Thirty-eighth International Conference on Machine Learning - ICML 2021 (Jul 18-24)

LTL2Action: Generalizing LTL Instructions for Multi-Task RL

Pashootan Vaezipoor[*]

Andrew Li[*]

Rodrigo Toro Icarte

Sheila McIlraith

University of Toronto and Vector Institute

UNIVERSITY OF TORONTO

VECTOR INSTITUTE | INSTITUT VECTEUR

SCHWARTZ REISMAN INSTITUTE FOR TECHNOLOGY AND SOCIETY

CIFAR

Motivation

How can humans specify high-level objectives to an RL agent?

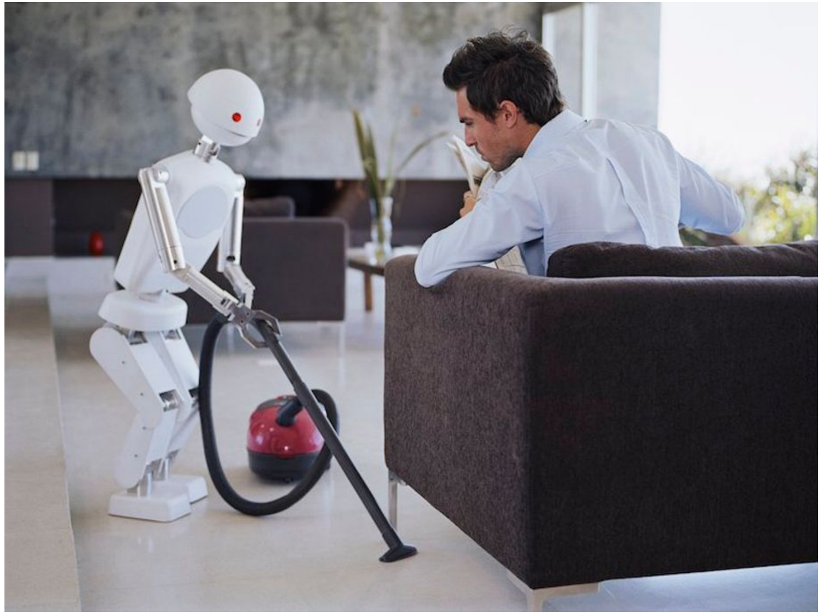


Photo: Javier Pierin (Getty Images)

Motivation

How can humans specify high-level objectives to an RL agent?

via Reward Function?

Infeasible for humans to program for every possible task.

Motivation

How can humans specify high-level objectives to an RL agent?

via Reward Function?

Infeasible for humans to program for every possible task.

via Natural Language?

Unclear what reward to optimize.

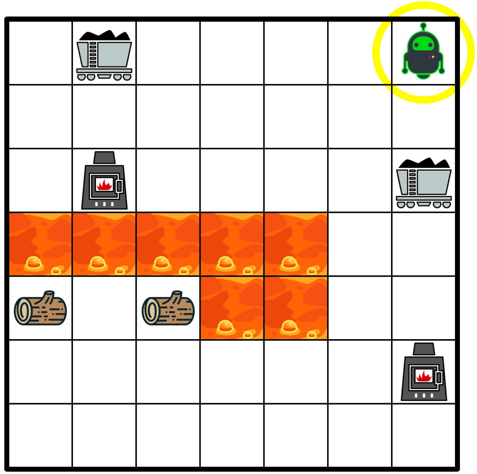
This Work

- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*

This Work

- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*
  - Expressive temporally extended goals

```
eventually (pickup_coal or pickup_wood)
    and (always (not on_lava))
```



This Work

- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*
    - Expressive temporally extended goals
    - **Automatic mapping** *Instruction* ➔ *Reward*

This Work

- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*
  - Expressive temporally extended goals
  - **Automatic mapping** *Instruction* ➔ *Reward*
  - Procedural sampling from **> 10^{39}** unique LTL instructions

