Teaching Multiple Tasks to a Reinforcement Learning Agent using Linear Temporal Logic (LTL)

Rodrigo Toro Icarte Toryn Q. Klassen
Richard Valenzano Sheila A. McIlraith

Computer Science
UNIVERSITY OF TORONTO

ELEMENT AI

VECTOR INSTITUTE | INSTITUT VECTEUR

AAMAS 2018

July 11

Outline

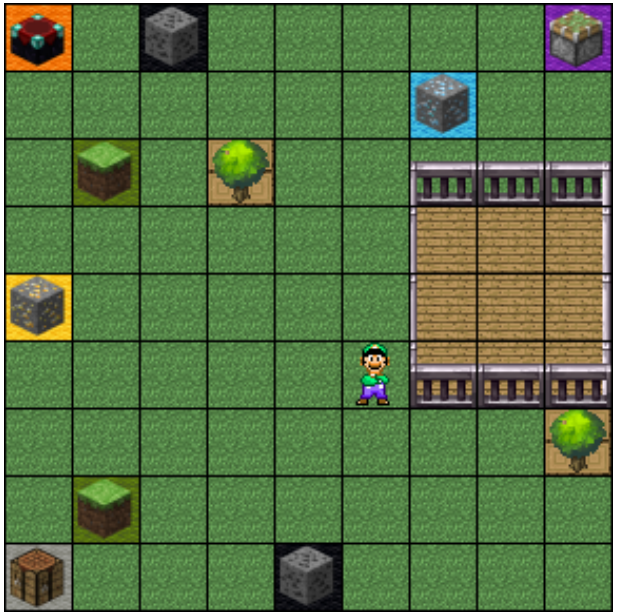
1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

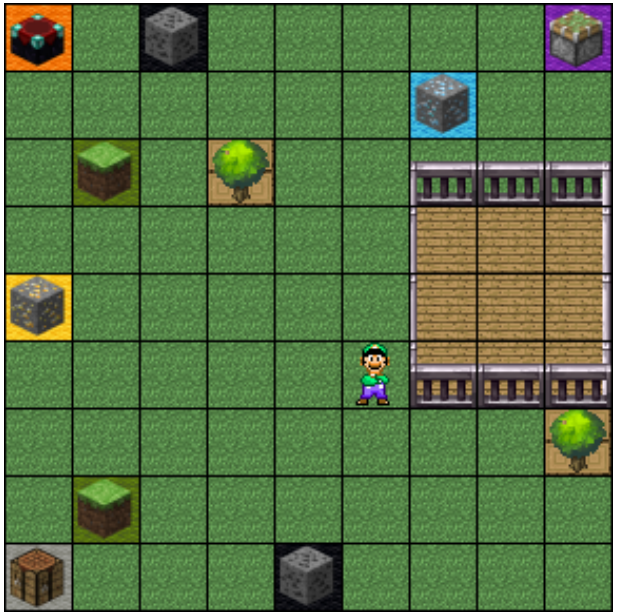
Motivation

How do you describe to an RL agent what to do?

Running example



Running example



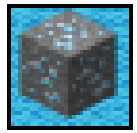
Luigi can collect raw materials:


wood

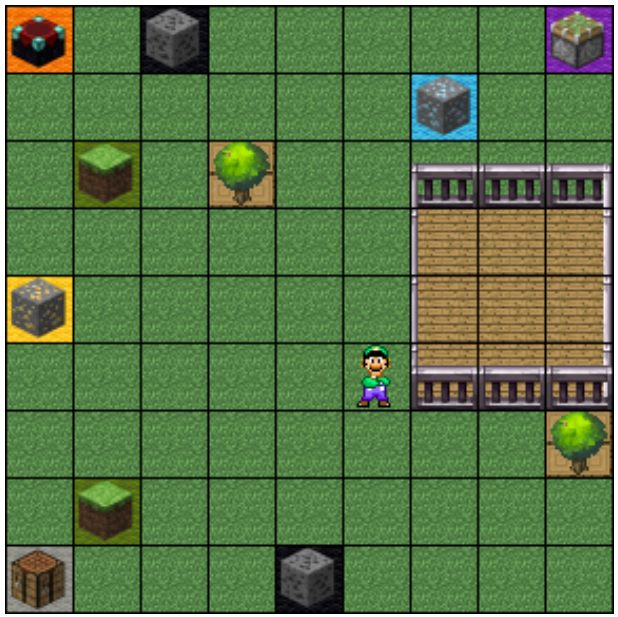

grass


iron


gold


gems

Running example

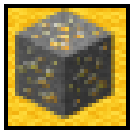


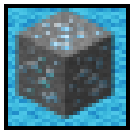
Luigi can collect raw materials:


wood


grass


iron


gold


gems

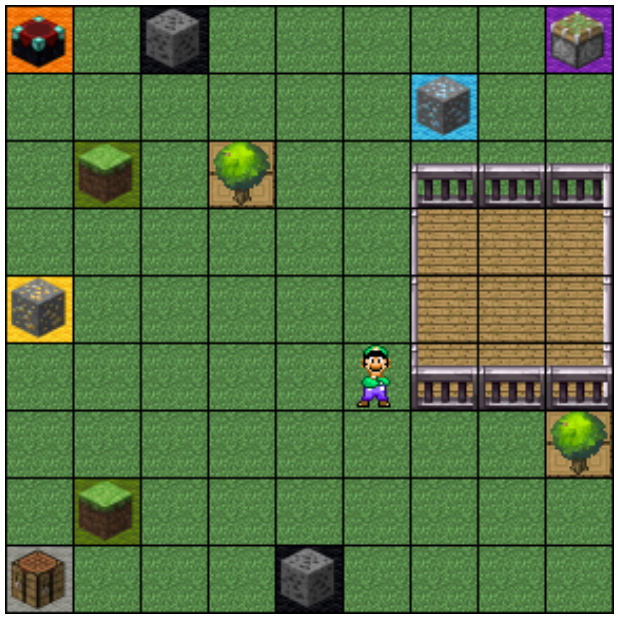
... and make new objects in:


factory


toolshed


workbench

Running example



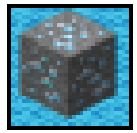
Luigi can collect raw materials:


wood


grass


iron


gold


gems

... and make new objects in:


factory


toolshed


workbench

Make a bridge: get wood, iron, and use the factory

Running example

Task type	Example
Single goal	get wood
Sequence of goals	get wood and then use the factory
Disjunctive goals	get wood or iron
Conjunctive goals	get grass and iron
Safety constraints	do not leave the shelter at night

Running example

Task type	Example
Single goal	get wood
Sequence of goals	get wood and then use the factory
Disjunctive goals	get wood or iron
Conjunctive goals	get grass and iron
Safety constraints	do not leave the shelter at night

How do you describe to an RL agent what to do?

Motivation

How do you describe to an RL agent what to do?

Description $\neq$ Reward function

Motivation

How do you describe to an RL agent what to do?

Description $\neq$ Reward function

Language $\rightarrow$ Reward function

Motivation

How do you describe to an RL agent what to do?

Description $\neq$ Reward function

Language $\rightarrow$ Reward function

Why would we want such a language?

Motivation

How do you describe to an RL agent what to do?

Description $\neq$ Reward function

Language $\rightarrow$ Reward function

Why would we want such a language?

To define new task faster.

To transfer learning between tasks.

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

1. Motivation
2. **Related work**
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Related works

Hindsight Experience Replay by Andrychowicz et al. (NIPS-17)

Language: Single goal condition
Advantage: Learn to achieve goals in parallel (off-policy RL)

Related works

Hindsight Experience Replay by Andrychowicz et al. (NIPS-17)

Language: Single goal condition
Advantage: Learn to achieve goals in parallel (off-policy RL)

Task	HER
get wood	✓
get wood and then use the factory	
get wood or iron	
get grass and iron	
do not leave the shelter at night	

Related works

Modular Multitask RL with Policy Sketches by Andreas et al. (ICML-17)

Language: Sequence of sub-goals (called sketch)
Advantage: Decompose the problem using the sketch.

Related works

Modular Multitask RL with Policy Sketches by Andreas et al. (ICML-17)

Language: Sequence of sub-goals (called sketch)
Advantage: Decompose the problem using the sketch.

Task	HER	Sketches
get wood	✓	✓
get wood and then use the factory		✓
get wood or iron		
get grass and iron		
do not leave the shelter at night		

This work

Teaching Multiple Tasks to an RL Agent using LTL

Language: Linear Temporal Logic (LTL)
Advantage: Decompose and use off-policy RL to learn subtasks.

This work

Teaching Multiple Tasks to an RL Agent using LTL

Language: Linear Temporal Logic (LTL)
Advantage: Decompose and use off-policy RL to learn subtasks.

Task	HER	Sketches	LTL
get wood	✓	✓	✓
get wood and then use the factory		✓	✓
get wood or iron			✓
get grass and iron			✓
do not leave the shelter at night			✓

This work

Teaching Multiple Tasks to an RL Agent using LTL

Language: Linear Temporal Logic (LTL)
Advantage: Decompose and use off-policy RL to learn subtasks.

Task	HER	Sketches	LTL
get wood	✓	✓	✓
get wood and then use the factory		✓	✓
get wood or iron			✓
get grass and iron			✓
do not leave the shelter at night			✓
Off-policy learning	✓		✓
Task decomposition		✓	✓

Related works

Other works using variants of LTL in RL:

- Reinforcement learning with temporal logic rewards
  by Li et al. (IROS, 2017)
- Environment-independent task specifications via GLTL
  by Littman et al. (arXiv, 2017)
- Logically constrained reinforcement learning
  by Hasanbeig et al. (arXiv, 2018)

Related works

Other works using variants of LTL in RL:

- Reinforcement learning with temporal logic rewards
  by Li et al. (IROS, 2017)
- Environment-independent task specifications via GLTL
  by Littman et al. (arXiv, 2017)
- Logically constrained reinforcement learning
  by Hasanbeig et al. (arXiv, 2018)

We exploit task decomposition and off-policy RL.

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

1. Motivation
2. Related work
3. Approach:
   - **Why (and how) we use LTL for specifying tasks in RL.**
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Defining tasks using LTL

Idea: Let's give the RL agent a set of high-level event detectors.

Defining tasks using LTL

Idea: Let's give the RL agent a set of high-level event detectors.

Example

$$\mathcal{P} = \{\texttt{got\_wood}, \texttt{got\_iron}, \texttt{got\_grass}, \texttt{used\_workbench}, \texttt{used\_factory}, \texttt{is\_night}, \texttt{at\_shelter}, \ldots\}$$

Defining tasks using LTL

Idea: Let's give the RL agent a set of high-level event detectors.

Example

$$\mathcal{P} = \{\texttt{got\_wood}, \texttt{got\_iron}, \texttt{got\_grass}, \texttt{used\_workbench}, \texttt{used\_factory}, \texttt{is\_night}, \texttt{at\_shelter}, ...\}$$

Use LTL to define tasks by composing occurrences of events in $\mathcal{P}$

Defining tasks using LTL

Linear Temporal Logic (syntax)

LTL augments propositional logic with temporal operators $\bigcirc$ (*next*), $\Diamond$ (*eventually*), and U (*until*):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \Diamond\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \text{ with } p \in \mathcal{P}$$

Defining tasks using LTL

Linear Temporal Logic (syntax)

LTL augments propositional logic with temporal operators $\bigcirc$ (*next*), $\Diamond$ (*eventually*), and U (*until*):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \Diamond\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \text{ with } p \in \mathcal{P}$$

Examples

$\Diamond$`got_wood`

$\Diamond$(`got_grass` $\wedge$ $\Diamond$`used_factory`)

$\Diamond$`got_wood` $\vee$ $\Diamond$`got_iron`

$\Diamond$`got_grass` $\wedge$ $\Diamond$`got_iron`

(`is_night` $\rightarrow$ `at_shelter`) U `got_wood`

Defining tasks using LTL

Linear Temporal Logic (syntax)

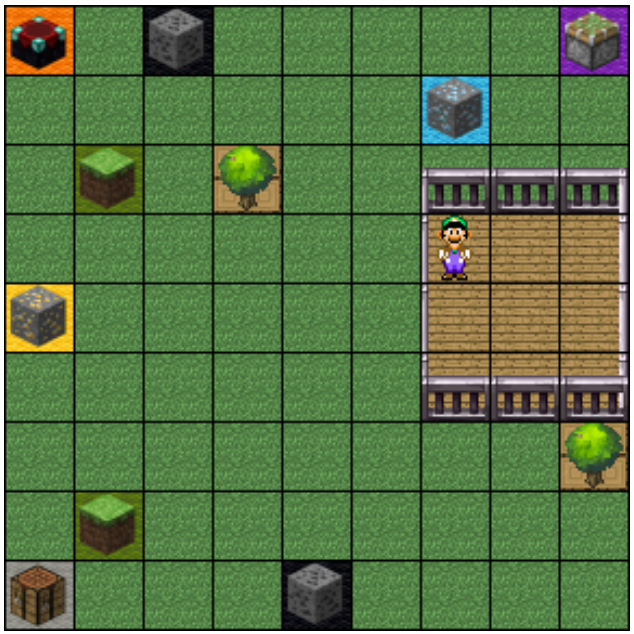
LTL augments propositional logic with temporal operators $\bigcirc$ (*next*), $\Diamond$ (*eventually*), and U (*until*):

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \Diamond\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \text{ with } p \in \mathcal{P}$$

Examples

eventually `got_wood`
eventually (`got_grass` and **eventually** `used_factory`)
eventually `got_wood` or **eventually** `got_iron`
eventually `got_grass` and **eventually** `got_iron`
(`is_night`$\rightarrow$`at_shelter`) **until** `got_wood`

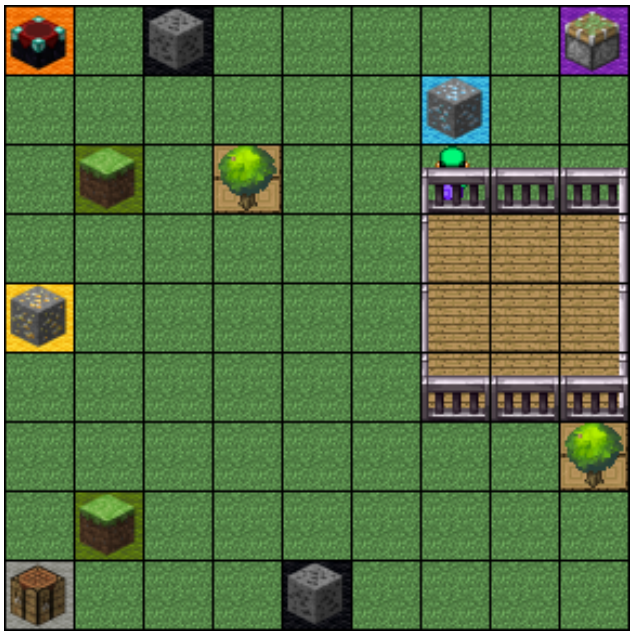
Example



Detected events
`at_shelter`

$\varphi =$ **eventually**(`got_iron` and **eventually** `used_factory`)
and **eventually** `got_gold`

