (D, 1) \quad (\neg D, 0) \quad (\neg D, 0) \quad (C, 0) \quad (B, 0) \quad (\neg C, 0) \quad \langle\neg B, 0\rangle\]
Reward machines in action
Reward machines in action
Reward machines in action
Reward machines in action

\[\langle A, 0 \rangle, \langle \neg A, 0 \rangle, \langle B, 0 \rangle, \langle \neg B, 0 \rangle, \langle C, 0 \rangle, \langle \neg C, 0 \rangle, \langle D, 1 \rangle, \langle \neg D, 0 \rangle, \langle A, 0 \rangle\]
Reward machines in action

\[
\begin{array}{ccccc}
B & * & * & C \\
* & O & * & * \\
A & * & * & D \\
\end{array}
\]

\[
\begin{array}{ccccc}
 & & & & \\
 & & & & \\
 & & & & \\
\end{array}
\]

\[
\langle A, 0 \rangle \quad \langle -A, 0 \rangle \\
\langle D, 1 \rangle \quad \langle A, 0 \rangle \\
\langle -D, 0 \rangle \quad \langle B, 0 \rangle \\
\langle C, 0 \rangle \quad \langle -B, 0 \rangle \\
\langle -C, 0 \rangle
\]

[Diagram of a reward machine with states and transitions]
Reward machines in action

\[ \langle A, 0 \rangle, \langle \neg A, 0 \rangle, \langle B, 0 \rangle, \langle \neg B, 0 \rangle, \langle C, 0 \rangle, \langle \neg C, 0 \rangle, \langle D, 1 \rangle, \langle \neg D, 0 \rangle \]
Reward machines in action

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>*</th>
<th>*</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td>*</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

* Toro Icarte et al: Using RM for Task Specification and Decomposition in RL 9 / 33
Reward machines in action

\[
\begin{array}{cccc}
\mathcal{A} & * & * & \mathcal{C} \\
* & \mathcal{O} & \mathcal{E} & * \\
\mathcal{A} & * & * & \mathcal{D} \\
\end{array}
\]

\[
\begin{array}{c}
\langle \neg \mathcal{A}, 0 \rangle \\
\langle \mathcal{D}, 1 \rangle \\
\langle \neg \mathcal{D}, 0 \rangle \\
\langle \neg \mathcal{C}, 0 \rangle \\
\langle \mathcal{C}, 0 \rangle \\
\langle \mathcal{B}, 0 \rangle \\
\langle A, 0 \rangle \\
\langle \neg \mathcal{B}, 0 \rangle \\
\end{array}
\]
Reward machines in action
Reward machines in action

\[
\langle A, 0 \rangle \\
\langle \neg A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle \neg B, 0 \rangle \\
\langle C, 0 \rangle \\
\langle \neg C, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg D, 0 \rangle \\
\langle \neg A, 0 \rangle \\
\langle A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle C, 0 \rangle
\]
Reward machines in action

\[ \langle A, 0 \rangle \rightarrow \langle \neg A, 0 \rangle \]
\[ \langle B, 0 \rangle \rightarrow \langle \neg B, 0 \rangle \]
\[ \langle C, 0 \rangle \rightarrow \langle \neg C, 0 \rangle \]
\[ \langle D, 0 \rangle \rightarrow \langle \neg D, 0 \rangle \]
\[ \langle A, 1 \rangle \rightarrow \langle \neg A, 0 \rangle \]
Reward machines in action

```
+---+---+---+---+
|   |   |   | C |
+---+---+---+---+
|   | O |   |   |
+---+---+---+---+
| A |   |   | D |
+---+---+---+---+
```

```
\begin{align*}
\langle A, 0 \rangle &\quad \langle \neg A, 0 \rangle \\
\langle D, 0 \rangle &\quad \langle \neg D, 0 \rangle \\
\langle C, 0 \rangle &\quad \langle \neg C, 0 \rangle \\
\langle B, 0 \rangle &\quad \langle \neg B, 0 \rangle \\
\langle A, 0 \rangle &\quad \langle D, 1 \rangle
\end{align*}
```

Toro Icarte et al: Using RMs for Task Specification and Decomposition in RL
Reward machines in action
Reward machines in action