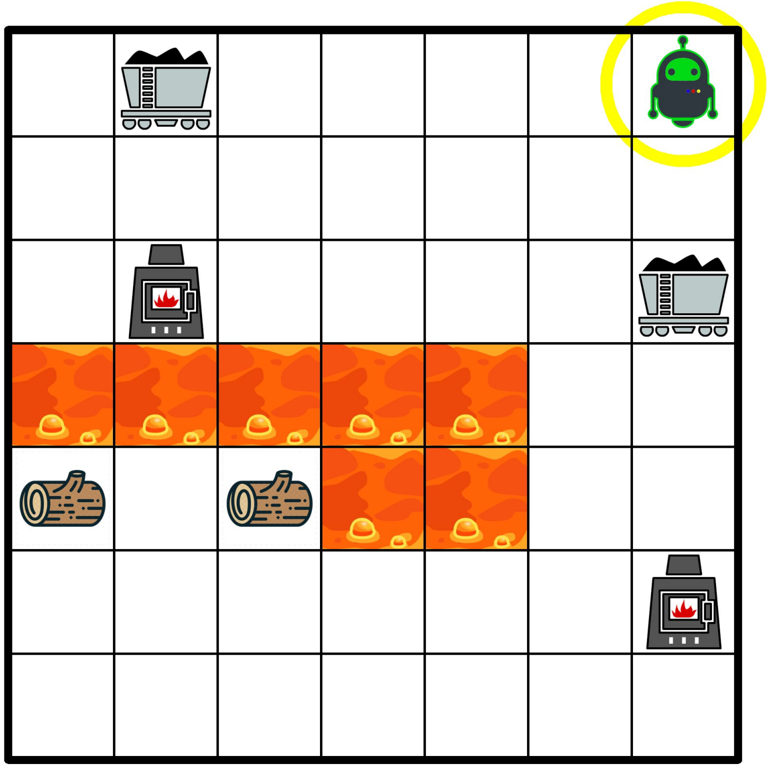
This Work

- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*
  - Expressive temporally extended goals
  - **Automatic mapping** *Instruction* ➔ *Reward*
  - Procedural sampling from **> 10^{39}** unique LTL instructions
- **Theoretical** benefits
  - **non-myopic** composition
  - **non-Markovian** rewards
- **Empirical** benefits
  - **Discrete** & **Continuous** (MuJoCo) environments

This Work

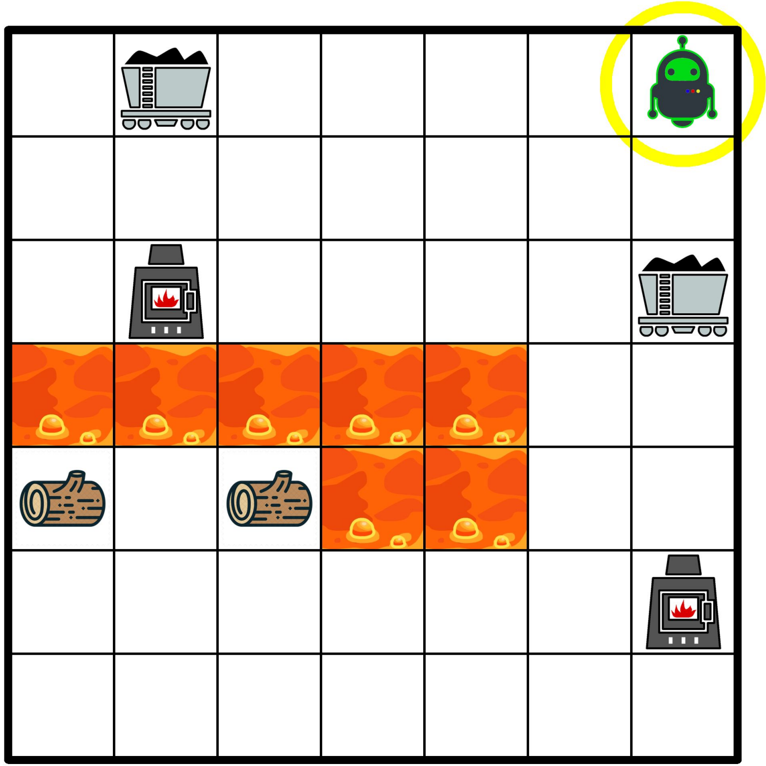
- Formal language of Linear Temporal Logic (LTL) *[Pnueli, 1977]*
  - Expressive temporally extended goals
  - **Automatic mapping** *Instruction* ➔ *Reward*
  - Procedural sampling from > $\mathbf{10^{39}}$ unique LTL instructions
- **Theoretical** benefits
  - **non-myopic** composition
  - **non-Markovian** rewards
- **Empirical** benefits
  - **Discrete** & **Continuous** (MuJoCo) environments
  - **Zero-shot generalization** to unseen and larger instructions

What tasks can be expressed in LTL?