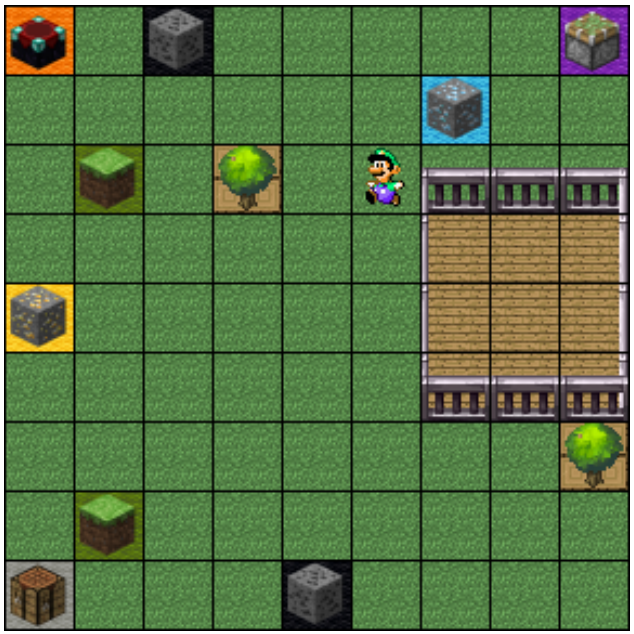
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

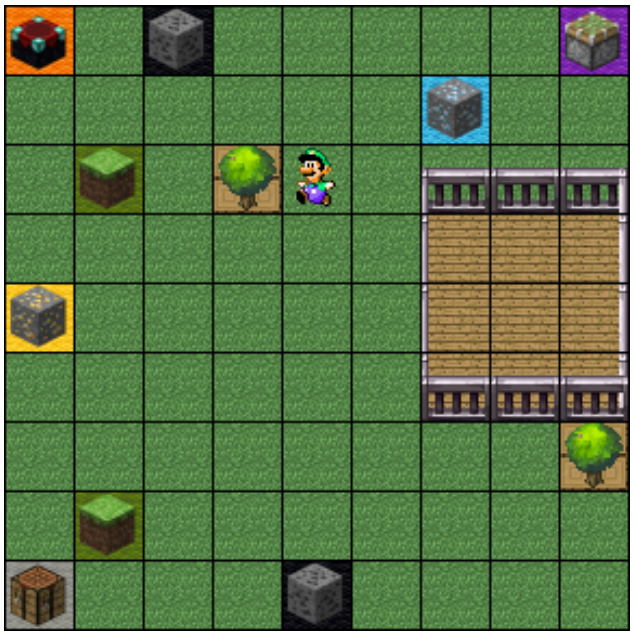
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

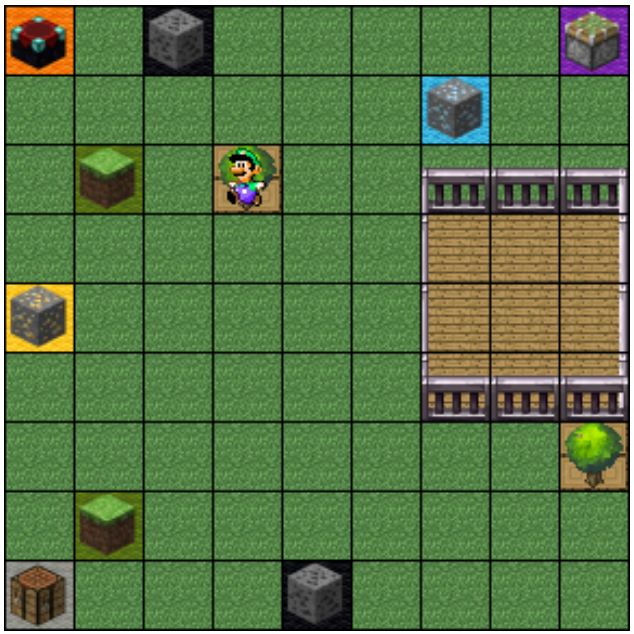
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

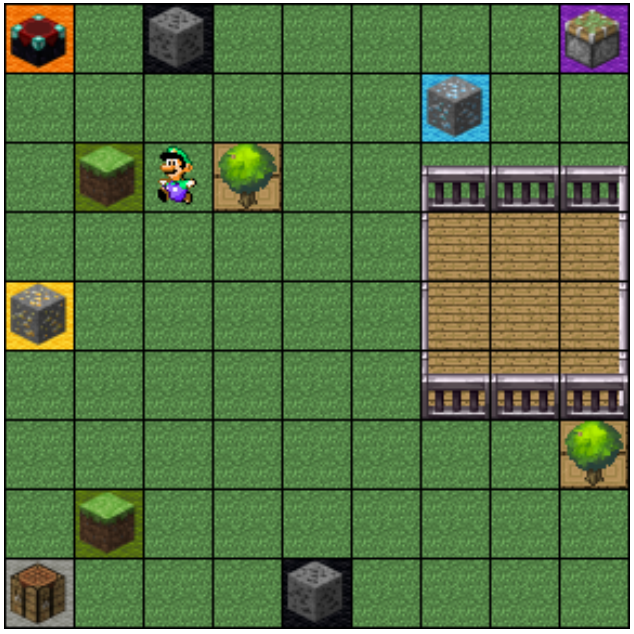
Example



Detected events
got_wood

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

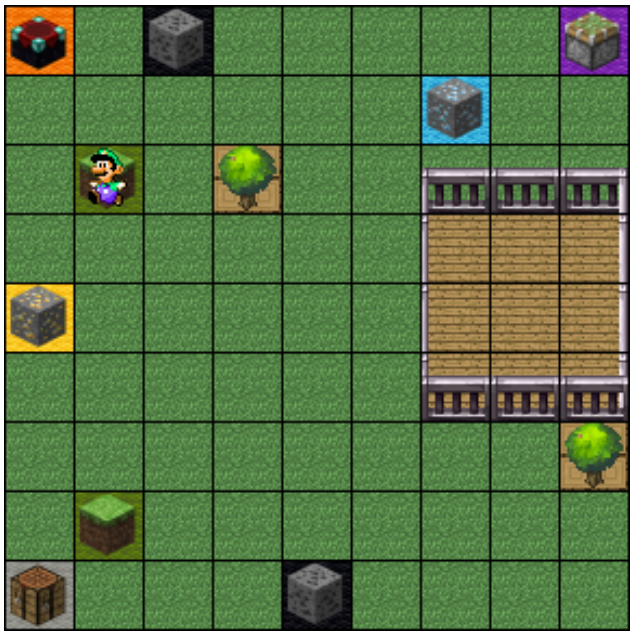
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

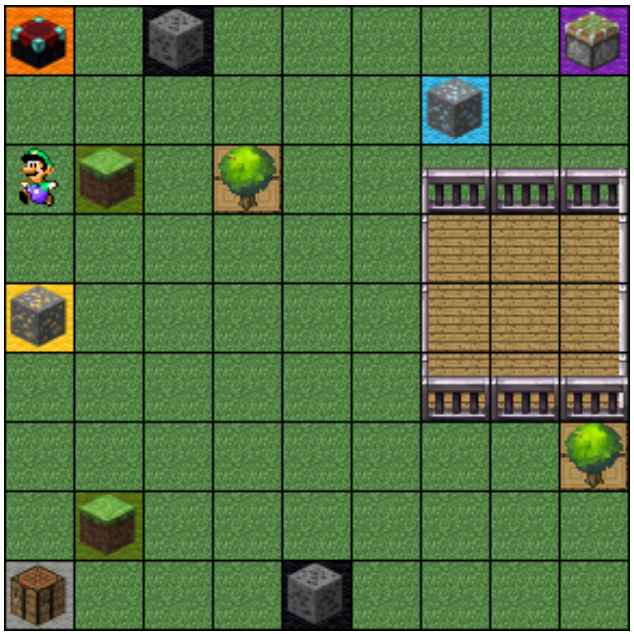
Example



Detected events
got_grass

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

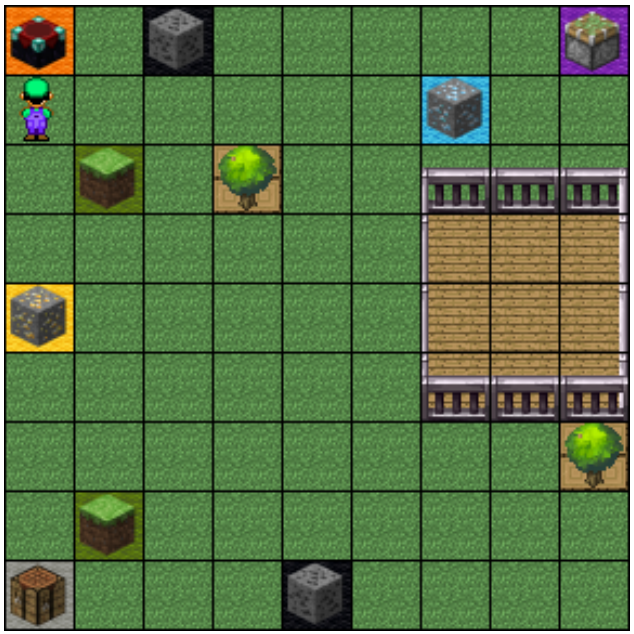
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

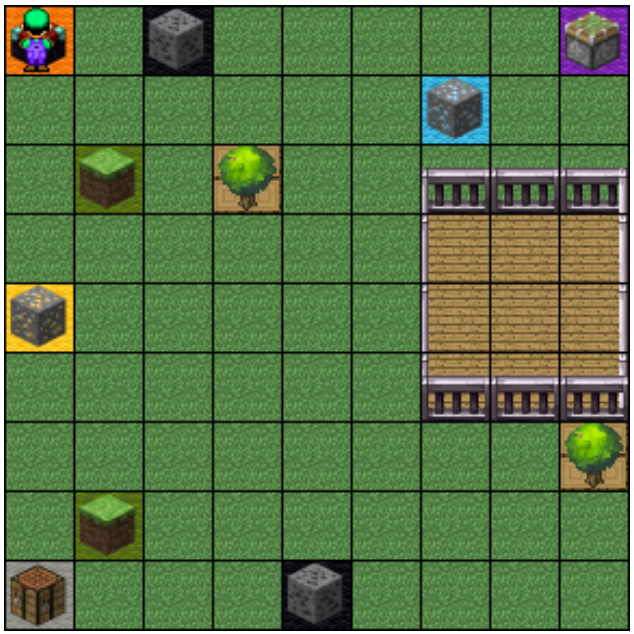
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

Example

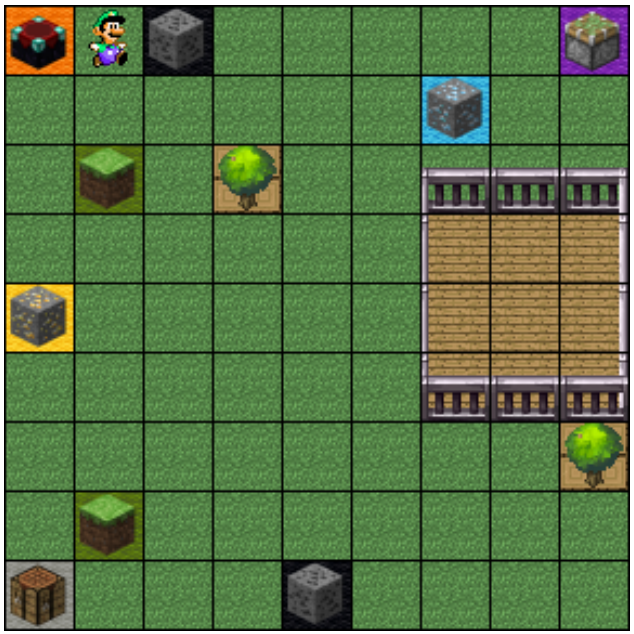


Detected events

`used_factory`

$\varphi =$ **eventually**(`got_iron` and **eventually** `used_factory`)
and **eventually** `got_gold`

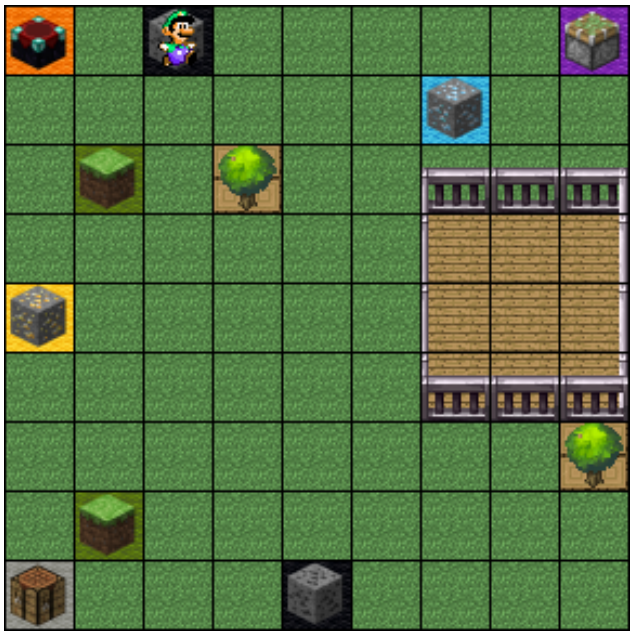
Example



Detected events
none

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

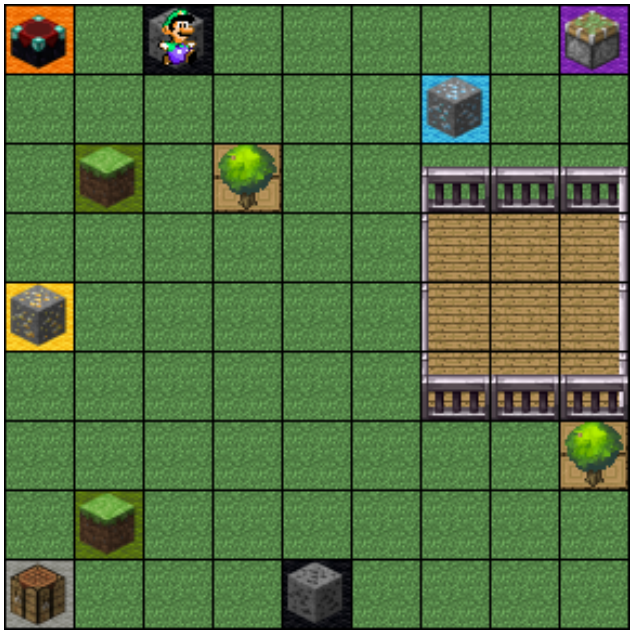
Example



Detected events
got_iron

$\varphi =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold

Example



Detected events

got_iron

$\varphi =$ **eventually** used_factory and **eventually** got_gold

LTL progression

LTL progression

Given an LTL formula φ and state s, we can *progress* φ using s:

- $\text{prog}(s, p) = \text{true}$ if $p \in L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, p) = \text{false}$ if $p \notin L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, \neg\varphi) = \neg\, \text{prog}(s, \varphi)$
- $\text{prog}(s, \varphi_1 \wedge \varphi_2) = \text{prog}(s, \varphi_1) \wedge \text{prog}(s, \varphi_2)$
- $\text{prog}(s, \bigcirc\varphi) = \varphi$
- $\text{prog}(s, \Diamond\varphi) = \text{prog}(s, \varphi) \vee \Diamond\varphi$
- $\text{prog}(s, \varphi_1 \,\mathsf{U}\, \varphi_2) = \text{prog}(s, \varphi_2) \vee (\text{prog}(s, \varphi_1) \wedge \varphi_1 \,\mathsf{U}\, \varphi_2)$

LTL progression

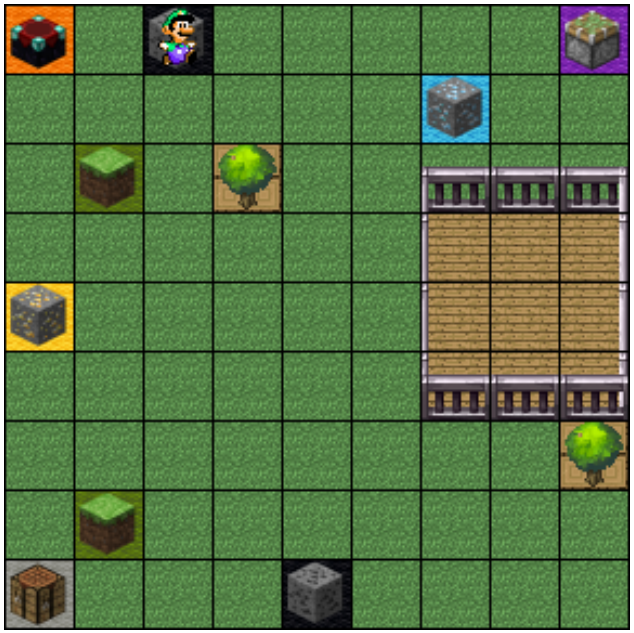
LTL progression

Given an LTL formula φ and state s, we can *progress* φ using s:

- $\text{prog}(s, p) = \text{true}$ if $p \in L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, p) = \text{false}$ if $p \notin L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, \neg\varphi) = \neg\,\text{prog}(s, \varphi)$
- $\text{prog}(s, \varphi_1 \wedge \varphi_2) = \text{prog}(s, \varphi_1) \wedge \text{prog}(s, \varphi_2)$
- $\text{prog}(s, \bigcirc\varphi) = \varphi$
- $\text{prog}(s, \Diamond\varphi) = \text{prog}(s, \varphi) \vee \Diamond\varphi$
- $\text{prog}(s, \varphi_1\,\mathsf{U}\,\varphi_2) = \text{prog}(s, \varphi_2) \vee (\text{prog}(s, \varphi_1) \wedge \varphi_1\,\mathsf{U}\,\varphi_2)$

This is a **correct** and **well-defined** procedure!