Diagram showing a reward machine and a grid with symbols for states and transitions.
Reward machines in action

\[
\langle A, 0 \rangle \rightarrow \langle \neg A, 0 \rangle \rightarrow \langle D, 1 \rangle \rightarrow \langle \neg D, 0 \rangle \rightarrow \langle C, 0 \rangle \rightarrow \langle B, 0 \rangle \rightarrow \langle \neg B, 0 \rangle \rightarrow \langle \neg C, 0 \rangle \rightarrow \langle C, 0 \rangle \rightarrow \langle B, 0 \rangle \rightarrow \langle \neg B, 0 \rangle \rightarrow \langle \neg D, 0 \rangle \rightarrow \langle D, 1 \rangle \rightarrow \langle A, 0 \rangle
\]
Reward machines in action

\[ (\neg A, 0) \quad (A, 0) \quad (D, 1) \quad (B, 0) \quad (C, 0) \quad (\neg B, 0) \quad (\neg D, 0) \quad (\neg C, 0) \]
### Reward machines in action

#### Table Representation

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td>*</td>
<td>O</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>D</td>
</tr>
</tbody>
</table>

#### Diagram Representation

![Diagram](attachment:image.png)
Reward machines in action

\[
\langle A, 0 \rangle \\
\langle \neg A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle \neg B, 0 \rangle \\
\langle C, 0 \rangle \\
\langle \neg C, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg D, 0 \rangle
\]
Reward machines in action

\[
\begin{align*}
B & \quad * & \quad * & \quad \rightarrow & \quad C \\
* & \quad O & \quad \rightarrow & \quad * \\
A & \quad * & \quad * & \quad D
\end{align*}
\]
Reward machines in action

\[
\langle A, 0 \rangle \quad \langle \neg A, 0 \rangle \\
\langle B, 0 \rangle \quad \langle \neg B, 0 \rangle \\
\langle C, 0 \rangle \quad \langle \neg C, 0 \rangle \\
\langle D, 1 \rangle \quad \langle A, 0 \rangle
\]
Reward machines in action

\[
\begin{align*}
&\langle A, 0 \rangle \\
&\langle \neg A, 0 \rangle \\
&\langle B, 0 \rangle \\
&\langle \neg B, 0 \rangle \\
&\langle C, 0 \rangle \\
&\langle \neg C, 0 \rangle \\
&\langle D, 1 \rangle \\
&\langle \neg D, 0 \rangle
\end{align*}
\]
Reward machines in action
Reward machines in action

\[
\begin{align*}
\langle A, 0 \rangle & \rightarrow \langle \neg A, 0 \rangle \\
\langle B, 0 \rangle & \rightarrow \langle \neg B, 0 \rangle \\
\langle C, 0 \rangle & \rightarrow \langle \neg C, 0 \rangle \\
\langle D, 1 \rangle & \rightarrow \langle D, 0 \rangle
\end{align*}
\]
Reward machines in action
Reward machines in action

\[ \langle A, 0 \rangle \quad \langle \neg A, 0 \rangle \quad \langle B, 0 \rangle \quad \langle \neg B, 0 \rangle \quad \langle C, 0 \rangle \quad \langle \neg C, 0 \rangle \quad \langle D, 1 \rangle \quad \langle \neg D, 0 \rangle \]

\[ \text{u}_0 \quad \text{u}_1 \quad \text{u}_2 \quad \text{u}_3 \]
Reward machines in action
Reward machines in action

B
*  
*  
*  
*  
C

A
*  
*  
*  
D

\[
(\neg A, 0) \\
(\neg B, 0) \\
(\neg C, 0) \\
(\neg D, 0) \\
(D, 1) \\
(A, 0) \\
(C, 0) \\
(B, 0)
\]

\[
\langle A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle C, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg A, 0 \rangle \\
\langle \neg B, 0 \rangle \\
\langle \neg C, 0 \rangle \\
\langle \neg D, 0 \rangle
\]
Reward machines in action

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td>*</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>O</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

\[
\langle A, 0 \rangle \\
\langle \neg A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle \neg B, 0 \rangle \\
\langle C, 0 \rangle \\
\langle \neg C, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg D, 0 \rangle \\
\]
Reward machines in action

B ♦️ * ♦️ * ♦️ C
* ♦️ O ♦️ ♦️ *
* ♦️ ♦️ ♦️ ♦️ ♦️ A
* ♦️ ♦️ ♦️ ♦️ D
Reward machines in action
Reward machines in action

The diagram on the left represents a reward machine with states A, B, C, and D, where each state is marked with symbols (*), (coffee cup), and (triangle). The transitions between states are indicated by arrows, such as from A to B, C, and D.