Assume agent can detect primitive high-level events:

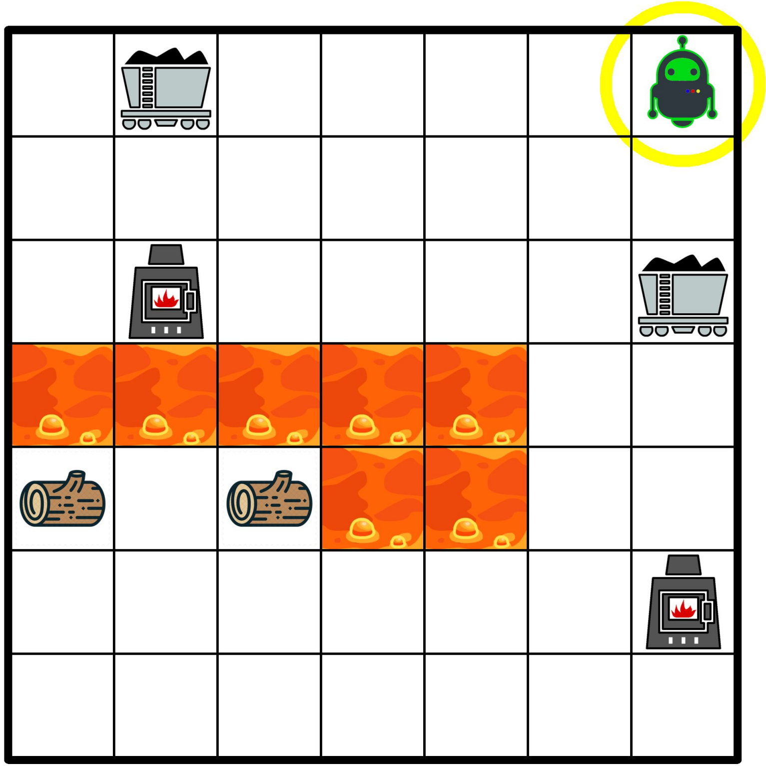
What tasks can be expressed in LTL?



Assume agent can detect primitive high-level events:

- `pickup_coal`
- `pickup_wood`

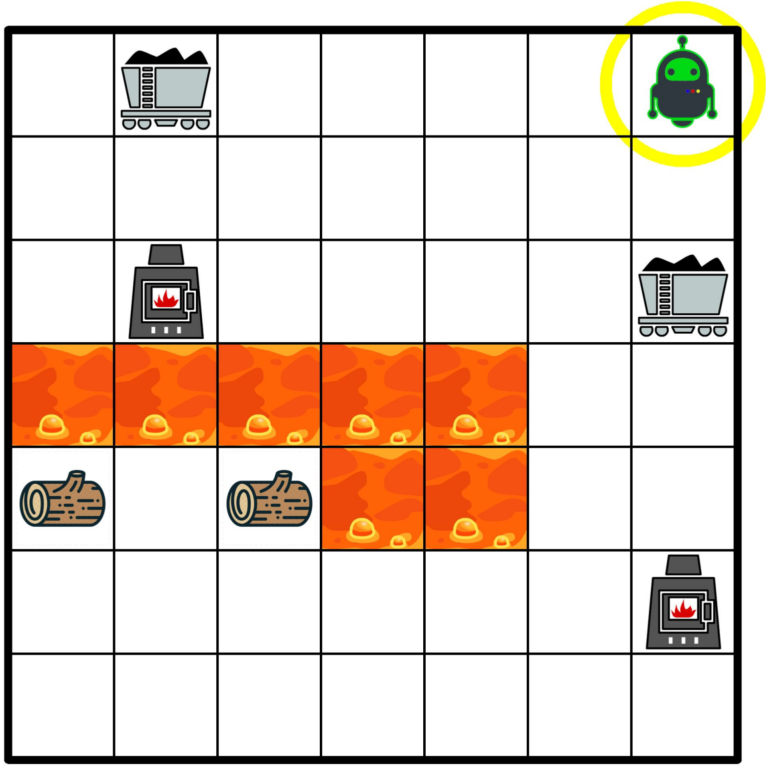
What tasks can be expressed in LTL?



Assume agent can detect primitive high-level events:

- `pickup_coal`
- `pickup_wood`
- `use_furnace`

What tasks can be expressed in LTL?