Example

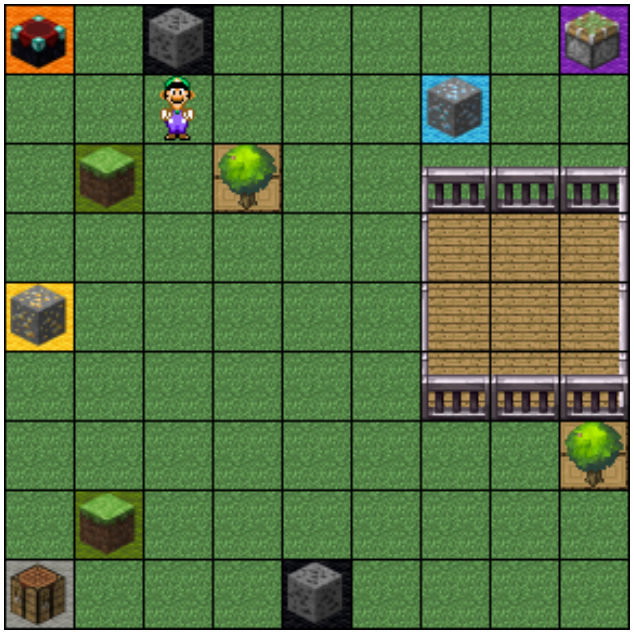


Detected events

got_iron

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

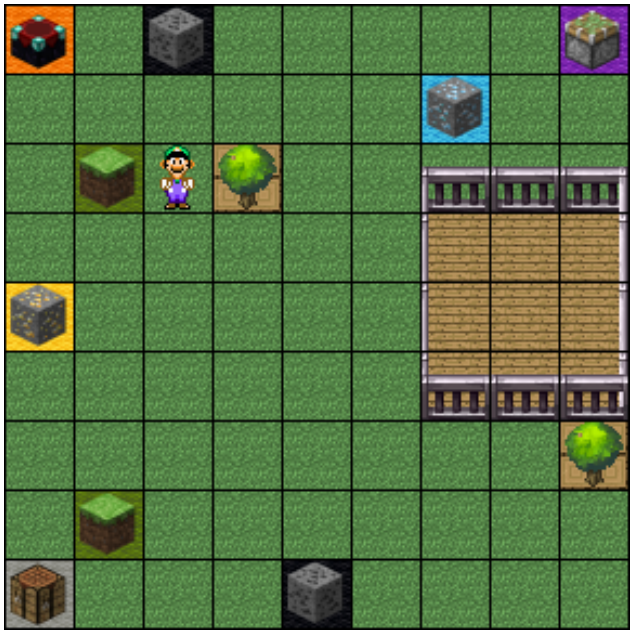


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

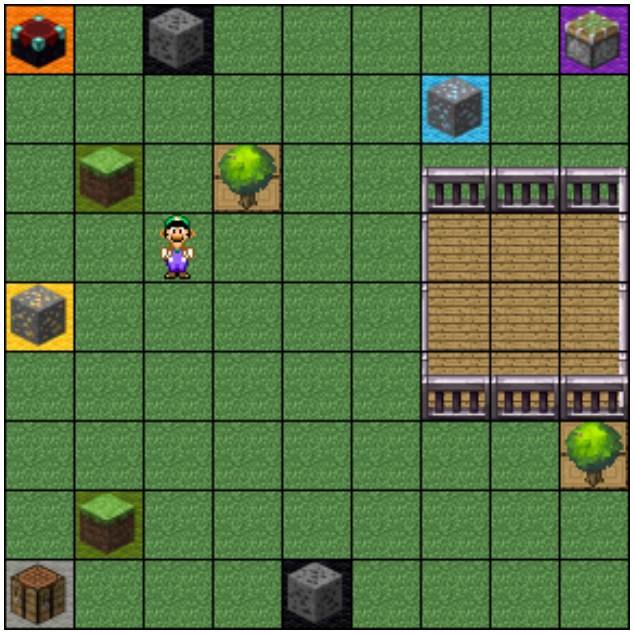


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

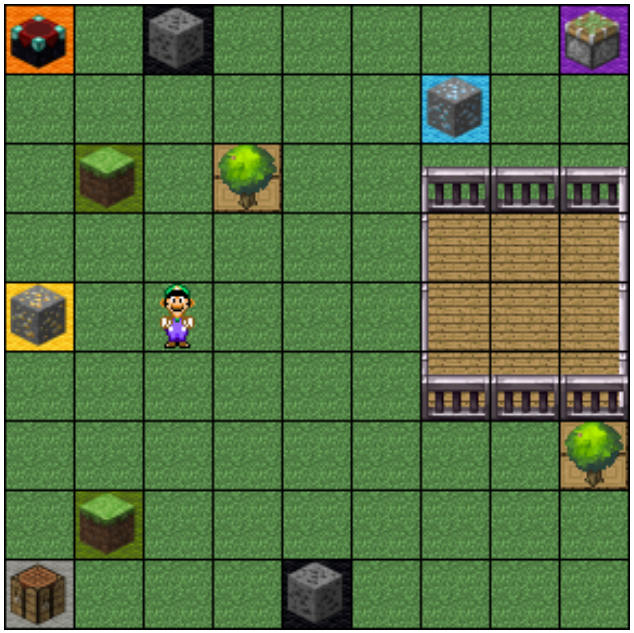


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

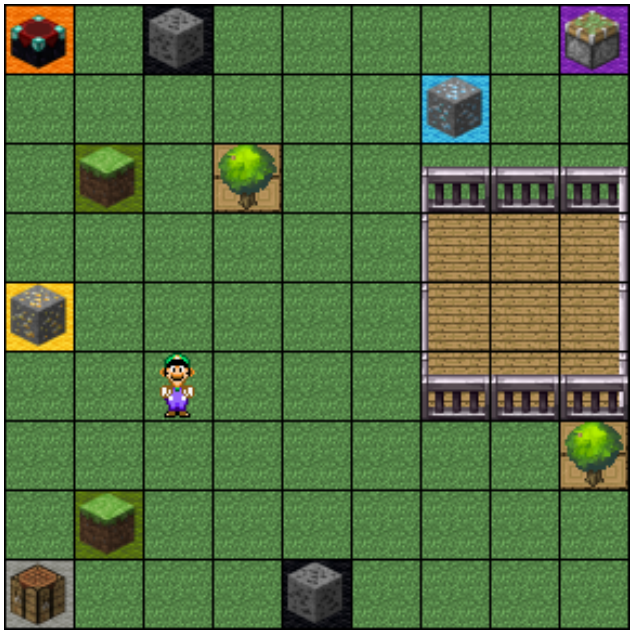


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

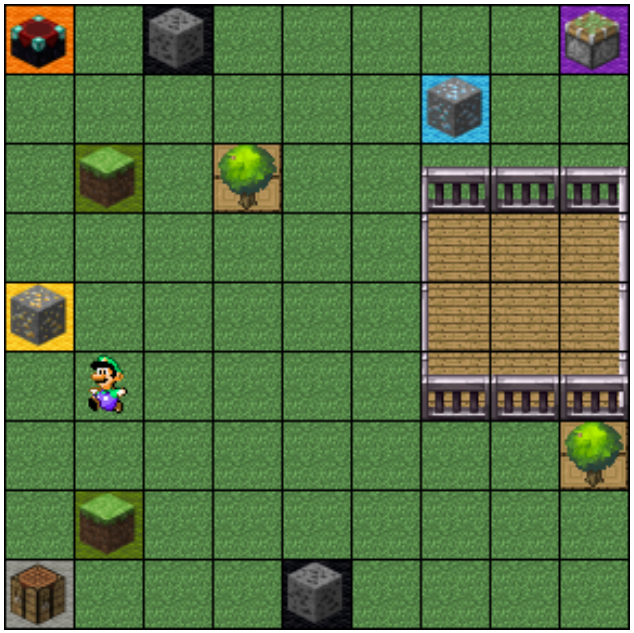


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

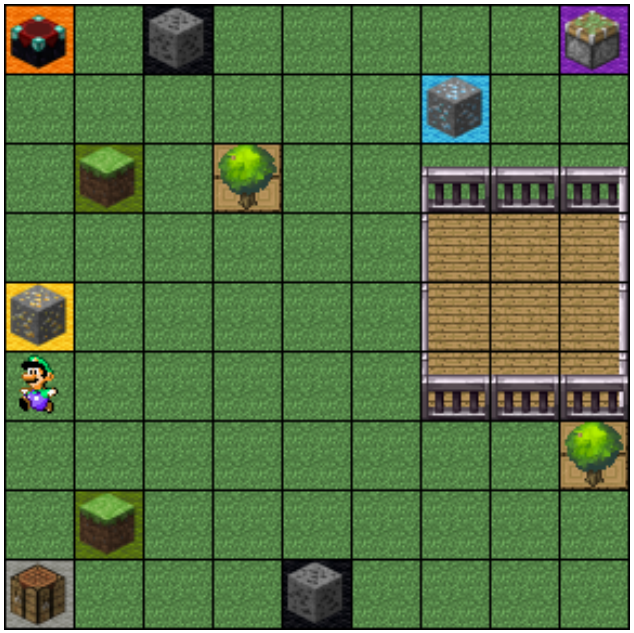


Detected events

`none`

$\varphi =$ **eventually** `used_factory` and **eventually** `got_gold`

Example

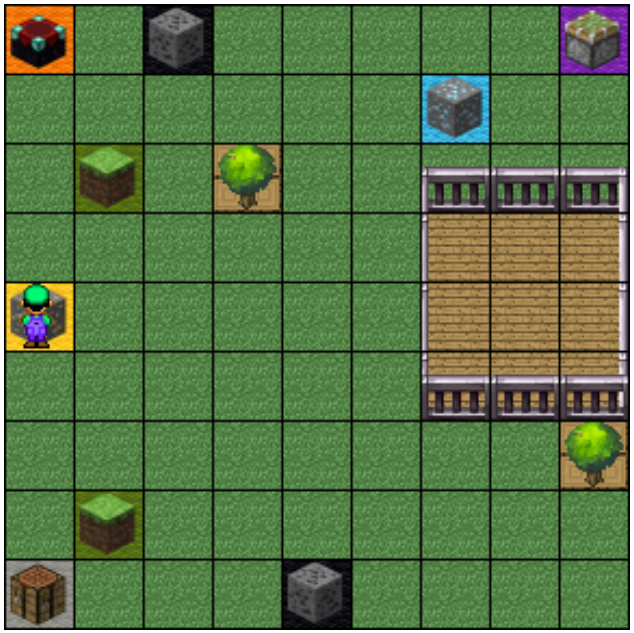


Detected events

none

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

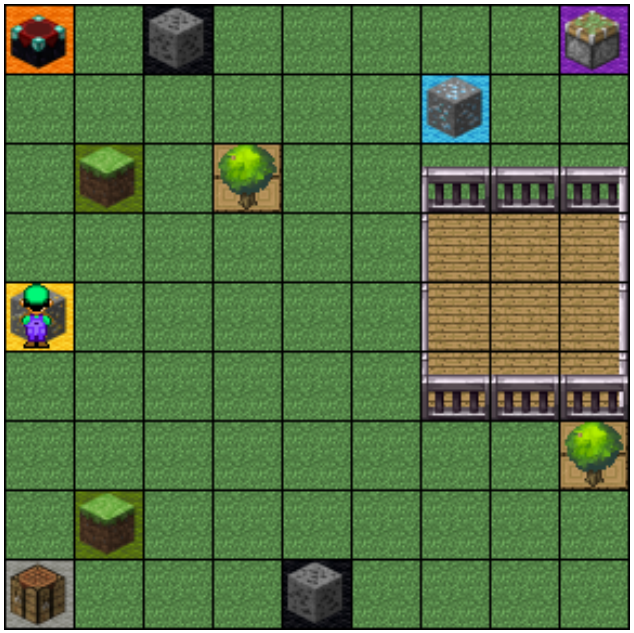


Detected events

got_gold

$\varphi =$ **eventually** used_factory and **eventually** got_gold

Example

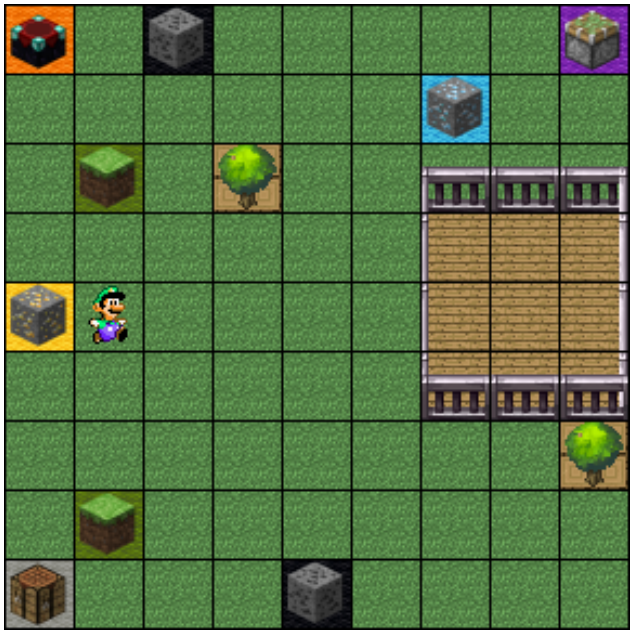


Detected events

`got_gold`

$\varphi =$ **eventually** `used_factory`

Example

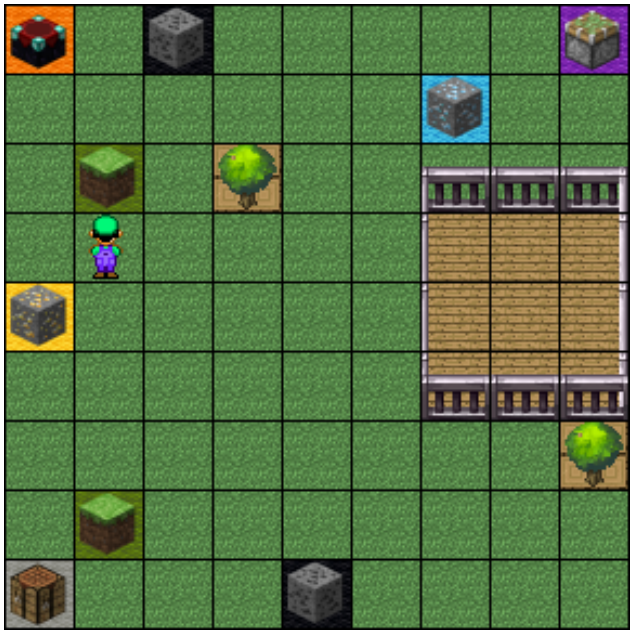


Detected events

none

$\varphi =$ **eventually** `used_factory`

Example

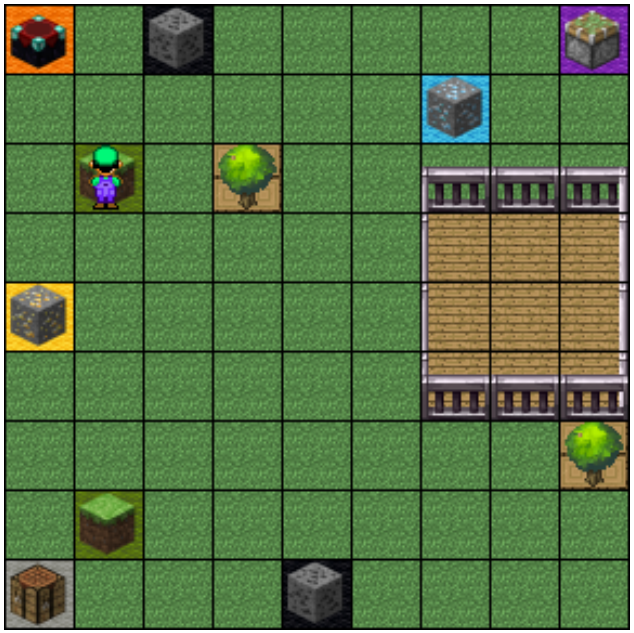


Detected events

none

$\varphi =$ **eventually** `used_factory`

Example

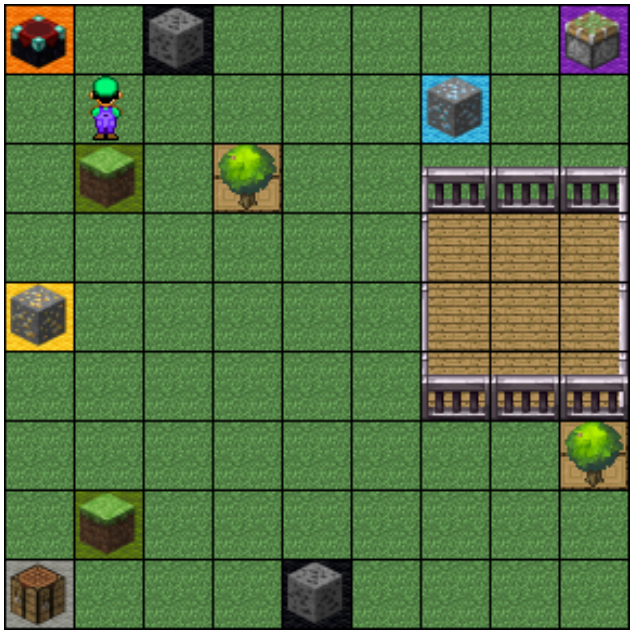


Detected events

`got_grass`

$\varphi =$ **eventually** `used_factory`

Example

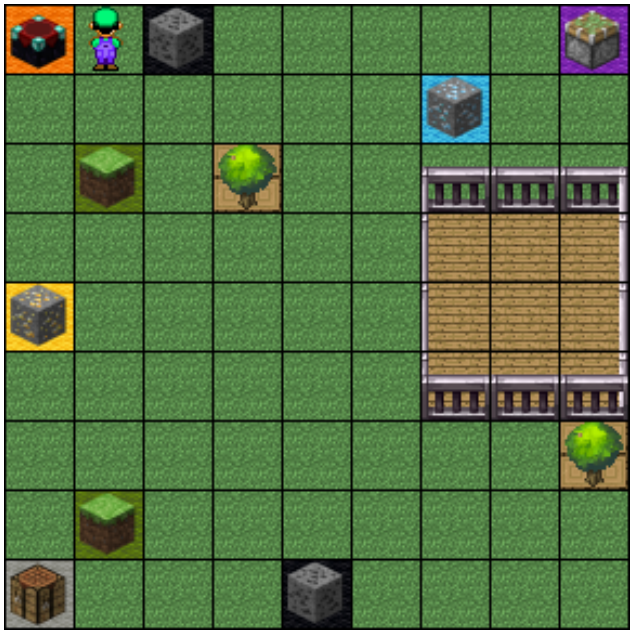


Detected events

none