The diagram on the right is a more detailed representation of the transition graph of the reward machine. It consists of four states: u₀, u₁, u₂, and u₃, with directed edges showing the transitions between them. The transitions are labeled with symbols (D, 1), (A, 0), (−D, 0), (C, 0), (B, 0), and (−C, 0).
Reward machines in action
Reward machines in action

\[
\langle A, 0 \rangle, \langle \neg A, 0 \rangle, \langle B, 0 \rangle, \langle \neg B, 0 \rangle, \langle C, 0 \rangle, \langle \neg C, 0 \rangle, \langle D, 0 \rangle, \langle D, 1 \rangle, \langle A, 0 \rangle, \langle B, 0 \rangle
\]
Reward machines in action
Reward machines in action
Reward machines in action

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>*</th>
<th>*</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
\langle \neg A, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg D, 0 \rangle \\
\langle C, 0 \rangle \\
\langle \neg C, 0 \rangle \\
\langle A, 0 \rangle \\
\langle B, 0 \rangle \\
\langle \neg B, 0 \rangle \\
\end{array}
\]

\[
\begin{array}{c}
u_0 \\
u_1 \\
u_2 \\
u_3 \\
\end{array}
\]
Reward machines in action

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>*</th>
<th>*</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td>*</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- **u₀** with transitions:
  - **D**, 1
  - **¬A**, 0

- **u₁** with transitions:
  - **A**, 0
  - **¬B**, 0

- **u₂** with transitions:
  - **B**, 0
  - **¬C**, 0

- **u₃** with transitions:
  - **C**, 0
  - **¬D**, 0
Other reward machines

**Task:** Deliver coffee to the office.

\[\langle \neg \text{☕}, 0 \rangle \quad \langle \neg o, 0 \rangle \quad \langle \text{true}, 0 \rangle\]
Other reward machines

**Task**: Deliver coffee to the office.
Other reward machines

Task: Deliver coffee to the office.

\[ \langle \neg \text{coffee}, 0 \rangle \quad \langle \text{coffee}, 0 \rangle \quad \langle \neg \text{o}, 0 \rangle \quad \langle \text{o}, 1 \rangle \quad \langle \text{true}, 0 \rangle \]
Other reward machines

**Task:** Deliver coffee to the office.
Other reward machines

**Task:** Deliver coffee and the mail to the office.

\[
\langle \neg \text{\text{Coffee}}, \neg \text{\text{Mail}}, 0 \rangle
\]

\[
\langle \text{\text{Mail}}, 0 \rangle
\]

\[
\langle \neg \text{\text{Mail}}, 0 \rangle
\]

\[
\langle \text{\text{Coffee}}, 0 \rangle
\]

\[
\langle \text{\text{Mail}}, 0 \rangle
\]

\[
\langle \neg \text{\text{Coffee}}, 0 \rangle
\]

\[
\langle \text{\text{true}}, 0 \rangle
\]

\[
\langle \neg \text{\text{true}}, 0 \rangle
\]
**Task:** Deliver coffee and the mail to the office.
Other reward machines

**Task**: Deliver coffee and the mail to the office.
**Task**: Deliver coffee to the office while avoiding the furniture.
**Task**: Deliver coffee to the office while avoiding the furniture.
**Task:** Deliver coffee to the office while avoiding the furniture.
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
Exploiting reward machines’ structure

We explored 4 ideas.
We explored 4 ideas.

**Baselines:**
- Q-Learning over a cross-product MDP (q-learning).
- Hierarchical RL based on options (HRL).
- Hierarchical RL with option pruning (HRL-RM).
Exploiting reward machines’ structure

We explored 4 ideas.