Assume agent can detect primitive high-level events:

- `pickup_coal`
- `pickup_wood`
- `use_furnace`
- `on_lava`

Tasks in LTL

Task Type	LTL Formula	English
Single Goal	`eventually pickup_coal`	“Get coal”
Ordered Goals	`eventually (pickup_coal and (eventually use_furnace))`	“Get coal, then use the furnace”
Unordered Goals	`(eventually pickup_coal) and (eventually pickup_wood)`	“Get coal and get wood, in any order”
Disjunctive Goals	`(eventually pickup_coal) or (eventually pickup_wood)`	“Get coal or get wood”
Safety	`(eventually pickup_wood) and (always (not on_lava))`	“Get wood while avoiding lava”

Tasks in LTL

Task Type	LTL Formula	English
Single Goal	`eventually pickup_coal`	"Get coal"
Ordered Goals	`eventually (pickup_coal and (eventually use_furnace))`	"Get coal, then use the furnace"
Unordered Goals	`(eventually pickup_coal) and (eventually pickup_wood)`	"Get coal and get wood, in any order"
Disjunctive Goals	`(eventually pickup_coal) or (eventually pickup_wood)`	"Get coal or get wood"
Safety	`(eventually pickup_wood) and (always (not on_lava))`	"Get wood while avoiding lava"

Tasks in LTL

Task Type	LTL Formula	English
Single Goal	**eventually** pickup_coal	"Get coal"
Ordered Goals	**eventually** (pickup_coal **and** (**eventually** use_furnace))	"Get coal, then use the furnace"
Unordered Goals	(**eventually** pickup_coal) **and** (**eventually** pickup_wood)	"Get coal and get wood, in any order"
Disjunctive Goals	(**eventually** pickup_coal) **or** (**eventually** pickup_wood)	"Get coal or get wood"
Safety	(**eventually** pickup_wood) **and** (**always** (**not** on_lava))	"Get wood while avoiding lava"

LTL Instruction → Reward

$$R = \begin{cases} 1 & \text{if instruction is } \textbf{satisfied} \\ -1 & \text{if instruction is } \textbf{falsified} \\ 0 & \text{otherwise} \end{cases}$$

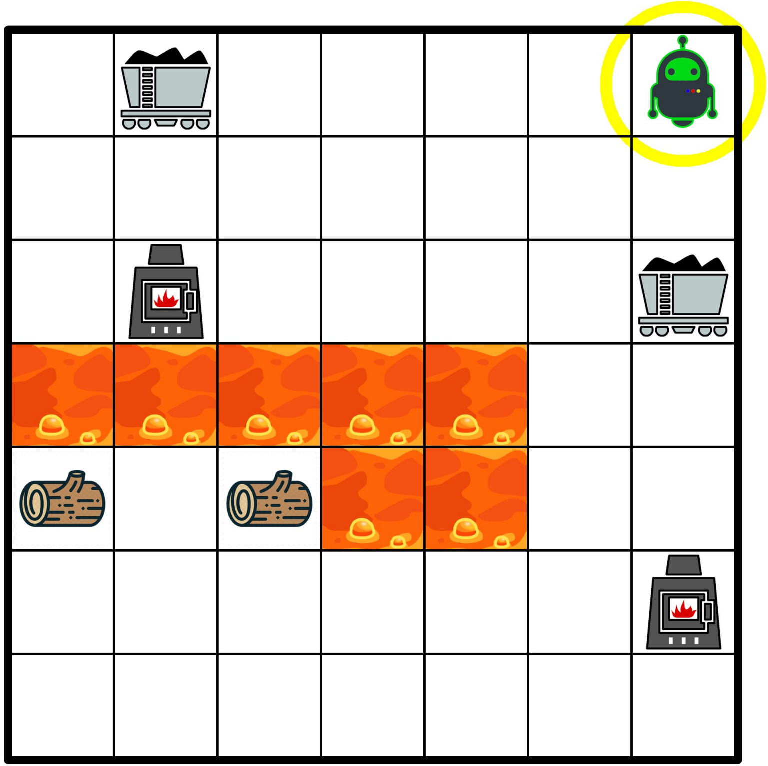
Task Decomposition

Task Decomposition

LTL Progression *[Bacchus & Kabanza, 2000]*

- Formally defined for all LTL formulas
- **Simplify instructions** once parts of the task are solved