$\varphi =$ **eventually** `used_factory`

Example

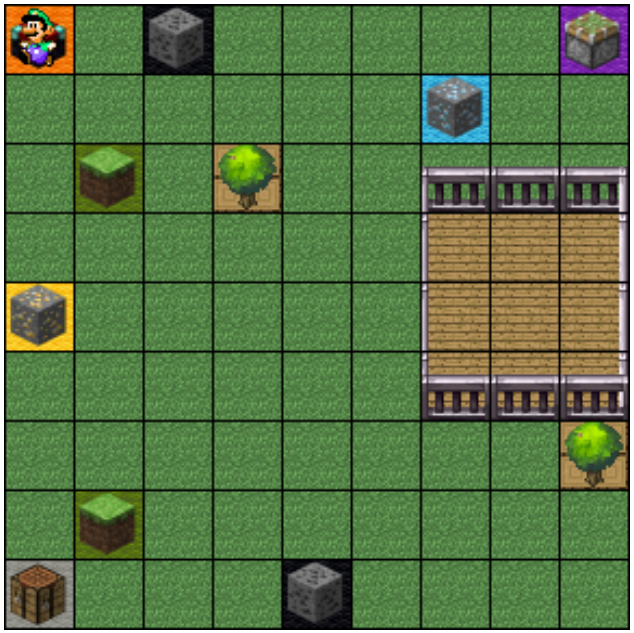


Detected events

none

$\varphi =$ **eventually** `used_factory`

Example

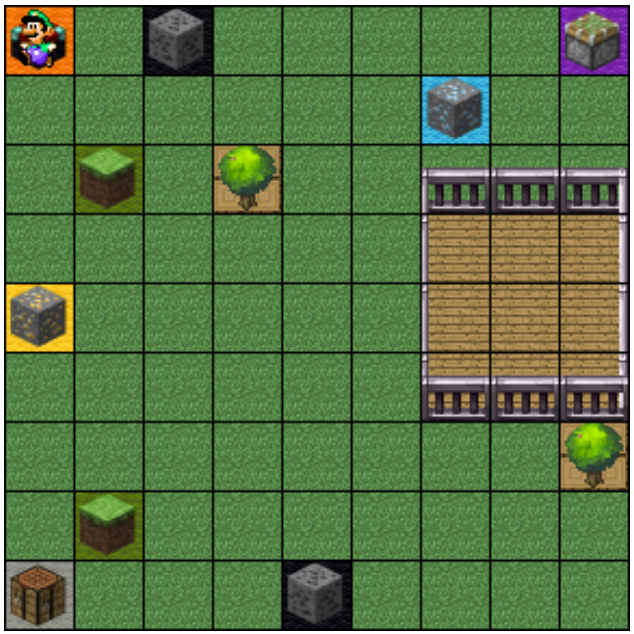


Detected events

`used_factory`

$\varphi =$ **eventually** `used_factory`

Example

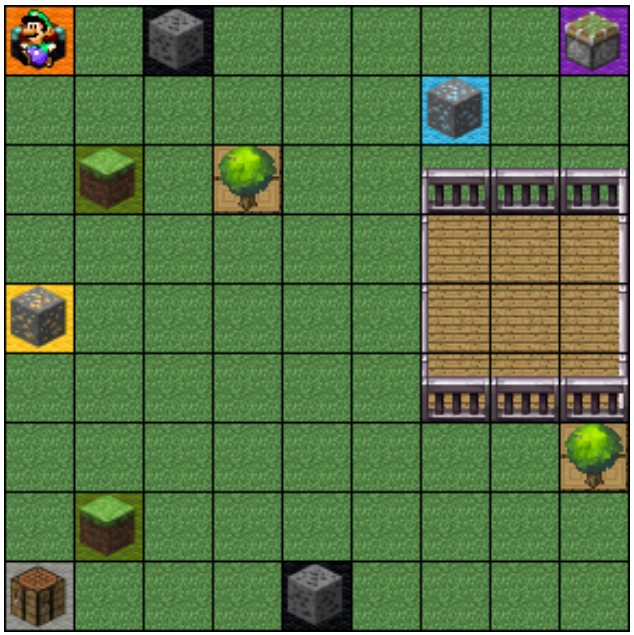


Detected events

`used_factory`

$\varphi =$ true (+1 reward)

Example

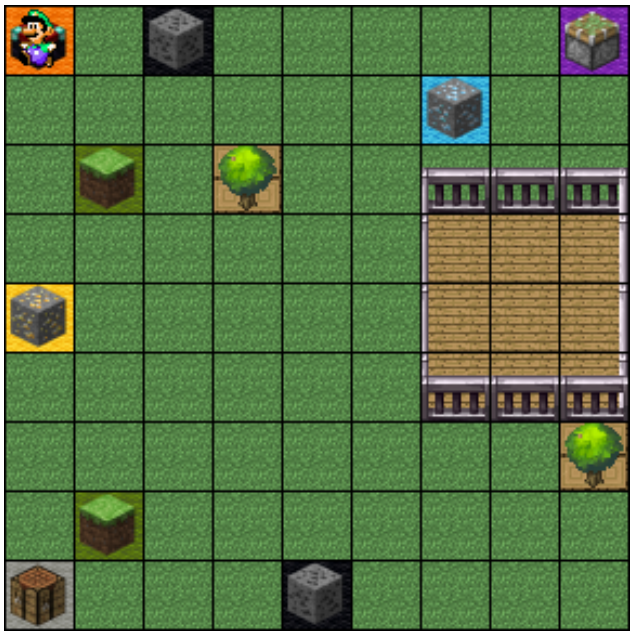


Detected events

`used_factory`

LTL formulas $\rightarrow$ Rewards

Example

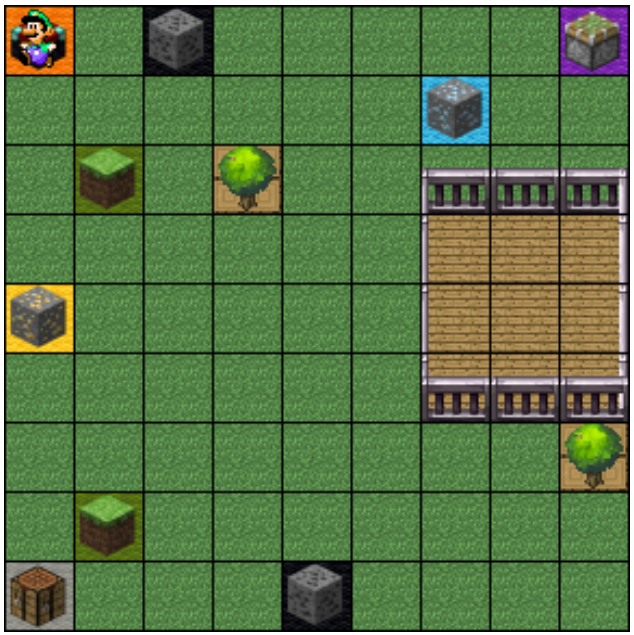


Detected events

`used_factory`

(φ can be learned using standard RL)

Example



Detected events

`used_factory`

We can do better than this!

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - **Why (and how) LPOPL speeds up learning of multiple tasks.**
4. Results
5. Concluding remarks

LPOPL: An example

Suppose Luigi has to learn two tasks:

LPOPL: An example

Suppose Luigi has to learn two tasks:

$$\varphi_1 = \textbf{eventually}(\texttt{got\_iron} \text{ and } \textbf{eventually } \texttt{used\_factory}) \text{ and } \textbf{eventually } \texttt{got\_gold}$$

$$\varphi_2 = \textbf{eventually}([\texttt{got\_grass} \text{ or } \texttt{got\_wood}] \text{ and } \textbf{eventually } \texttt{used\_factory})$$

LPOPL: An example

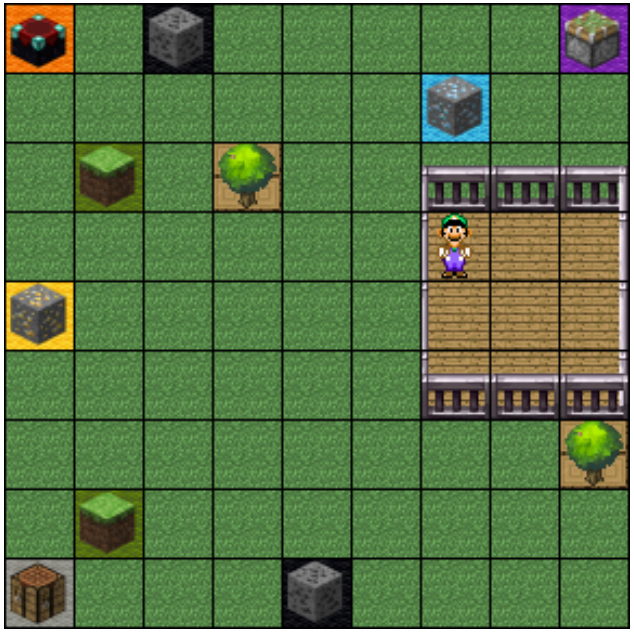
Suppose Luigi has to learn two tasks:

$\varphi_1 =$ **eventually**(`got_iron` and **eventually** `used_factory`) and **eventually** `got_gold`

$\varphi_2 =$ **eventually**([`got_grass` or `got_wood`] and **eventually** `used_factory`)