**Baselines:**
- Q-Learning over a cross-product MDP (q-learning).
- Hierarchical RL based on options (HRL).
- Hierarchical RL with option pruning (HRL-RM).

**Our approach:**
- Q-learning for Reward Machines (QRM).
Q-learning baseline

Reward machines might define non-Markovian rewards.
Q-learning baseline

Reward machines might define non-Markovian rewards.
Q-learning baseline

Reward machines might define non-Markovian rewards.

<table>
<thead>
<tr>
<th>B</th>
<th>*</th>
<th>*</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>o</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

\[
\langle A, 0 \rangle \quad \langle -A, 0 \rangle \\
\langle D, 1 \rangle \\
\langle -D, 0 \rangle \\
\langle C, 0 \rangle \\
\langle -C, 0 \rangle \\
\langle B, 0 \rangle \\
\langle -B, 0 \rangle \\
\langle A, 0 \rangle \\
\langle -A, 0 \rangle \\
\langle D, 1 \rangle \\
\langle -D, 0 \rangle \\
\langle C, 0 \rangle \\
\langle -C, 0 \rangle \\
\langle B, 0 \rangle \\
\langle -B, 0 \rangle \
\]

Solution (q-learning baseline)
Include the RM state to the agent's state representation.
Learn policies using standard q-learning.
Q-learning baseline

Reward machines might define non-Markovian rewards.

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>*</th>
<th>*</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>O</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\langle A, 0 \rangle & \quad \langle \neg A, 0 \rangle \\
\langle D, 1 \rangle & \quad \langle A, 0 \rangle \\
\langle \neg D, 0 \rangle & \quad \langle \neg B, 0 \rangle \\
\langle C, 0 \rangle & \quad \langle B, 0 \rangle \\
\langle \neg C, 0 \rangle & \\
\end{align*}
\]
Q-learning baseline

Reward machines might define non-Markovian rewards.

Solution (q-learning baseline)
Include the RM state to the agent's state representation.

Learn policies using standard q-learning.

Toro Icarte et al: Using RMs for Task Specification and Decomposition in RL
Reward machines might define non-Markovian rewards.
Q-learning baseline

Reward machines might define non-Markovian rewards.

\[
\begin{array}{cccc}
B & * & * & C \\
* & o & * & \\
A & * & * & \\
\end{array}
\]

Solution (q-learning baseline)

Include the RM state to the agent's state representation.

Learn policies using standard q-learning.

Toro Icarte et al: Using RMs for Task Specification and Decomposition in RL
Q-learning baseline

Reward machines might define non-Markovian rewards.

![Diagram of reward machines]
Q-learning baseline

Reward machines might define non-Markovian rewards.
Q-learning baseline

Reward machines might define non-Markovian rewards.

Solution (q-learning baseline)

Include the RM state to the agent’s state representation.
Learn policies using standard q-learning.
Hierarchical RL baseline

Learn meta-controller over a set of options (macro-actions).

Define one option per proposition in the RM's transitions.

Optimize $\pi_i$ to satisfy $i$ optimally.
### HRL baseline

Learn meta-controller over a set of options (macro-actions). Define one option per proposition in the RM’s transitions. Optimize $\pi_i$ to satisfy $i$ optimally.
Hierarchical RL baseline

HRL baseline

Learn meta-controller over a set of options (macro-actions). Define one option per proposition in the RM’s transitions. Optimize $\pi_i$ to satisfy $i$ optimally.
Hierarchical RL with RM pruning baseline

**HRL-RM baseline**

Prune useless options using the current reward machine state.
Hierarchical RL with RM pruning baseline

**HRL-RM baseline**

Prune useless options using the current reward machine state.