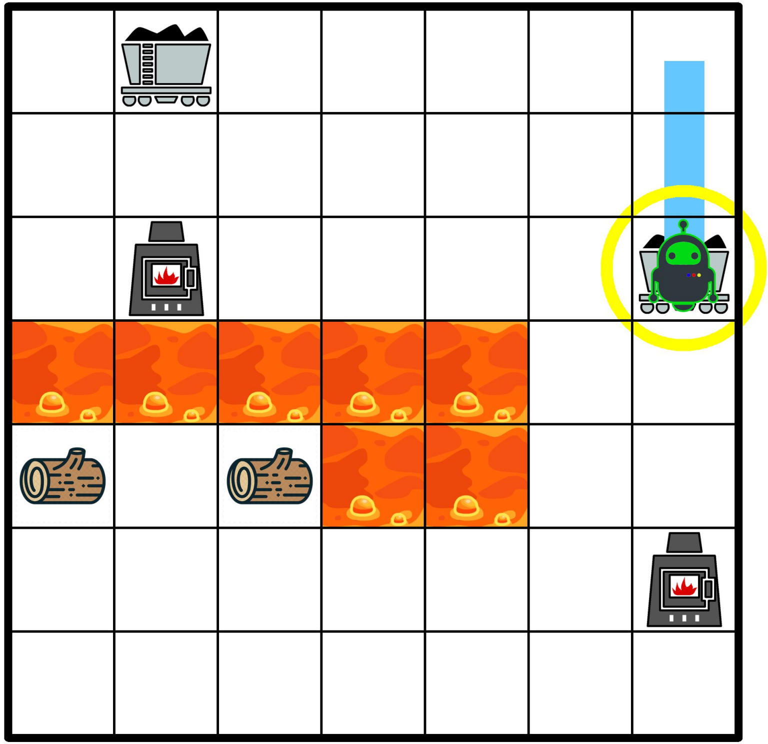
LTL Progression — Example



“Get coal or wood, then use the furnace.”

```
eventually ((pickup_coal or pickup wood)
and (eventually use_furnace))
```

LTL Progression — Example



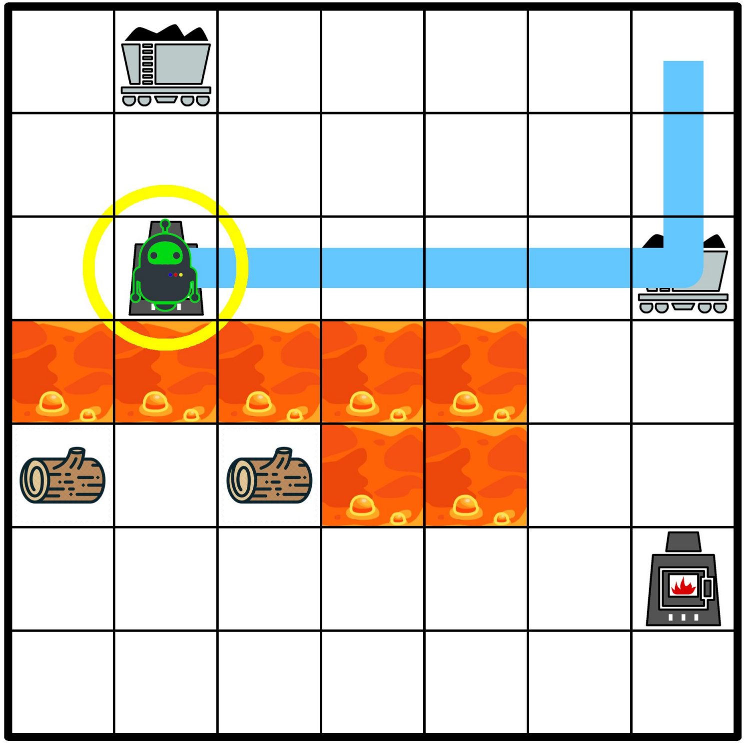
“Get coal or wood, then use the furnace.”

```
eventually ((pickup_coal  or pickup wood)
and (eventually use_furnace))
```

“Use the furnace.”

```
eventually use_furnace
```

LTL Progression — Example



"Get coal or wood, then use the furnace."

```
eventually ((pickup_coal or pickup wood)
and (eventually use_furnace))
```

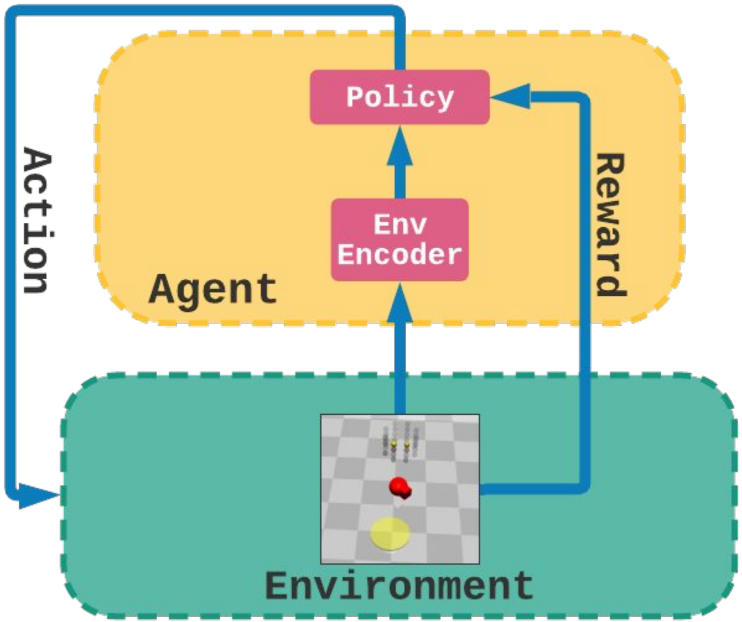
"Use the furnace."

```
eventually use_furnace
```