... and begins by trying to solve φ_1

LPOPL: An example



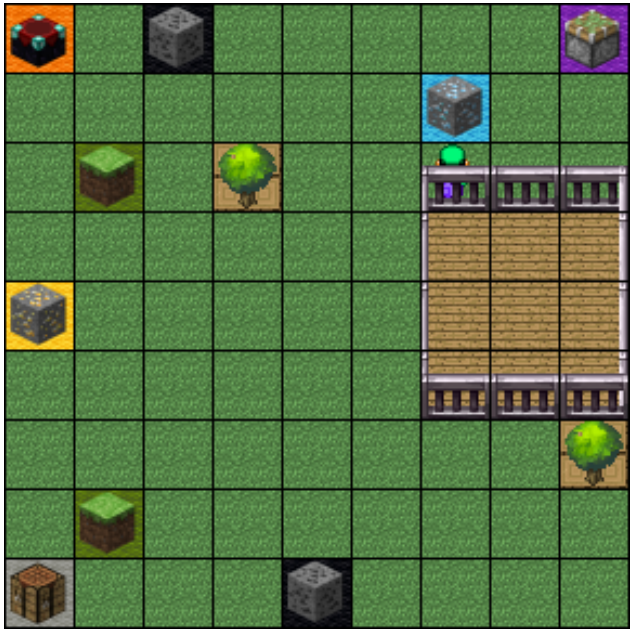
Detected events

at_shelter

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example

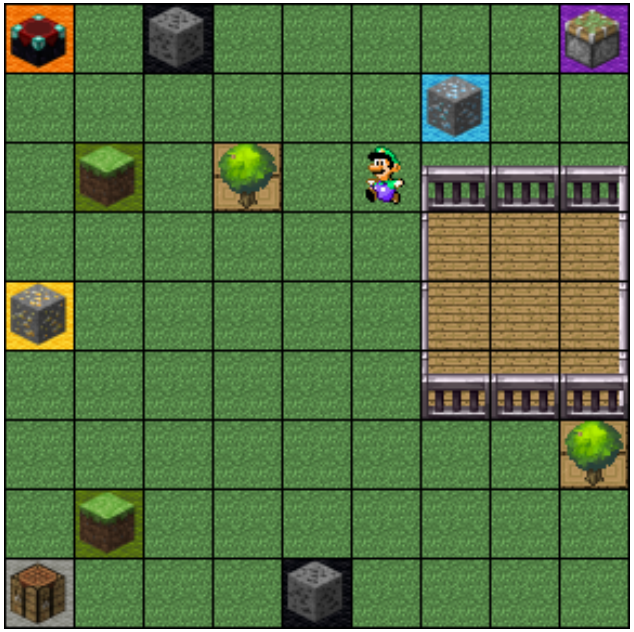


Detected events
none

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example

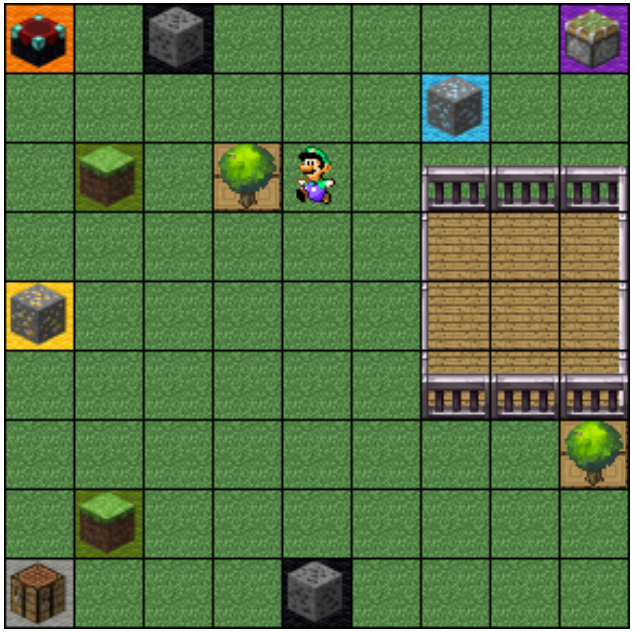


Detected events
none

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example

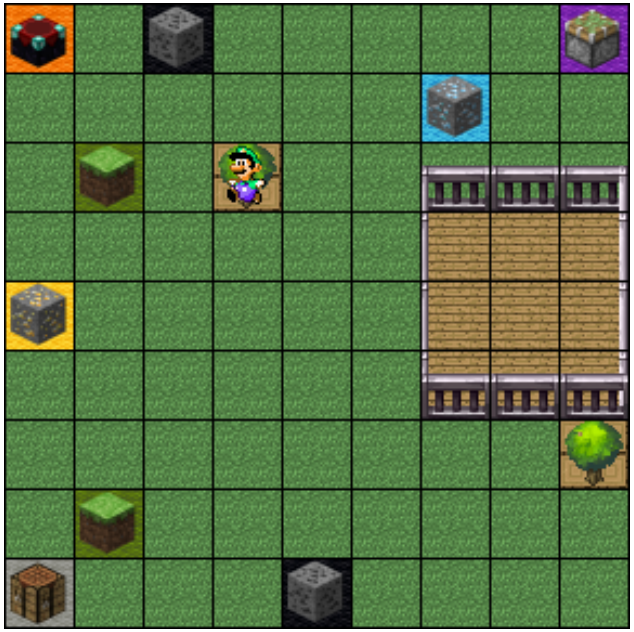


Detected events
none

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example

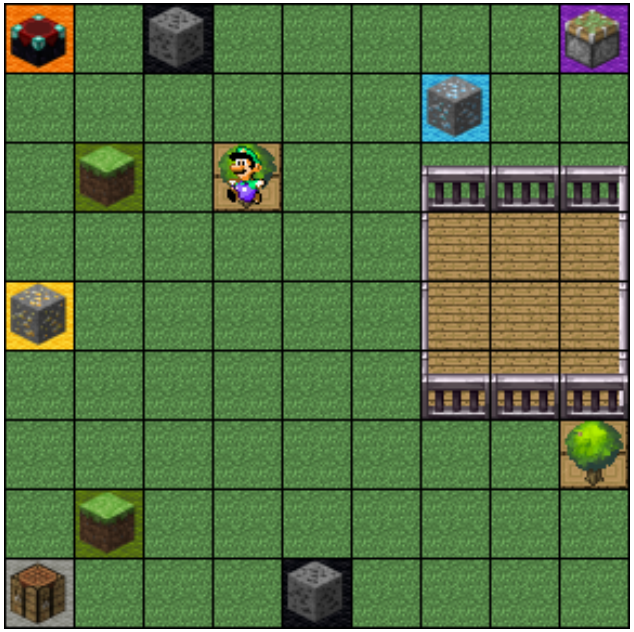


Detected events
got_wood

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example

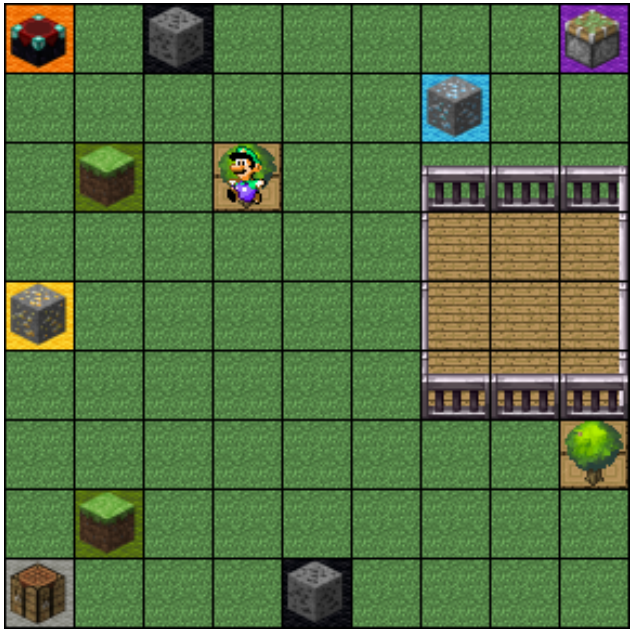


Detected events
got_wood

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL: An example



Detected events
got_wood

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory) and **eventually** got_gold

$\varphi_2 =$ **eventually**([got_grass or got_wood] and **eventually** used_factory)

LPOPL learns all the tasks in parallel!

LPOPL overview

Step 1: Decompose the tasks into subtasks using LTL progression.

LPOPL overview

Step 1: Decompose the tasks into subtasks using LTL progression.

$\varphi_1 =$ **eventually**(`got_iron` and **eventually** `used_factory`)
and **eventually** `got_gold`
$\varphi_2 =$ **eventually**([`got_grass` or `got_wood`]
and **eventually** `used_factory`)

LPOPL overview

Step 1: Decompose the tasks into subtasks using LTL progression.

$\varphi_1 =$ **eventually**(got_iron and **eventually** used_factory)
and **eventually** got_gold
$\varphi_2 =$ **eventually**([got_grass or got_wood]
and **eventually** used_factory)
$\varphi_3 =$ **eventually**(got_iron and **eventually** used_factory)
$\varphi_4 =$ **eventually** used_factory and **eventually** got_gold
$\varphi_5 =$ **eventually** used_factory
$\varphi_6 =$ **eventually** got_gold
$\varphi_7 =$ true

LPOPL overview

Step 1: Decompose the tasks into subtasks using LTL progression.

$$
\begin{aligned}
\varphi_1 &= \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \\
&\quad \wedge \Diamond\texttt{got\_gold} \\
\varphi_2 &= \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \\
&\quad \wedge \Diamond\texttt{used\_factory}) \\
\varphi_3 &= \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \\
\varphi_4 &= \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold} \\
\varphi_5 &= \Diamond\texttt{used\_factory} \\
\varphi_6 &= \Diamond\texttt{got\_gold} \\
\varphi_7 &= \text{true}
\end{aligned}
$$

LPOPL overview

Step 1: Decompose the tasks into subtasks using LTL progression.

$$
\begin{aligned}
\varphi_1 &= \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \\
&\quad \wedge \Diamond\texttt{got\_gold} \\
\varphi_2 &= \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \\
&\quad \wedge \Diamond\texttt{used\_factory}) \\
\varphi_3 &= \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \\
\varphi_4 &= \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold} \\
\varphi_5 &= \Diamond\texttt{used\_factory} \\
\varphi_6 &= \Diamond\texttt{got\_gold} \\
\varphi_7 &= \text{true}
\end{aligned}
$$

Step 2: Learn one policy per subtask using off-policy learning.

LPOPL using tabular q-learning

Standard q-learning update given experience (s, a, r, s'):

$$Q(s,a) \xleftarrow{\alpha} r + \gamma \max_{a'} Q(s', a')$$

... where $x \xleftarrow{\alpha} y$ is a shorthand notation for $x \leftarrow x + \alpha \cdot (y - x)$.

LPOPL using tabular q-learning

Standard q-learning update given experience (s, a, r, s'):

$$Q(s,a) \xleftarrow{\alpha} r + \gamma \max_{a'} Q(s', a')$$

... where $x \xleftarrow{\alpha} y$ is a shorthand notation for $x \leftarrow x + \alpha \cdot (y - x)$.

LPOPL update using q-learning given experience (s, a, s'):

$$Q_\varphi(s,a) \xleftarrow{\alpha} r_\varphi + \gamma \max_{a'} Q_{\varphi'}(s', a')$$

... where $\varphi' = \text{prog}(s', \varphi)$ and $r_\varphi = 1$ iff $\varphi \neq \varphi' = \text{true}$.

LPOPL using tabular q-learning

Standard q-learning update given experience (s, a, r, s'):

$$Q(s,a) \xleftarrow{\alpha} r + \gamma \max_{a'} Q(s', a')$$

... where $x \xleftarrow{\alpha} y$ is a shorthand notation for $x \leftarrow x + \alpha \cdot (y - x)$.

LPOPL update using q-learning given experience (s, a, s'):

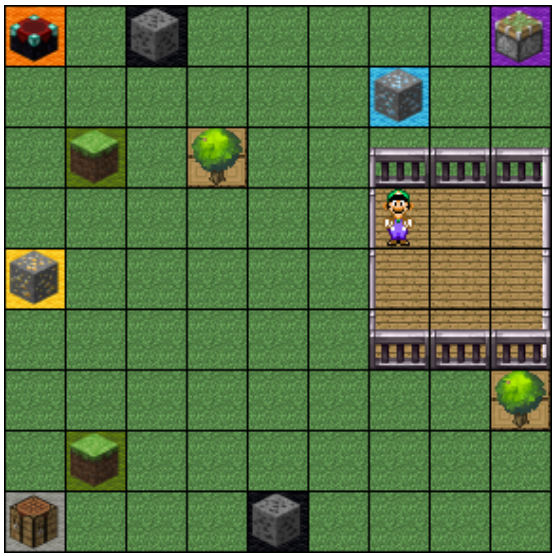
$$Q_\varphi(s,a) \xleftarrow{\alpha} r_\varphi + \gamma \max_{a'} Q_{\varphi'}(s', a')$$

... where $\varphi' = \text{prog}(s', \varphi)$ and $r_\varphi = 1$ iff $\varphi \neq \varphi' = \text{true}$.

Theorem

LPOPL using tabular q-learning converges to an optimal policy.

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

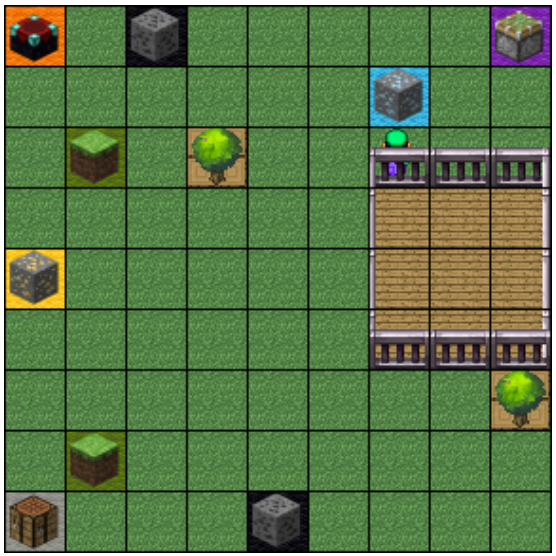
$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `at_shelter`

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

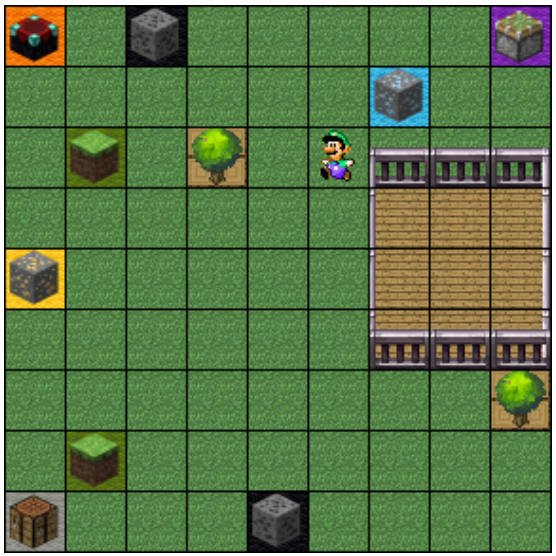
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

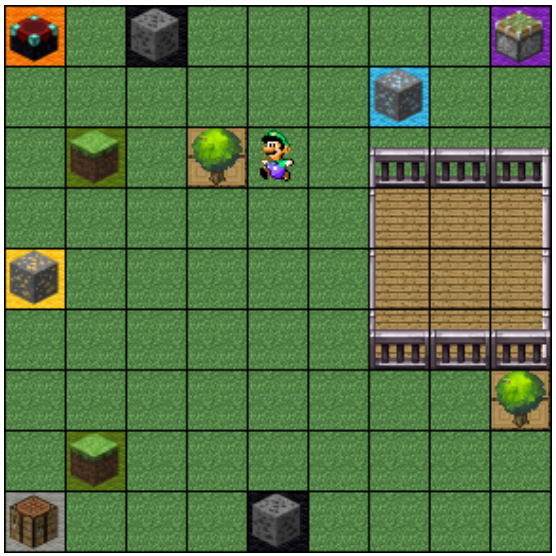
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