---

\[
\begin{array}{cccc}
B & * & * & C \\
* & o & * & * \\
A & * & * & D \\
\end{array}
\]

---

Meta-Controller

\[
\begin{array}{cccc}
\pi_A & \pi_B & \pi_C & \pi_D \\
\end{array}
\]
Hierarchical RL with RM pruning baseline

**HRL-RM baseline**

Prune useless options using the current reward machine state.
Hierarchical RL might converge to suboptimal policies
Hierarchical RL might converge to suboptimal policies

Reason: Policy \( \pi \) goes to the closest state.
Hierarchical RL might converge to suboptimal policies
Hierarchical RL might converge to suboptimal policies

Reason: Policy $\pi_c$ goes to the closest $c$. 
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
3. Reuse experience to update all the q-values at the same time.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learner for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

**QRM (our approach)**

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

QRM (our approach)

1. Learn one policy (q-function) per state in the reward machine.

2. Select actions using the policy of the current RM state.
Q-learning for Reward Machines (QRM)

**QRM (our approach)**

1. Learn one policy (q-function) per state in the reward machine.
2. Select actions using the policy of the current RM state.
3. Reuse experience to update all the q-values at the same time.
QRM learning step

\[
\langle A, 0 \rangle \quad \langle \neg A, 0 \rangle \\
\langle B, 0 \rangle \quad \langle \neg B, 0 \rangle \\
\langle C, 0 \rangle \quad \langle \neg C, 0 \rangle \\
\langle D, 0 \rangle \quad \langle \neg D, 0 \rangle
\]
QRM learning step

\[ \langle A, 0 \rangle, \langle \neg A, 0 \rangle, \langle B, 0 \rangle, \langle \neg B, 0 \rangle, \langle C, 0 \rangle, \langle \neg C, 0 \rangle, \langle D, 1 \rangle, \langle \neg D, 0 \rangle \]
QRM learning step

\[
\begin{align*}
\langle A, 0 \rangle & \quad \langle \neg A, 0 \rangle \\
\langle B, 0 \rangle & \quad \langle \neg B, 0 \rangle \\
\langle C, 0 \rangle & \quad \langle \neg C, 0 \rangle \\
\langle D, 1 \rangle & \quad q_3 \quad q_1 \\
\langle \neg D, 0 \rangle & \quad q_2 \\
\end{align*}
\]
QRM learning step

\[
\langle \neg A, 0 \rangle \\
\langle A, 0 \rangle \\
\langle D, 1 \rangle \\
\langle \neg D, 0 \rangle \\
\langle C, 0 \rangle \\
\langle B, 0 \rangle \\
\langle \neg C, 0 \rangle
\]
QRM learning step

\[
\langle A, 0 \rangle, \langle ¬A, 0 \rangle, \langle B, 0 \rangle, \langle ¬B, 0 \rangle, \langle C, 0 \rangle, \langle ¬C, 0 \rangle, \langle D, 1 \rangle, \langle ¬D, 0 \rangle
\]

\[
\langle A, 0 \rangle \rightarrow \langle B, 0 \rangle \rightarrow \langle C, 0 \rangle \rightarrow \langle ¬C, 0 \rangle \rightarrow \langle D, 1 \rangle \rightarrow \langle ¬D, 0 \rangle
\]
QRM learning step
QRM learning step

\[ q_0(s, a) \leftarrow 0 + \gamma \max_{a'} q_0(s', a') \]
QRM learning step

\[
q_1(s, a) \leftarrow 0 + \gamma \max_{a'} q_1(s', a')
\]
QRM learning step

\[ q_2(s, a) \leftarrow 0 + \gamma \max_{a'} q_2(s', a') \]
QRM learning step
QRM learning step
QRM learning step

\[
\langle A, 0 \rangle \quad \langle B, 0 \rangle \\
\langle C, 0 \rangle \quad \langle D, 1 \rangle
\]

\[
\langle \neg A, 0 \rangle \quad \langle \neg B, 0 \rangle \\
\langle \neg C, 0 \rangle \quad \langle \neg D, 0 \rangle
\]
QRM learning step

\[
\langle A, 0 \rangle \quad \langle \neg A, 0 \rangle
\]

\[
\langle B, 0 \rangle \quad \langle \neg B, 0 \rangle
\]

\[
\langle C, 0 \rangle \quad \langle \neg C, 0 \rangle
\]

\[
\langle D, 1 \rangle \quad \langle \neg D, 0 \rangle
\]

\[
q_0 \quad q_1 \quad q_2 \quad q_3
\]

\[
s_0 \quad s_1 \quad s_2 \quad s_3
\]

\[
q_0 \quad q_1 \quad q_2 \quad q_3
\]
\[ q_3(s, a) \leftarrow \alpha 1 + \gamma \max_{a'} q_0(s', a') \]
Theorem

QRM converges to an optimal policy in the limit.
1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
Experiments

Two discrete grid domains:
- Office domain (4 tasks).
- Craft domain (10 tasks).
Experiments

Two discrete grid domains:
- Office domain (4 tasks).
- Craft domain (10 tasks).