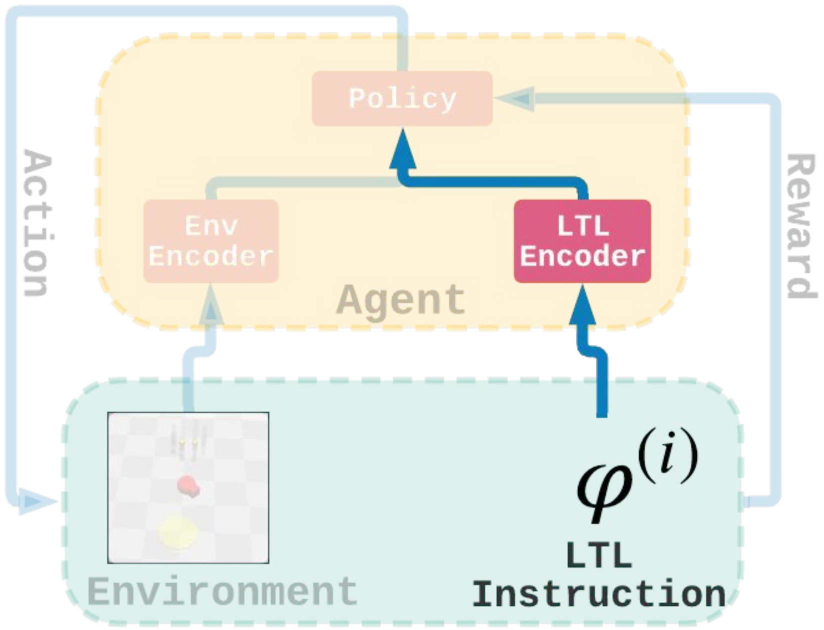
Complete

```
True
```

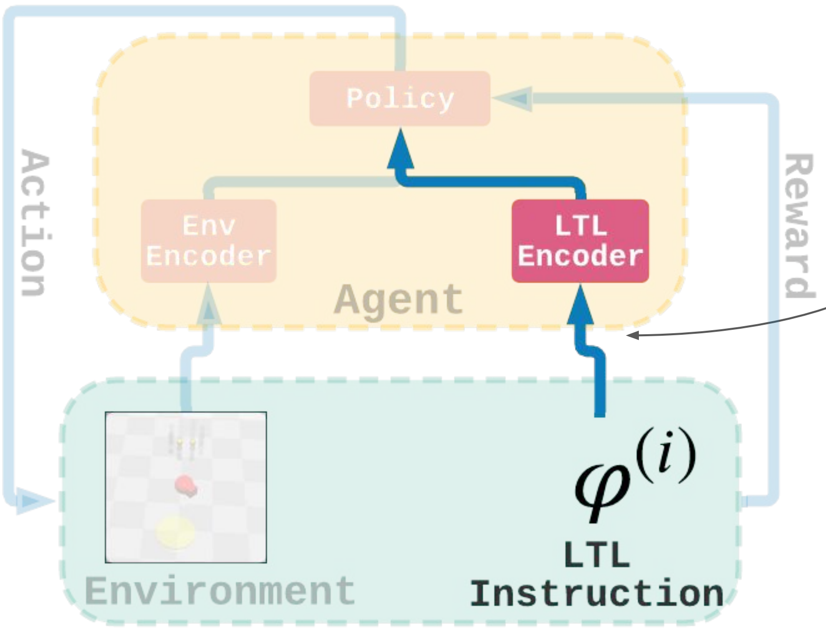
Our Approach



Our Approach

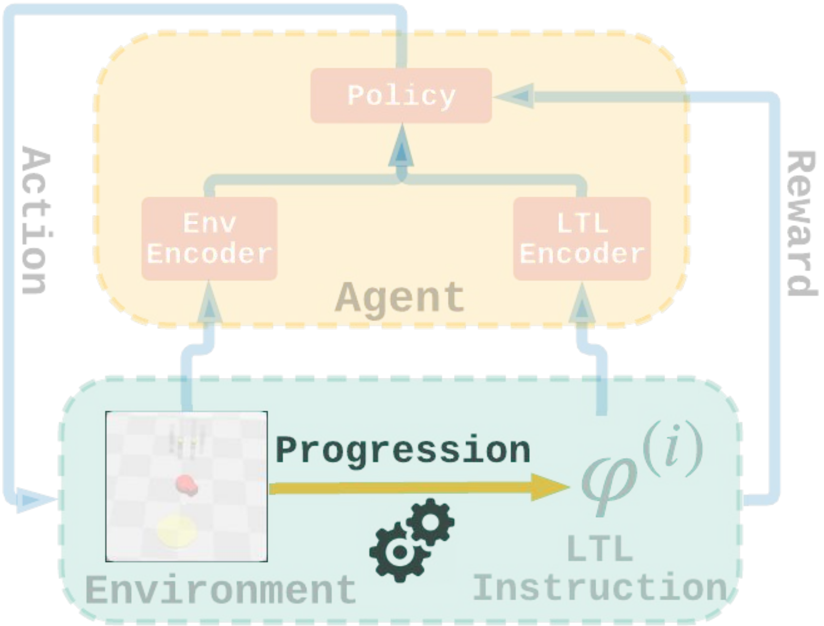


Our Approach



Can encode any LTL
instruction!

Our Approach



Theorems

Theorems

- ✓ Retains optimal convergence guarantees

Theorems

- ✓ Retains optimal convergence guarantees
- ✓ Markov assumptions hold

Results

Avoidance Tasks

(> 970 million possible tasks)

$$
\begin{aligned}
\mathrm{formula} &:= \mathrm{sequence} \wedge \mathrm{formula} \mid \mathrm{sequence} \\
\mathrm{sequence} &:= \neg\mathsf{prop}(\mathsf{prop} \wedge \mathrm{sequence}) \mid \neg\mathsf{prop}\,\mathsf{prop}
\end{aligned}
$$