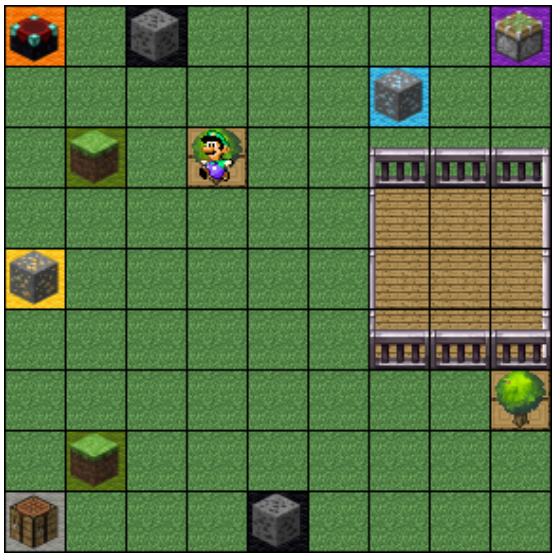
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_wood`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

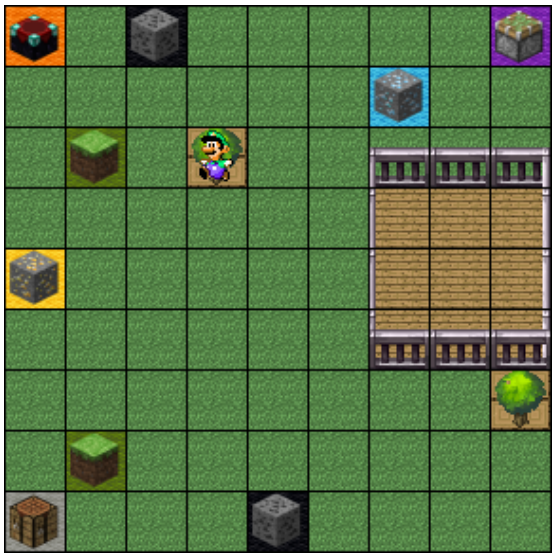
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_wood`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

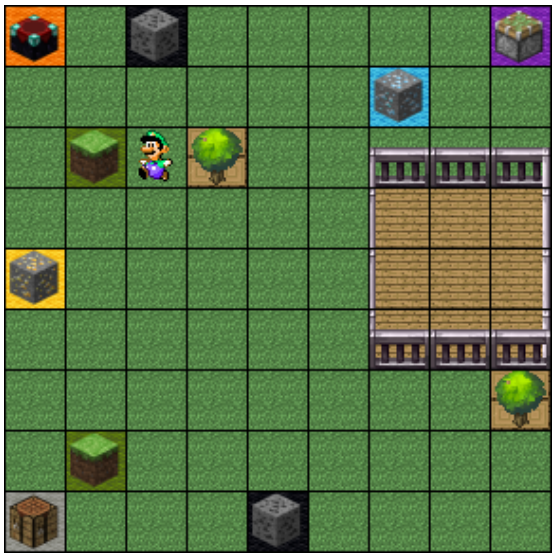
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

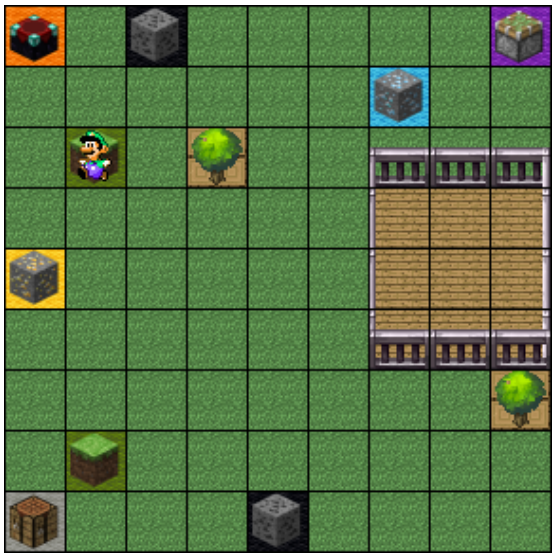
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_grass`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

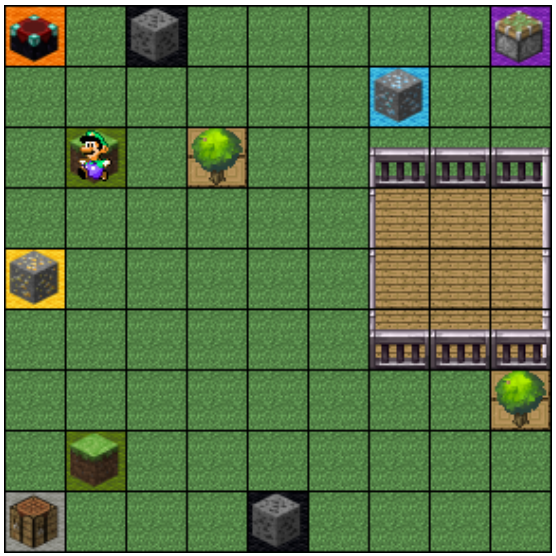
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_grass`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

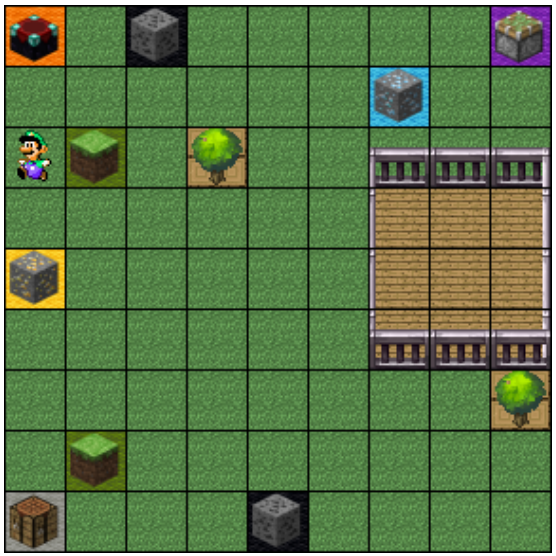
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

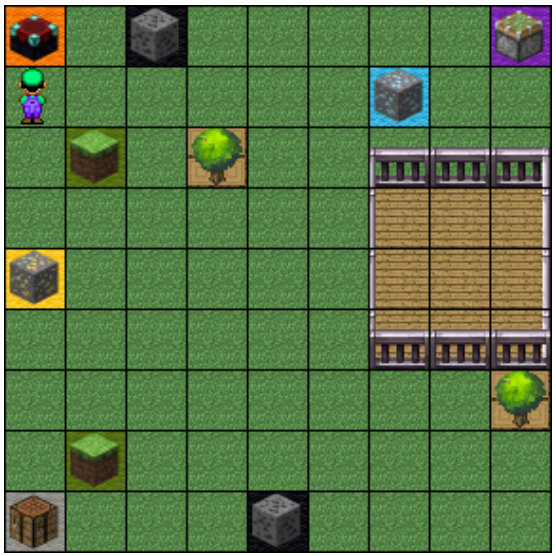
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

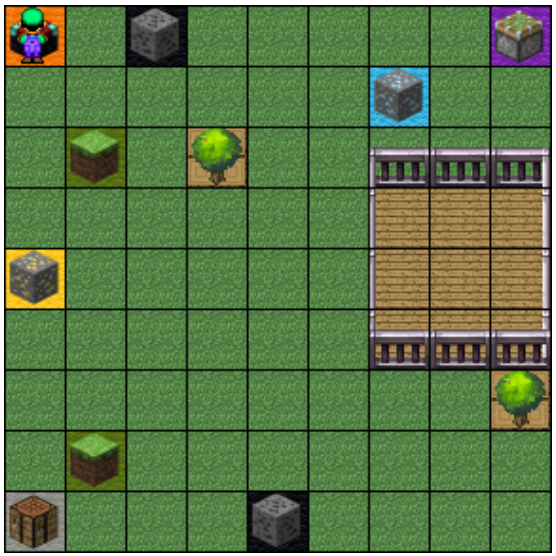
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `used_factory`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

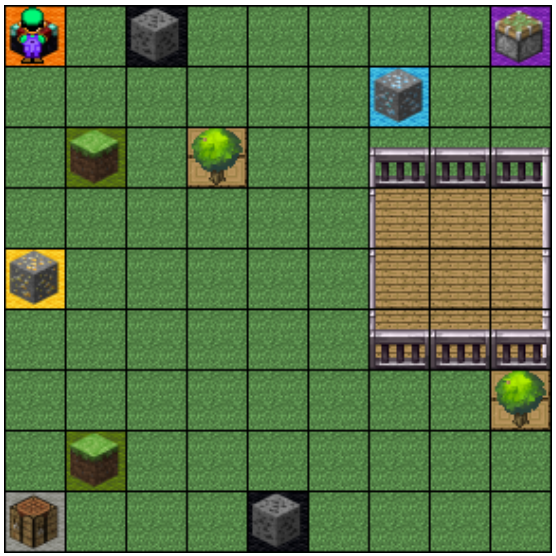
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} 1$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `used_factory`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

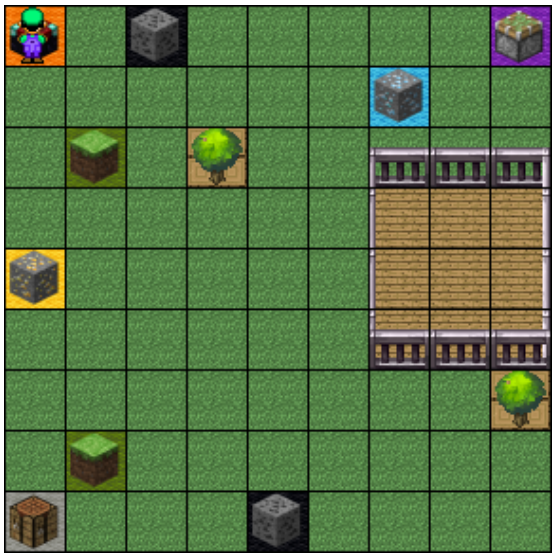
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} 1$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `used_factory`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

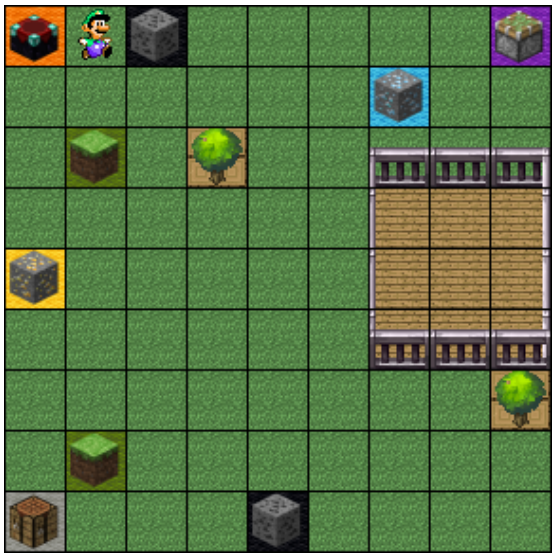
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} 1$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

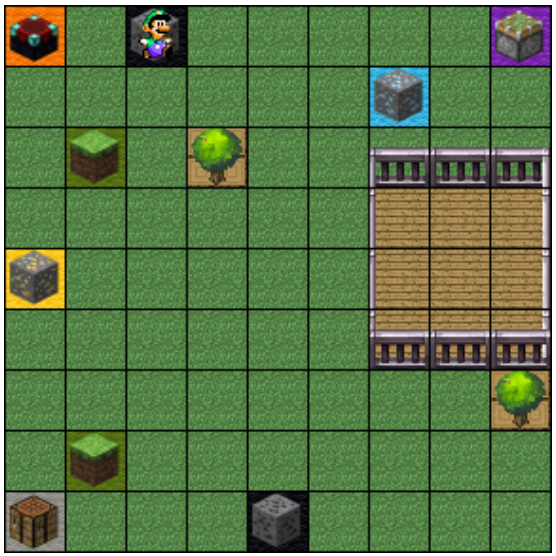
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_iron`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

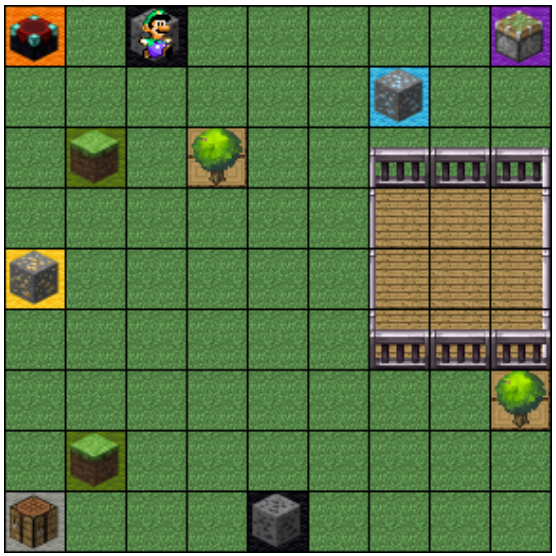
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_iron`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

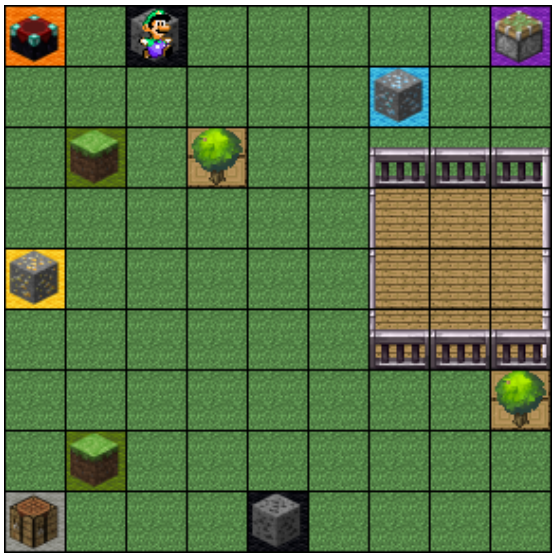
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_iron`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

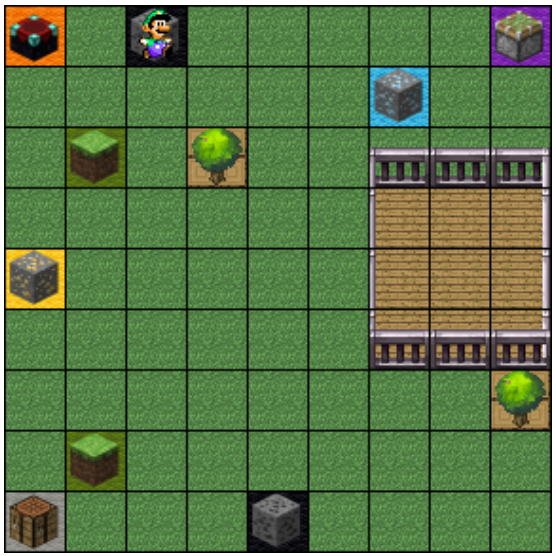
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `got_iron`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

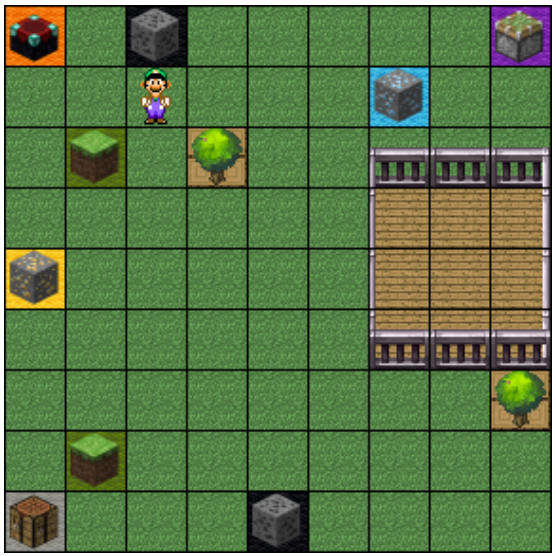
$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Example