One continuous state space domain:
- Water domain (10 tasks).
Discrete domains

## Algorithms

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimality?</th>
<th>Decomposition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-Learning over a cross-product MDP (Q-learning)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchical RL based on options (HRL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchical RL with option pruning (HRL-RM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q-learning for reward machines (QRM)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Discrete domains

### Algorithms

- Q-Learning over a cross-product MDP (**Q-learning**)
- Hierarchical RL based on options (**HRL**)
- Hierarchical RL with option pruning (**HRL-RM**)
- Q-learning for reward machines (**QRM**)

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimality?</th>
<th>Decomposition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-learning</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>HRL</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HRL-RM</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>QRM</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
The office domain

4 tasks (30 independent trials)
The office domain

4 tasks (30 independent trials)
The craft domain

Minecraft World

Number of training steps

Normalized discounted reward

10 tasks defined by Andreas et al.\(^1\) over 10 random maps (3 trials)

---

\(^1\) Modular Multitask Reinforcement Learning with Policy Sketches by Andreas et al. (ICML-17)
The craft domain

10 tasks defined by Andreas et al.\(^1\) over 10 random maps (3 trials)

\(^1\)Modular Multitask Reinforcement Learning with Policy Sketches by Andreas et al. (ICML-17)
From tabular QRM to Deep QRM

We replaced q-learning by Double DQN with prioritized experience replay in our four approaches.
Continuous domains

From tabular QRM to Deep QRM

We replaced q-learning by Double DQN with prioritized experience replay in our four approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimality?</th>
<th>Decomposition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDQN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHRL</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>DHRL-RM</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>DQRM</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
The water domain

10 tasks over 10 random maps (3 trials per map)
The water domain

10 tasks over 10 random maps (3 trials per map)
Outline

1. Motivation
2. What is a reward machine (RM)?
3. How to exploit a reward machine’s structure
4. Results
5. Related work
6. Concluding remarks
Outline

1. Motivation
2. What is a reward machine (RM)?
3. How to exploit a reward machine’s structure
4. Results
5. Related work
6. Concluding remarks
Related work

Hierarchical RL (task decomposition):


HRL might converge to suboptimal policies.

Lot of relations between QRM and HRL (more in the paper!)

Toro Icarte et al: Using RMs for Task Specification and Decomposition in RL
Hierarchical RL (task decomposition):


Hierarchical RL (task decomposition):


HRL might converge to suboptimal policies.
Related work

Hierarchical RL (task decomposition):


HRL might converge to suboptimal policies.

Lot of relations between QRM and HRL (more in the paper!)
Related work

Linear Temporal Logic (task specification):


Linear Temporal Logic (task specification):


Related work

Linear Temporal Logic (task specification):


RM’s can express reward functions that cannot be expressed in LTL.
1. Motivation
2. What is a reward machine (RM)?
3. How to exploit a reward machine’s structure
4. Results
5. Related work
6. Concluding remarks
Outline

1 Motivation
2 What is a reward machine (RM)?
3 How to exploit a reward machine’s structure
4 Results
5 Related work
6 Concluding remarks
We proposed to show the reward function’s code to the agent.
Concluding remarks

... so it can exploit the reward’s structure.

```
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
```
To define reward functions, we used reward machines.
Concluding remarks

... and showed how to decompose the problem using QRM.
Concluding remarks

QRM outperformed plain RL and HRL in 2 discrete domains.

Office World

Minecraft World

Legend:
- Q-Learning
- HRL
- HRL-RM
- QRM
... and was also effective when combined with deep learning.
Title: Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning

Code: https://bitbucket.org/RToroIcarte/qrm

Poster: #147