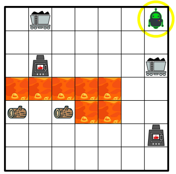
Partially-Ordered Tasks

(> 10^{39} possible tasks)

$$
\begin{aligned}
\mathrm{formula} &:= \mathrm{sequence} \wedge \mathrm{formula} \mid \mathrm{sequence} \\
\mathrm{sequence} &:= \Diamond(\mathrm{term} \wedge \mathrm{sequence}) \mid \Diamond\mathrm{term} \\
\mathrm{term} &:= \mathsf{prop} \mid \mathsf{prop} \vee \mathsf{prop}
\end{aligned}
$$

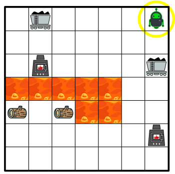
Results

Gridworld (Discrete)

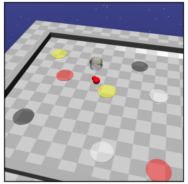


Results

Gridworld (Discrete)

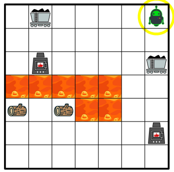


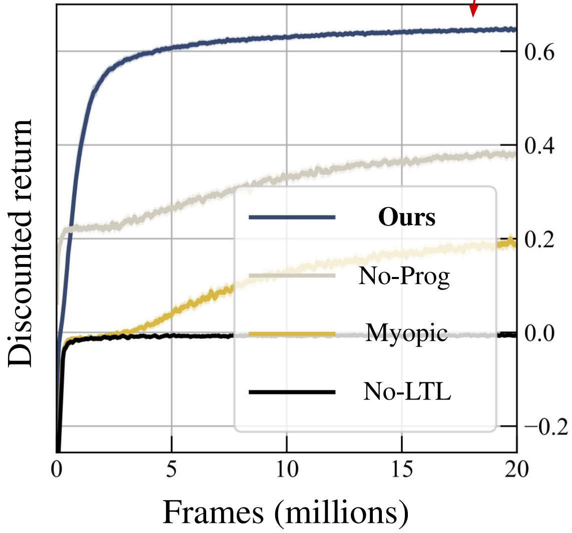
MuJoCo (Continuous)



Results

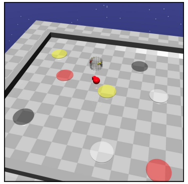
Gridworld (Discrete)