Subtasks

$\varphi_1 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory}) \wedge \Diamond\texttt{got\_gold}$

$\varphi_2 = \Diamond([\texttt{got\_grass} \vee \texttt{got\_wood}] \wedge \Diamond\texttt{used\_factory})$

$\varphi_3 = \Diamond(\texttt{got\_iron} \wedge \Diamond\texttt{used\_factory})$

$\varphi_4 = \Diamond\texttt{used\_factory} \wedge \Diamond\texttt{got\_gold}$

$\varphi_5 = \Diamond\texttt{used\_factory}$

$\varphi_6 = \Diamond\texttt{got\_gold}$

Detected events: `none`

Q-updates

$Q_{\varphi_1}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_1}(s',a')$

$Q_{\varphi_2}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_2}(s',a')$

$Q_{\varphi_3}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_3}(s',a')$

$Q_{\varphi_4}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_4}(s',a')$

$Q_{\varphi_5}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_5}(s',a')$

$Q_{\varphi_6}(s,a) \xleftarrow{\alpha} \gamma \max_{a'} Q_{\varphi_6}(s',a')$

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. **Results**
5. Concluding remarks

Experiments

Goal

- Study LPOPL + DQN
- Compare with standard RL
- Compare with alternative decomposition methods for RL

Experiments

Goal

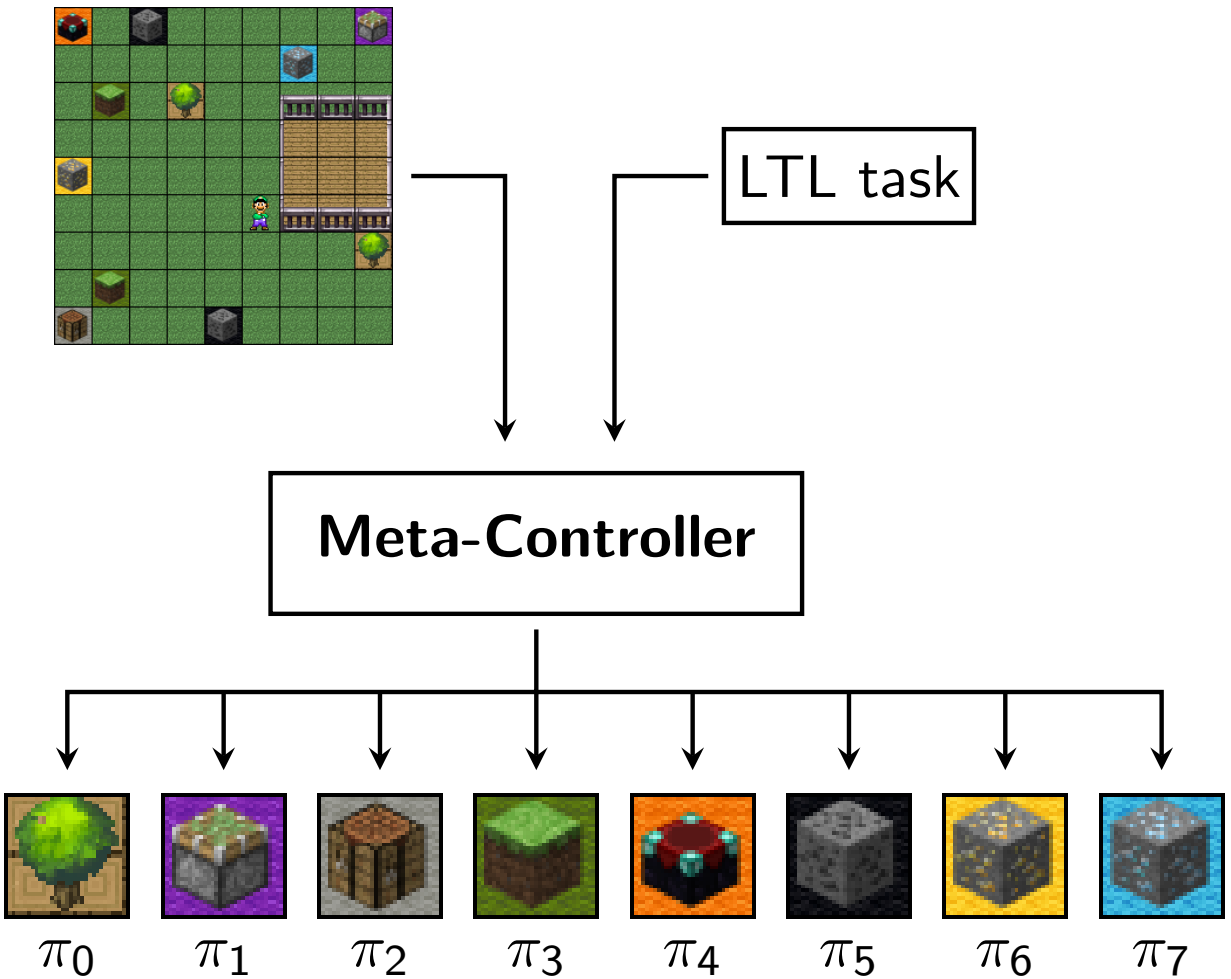
- Study LPOPL + DQN
- Compare with standard RL
- Compare with alternative decomposition methods for RL

Baselines

- DQN-L: Standard DQN[1] (no decomposition).
- HRL-E: Hierarchical Deep RL proposed by Kulkarni et al.[2]
- HRL-L: HRL-E but exploiting LTL to prune *useless* options.

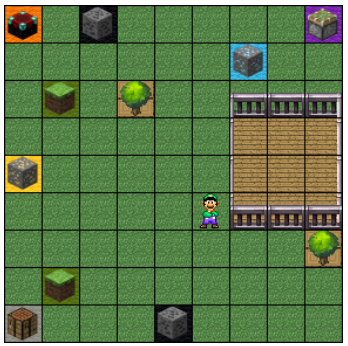
[1] Human-level control through deep reinforcement learning (Nature-15)
[2] HDRL: Integrating Temporal Abstraction and Intrinsic Motivation (NIPS-16)

Baselines: HRL-E



(π_i learns to reach object i)

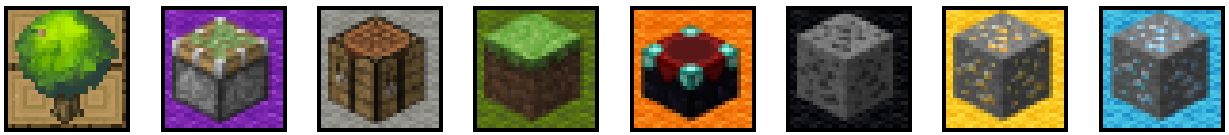
Baselines: HRL-L



$\Diamond$got_wood $\vee$ $\Diamond$got_gold

LTL task

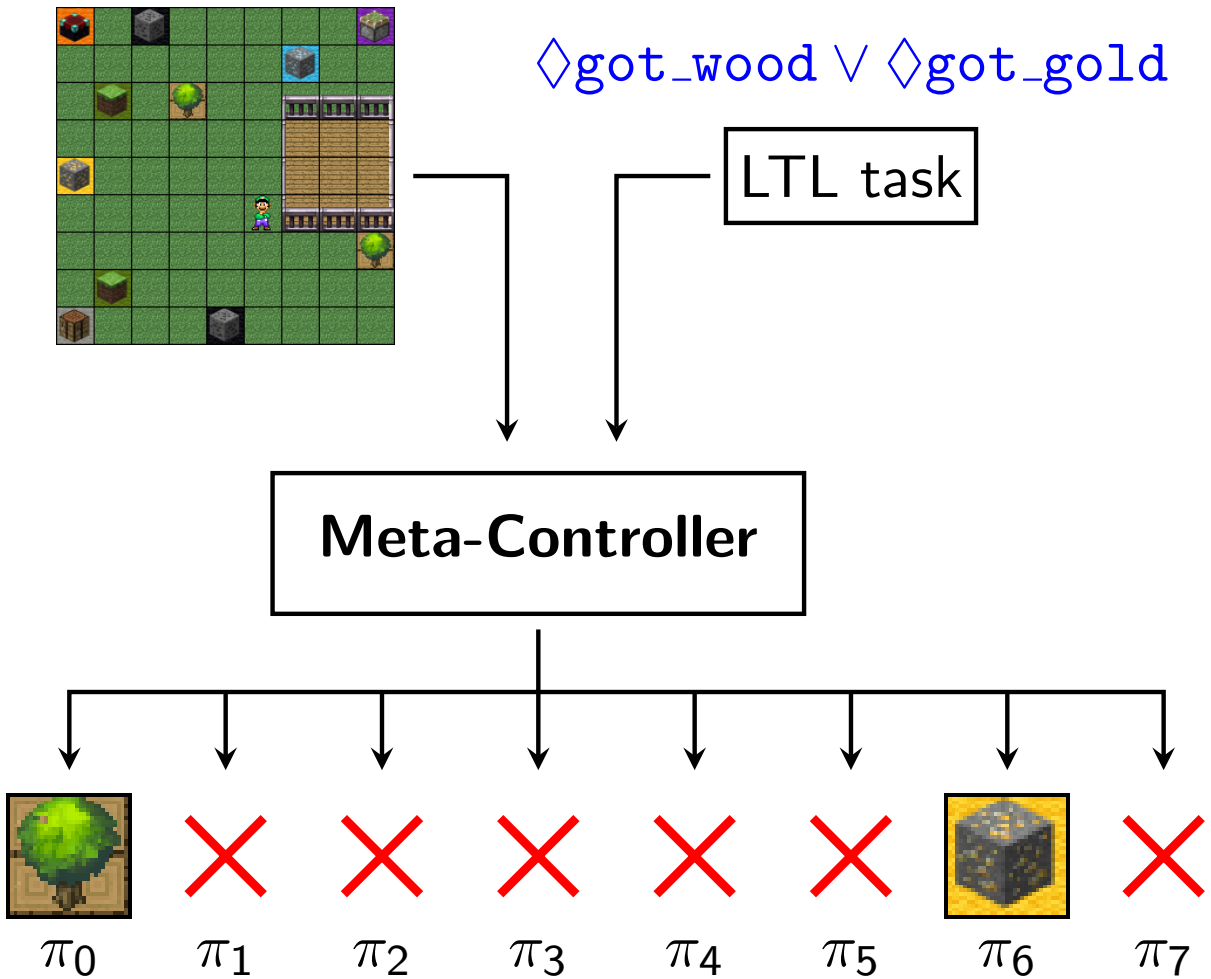
Meta-Controller



π_0 π_1 π_2 π_3 π_4 π_5 π_6 π_7

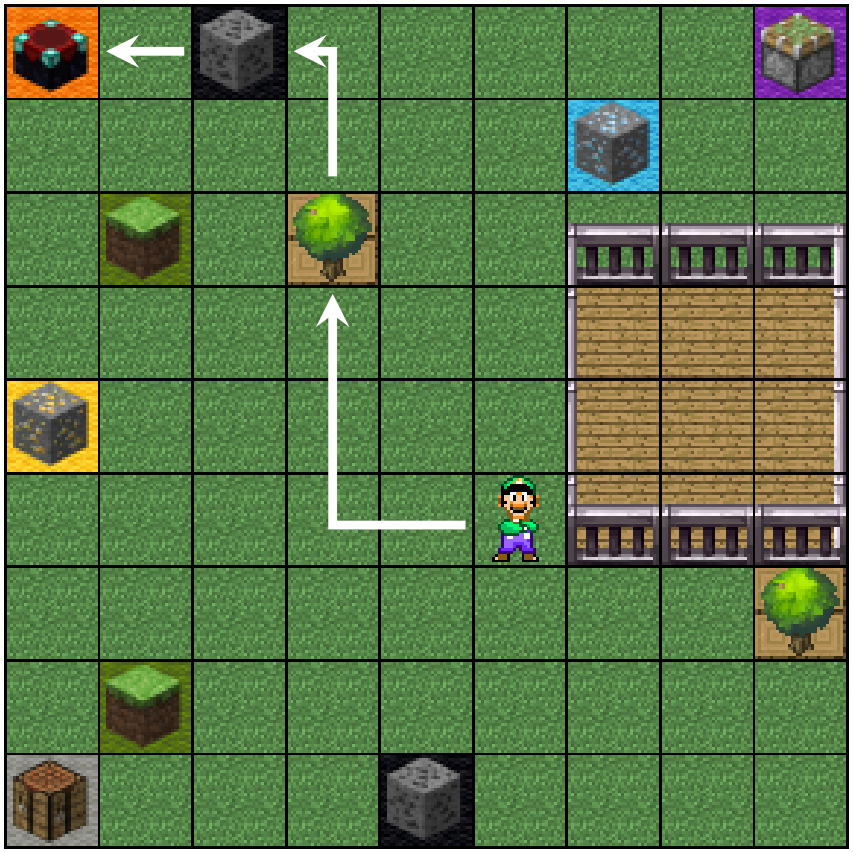
(π_i learns to reach object i)

Baselines: HRL-L

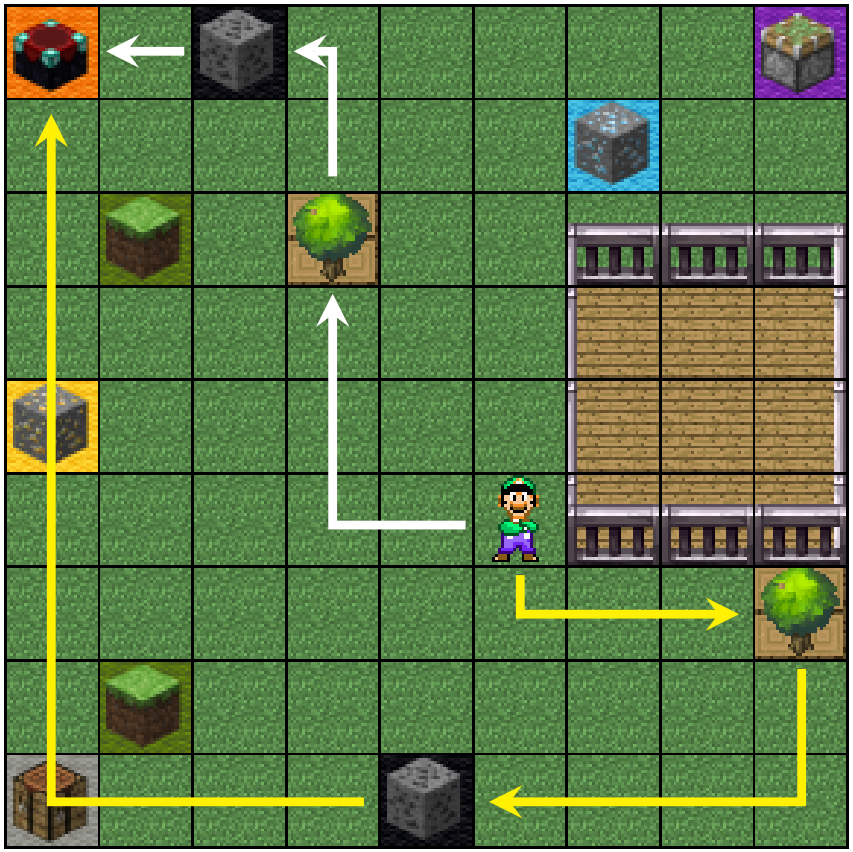


(π_i learns to reach object i)