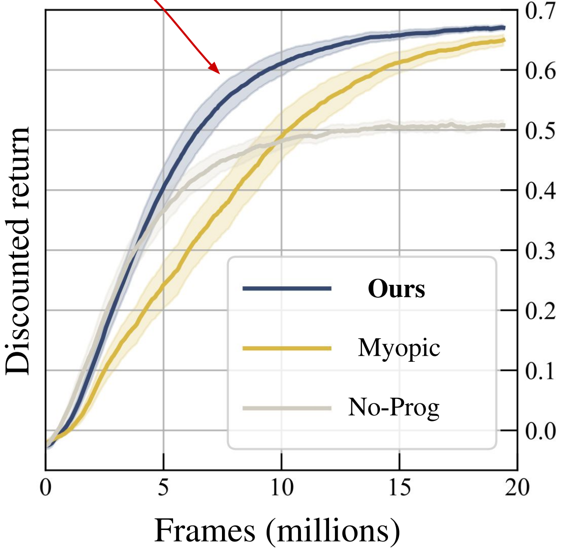




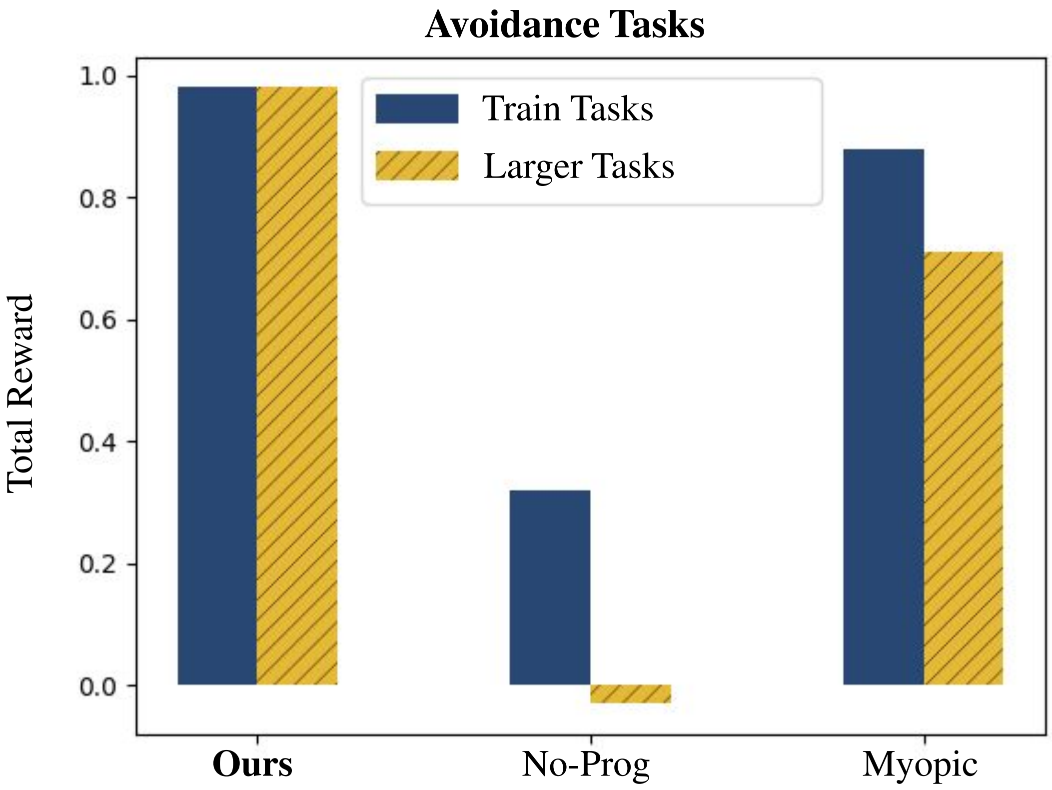
Ours

MuJoCo (Continuous)





Zero-Shot Generalization



Other topics …

Other topics …

- A environment-agnostic **pre-training** scheme to learn LTL semantics

Other topics …

- A environment-agnostic **pre-training** scheme to learn LTL semantics
- What neural architecture best encodes LTL instructions?

Other topics …

- A environment-agnostic **pre-training** scheme to learn LTL semantics
- What neural architecture best encodes LTL instructions?
- Procedural generation of meaningful LTL instructions

Other topics …

- A environment-agnostic **pre-training** scheme to learn LTL semantics
- What neural architecture best encodes LTL instructions?
- Procedural generation of meaningful LTL instructions

Come to our poster!

Code is available at:

https://github.com/LTL2Action/LTL2Action