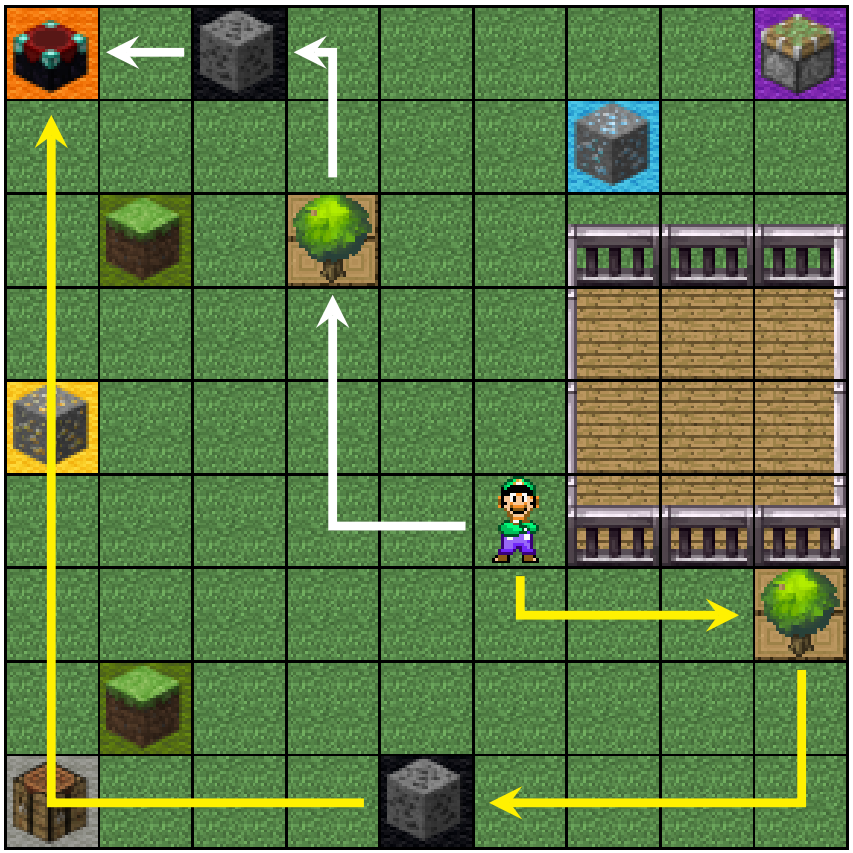
Hierarchical RL might converge to suboptimal policies



Hierarchical RL might converge to suboptimal policies



Hierarchical RL might converge to suboptimal policies



We tested over 5 random grids and 5 adversarial grids.

Experiment 1: subgoal sequences

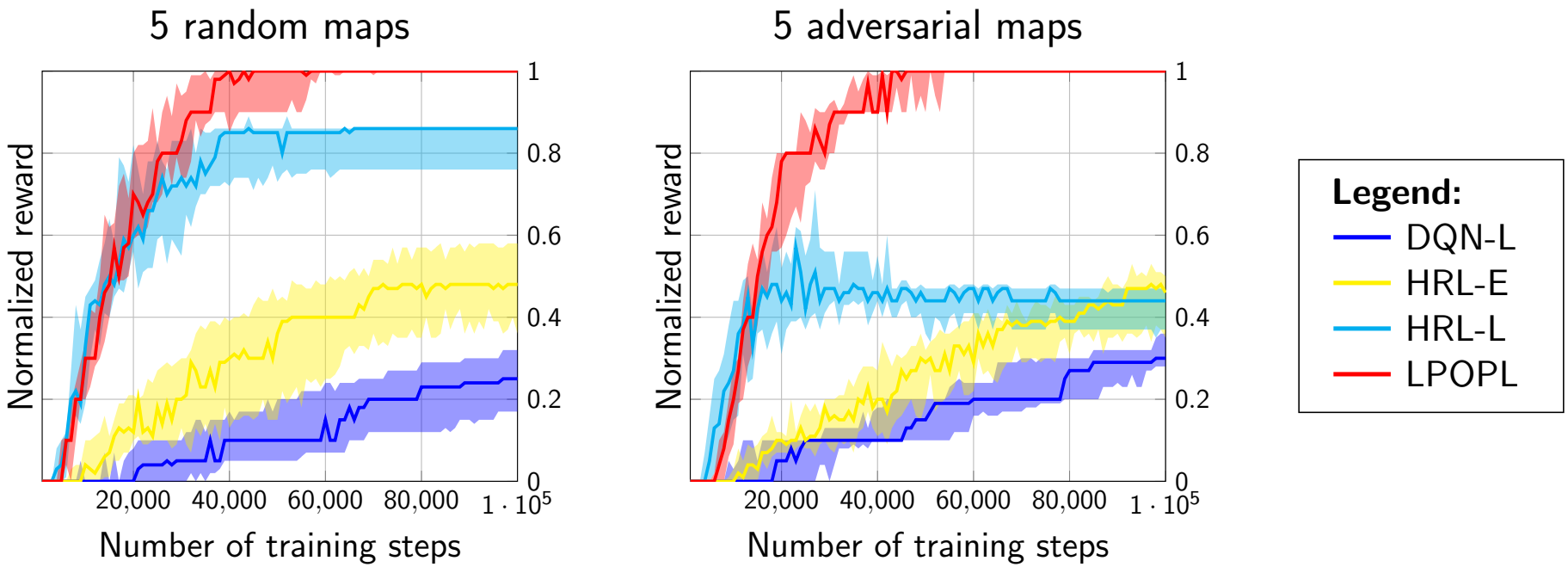
Description

10 tasks defined as sequence of subgoals (Andreas et al., 2017)
e.g. get iron, then get wood, then use factory.

Experiment 1: subgoal sequences

Description

10 tasks defined as sequence of subgoals (Andreas et al., 2017)
e.g. get iron, then get wood, then use factory.



Experiment 2: Interleaving subtasks

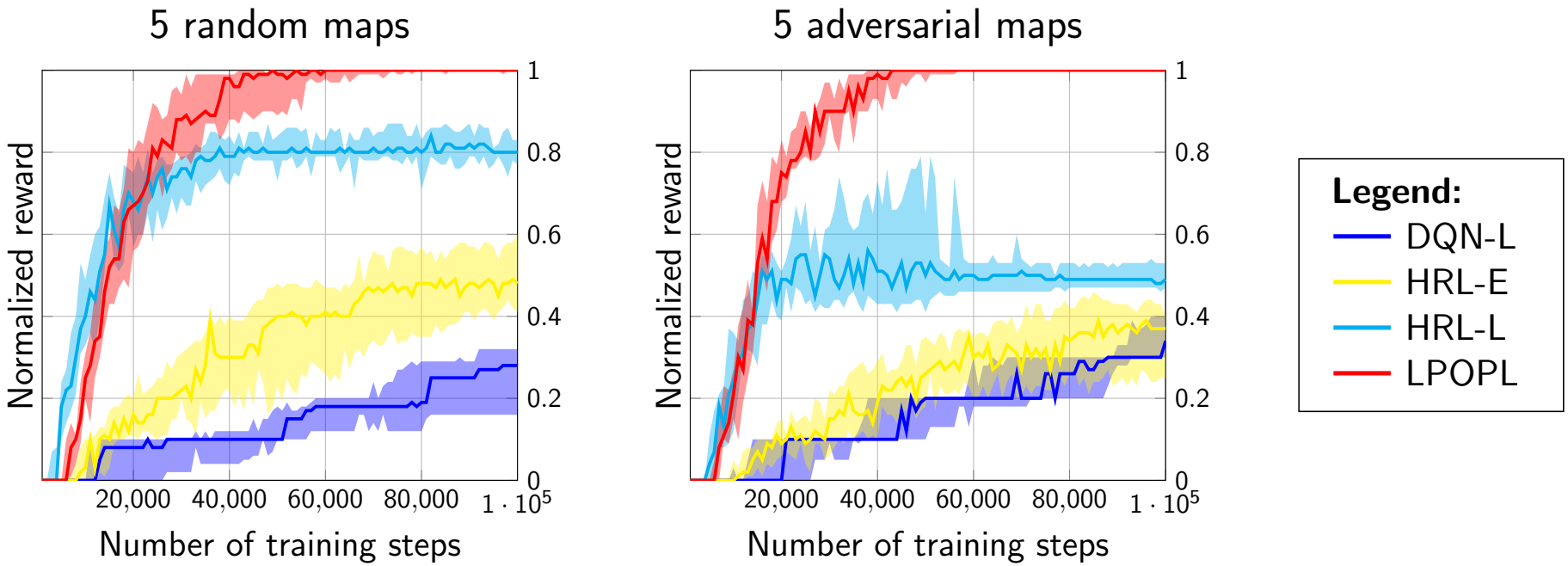
Description

Same set of 10 tasks but removing unnecessary order constraints.
e.g. (get iron and get wood), then use factory.

Experiment 2: Interleaving subtasks

Description

Same set of 10 tasks but removing unnecessary order constraints. e.g. (get iron and get wood), then use factory.



Experiment 3: Safety constraints

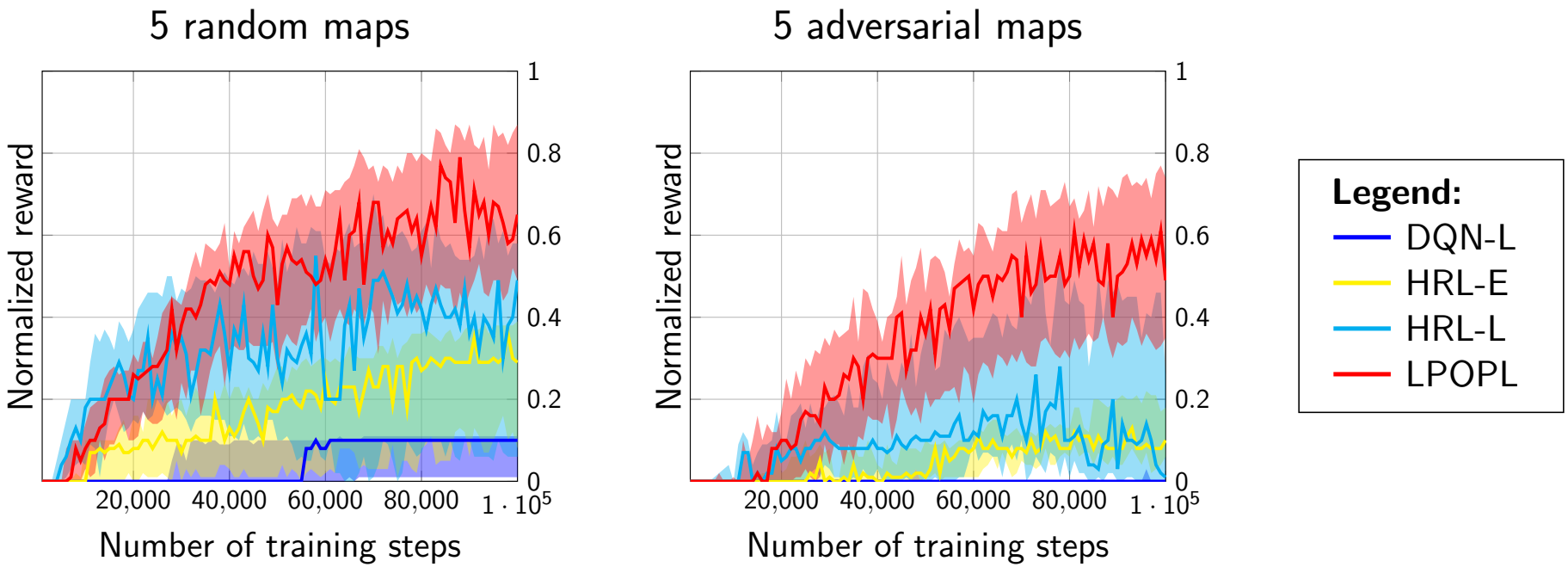
Description

Same set of 10 tasks but including the safety constraint of being at the shelter during the night.

Experiment 3: Safety constraints

Description

Same set of 10 tasks but including the safety constraint of being at the shelter during the night.



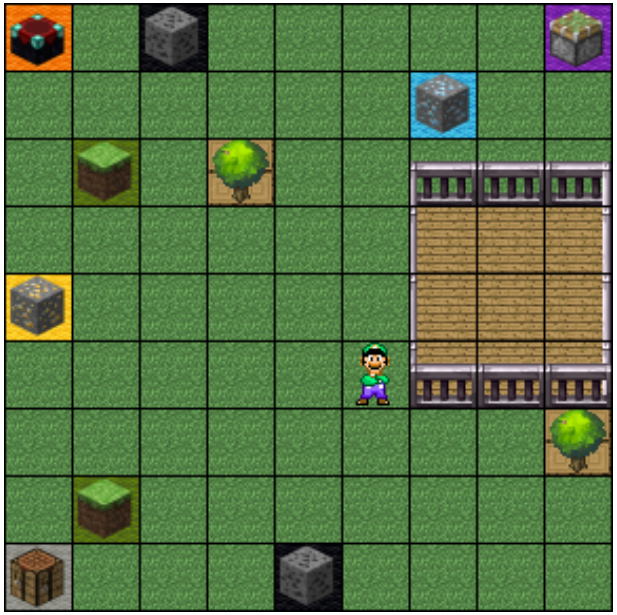
Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. Concluding remarks

Outline

1. Motivation
2. Related work
3. Approach:
   - Why (and how) we use LTL for specifying tasks in RL.
   - Why (and how) LPOPL speeds up learning of multiple tasks.
4. Results
5. **Concluding remarks**

Concluding remarks



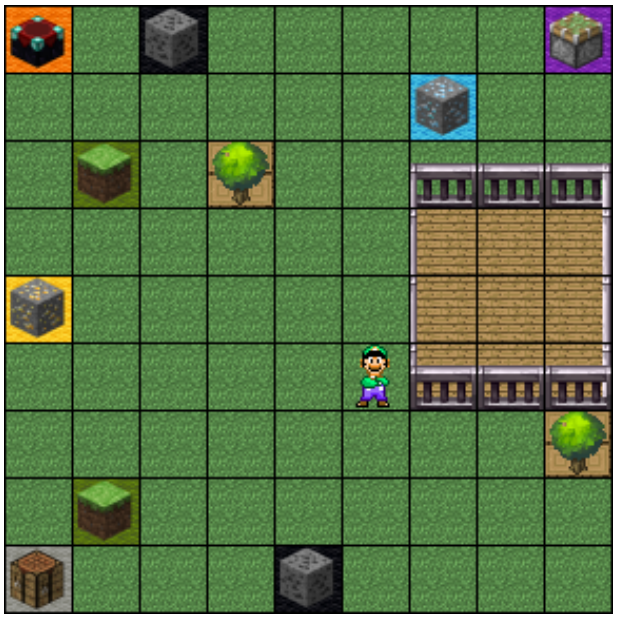
Problem

How to tell an RL agent what to do.

Our approach

Define tasks using LTL.
Decompose tasks using LTL progression.
Learn subtasks using off-policy RL.

Concluding remarks



Problem

How to tell an RL agent what to do.

Our approach

Define tasks using LTL.
Decompose tasks using LTL progression.
Learn subtasks using off-policy RL.

Experiments:

- LPOPL outperformed HRL and DQN over a variety of tasks.
- `https://bitbucket.org/RToroIcarte/lpopl`

Thanks!