LTL2Action: Generalizing LTL Instructions for Multi-Task RL

Pashootan Vaezipoor*†‡ Andrew C. Li*†‡ Rodrigo Toro Icarte†‡ Sheila McIlraith†‡

†Department of Computer Science, University of Toronto ‡Vector Institute

†{pashootan, andrewli, rntoro, sheila}@cs.toronto.edu

Abstract

We address the problem of teaching a deep reinforcement learning (RL) agent to **follow instructions in multi-task environments**. Instructions are expressed in a **well-known formal language** – ***linear temporal logic* (LTL)** – and can specify a diversity of complex, temporally extended behaviours, including conditionals and alternative realizations. Our proposed learning approach exploits the **compositional syntax and the semantics** of LTL, enabling our RL agent to learn task-conditioned policies that generalize to new instructions, not observed during training. To reduce the overhead of learning LTL semantics, we introduce an environment-agnostic LTL pretraining scheme which improves sample-efficiency in downstream environments. Experiments on discrete and continuous domains target combinatorial task sets of **up to $\sim 10^{39}$ unique tasks** and demonstrate the strength of our approach in learning to **solve (unseen) tasks, given LTL instructions**. a

Background

Multi-Task Reinforcement Learning

Goal: Train a single task-conditioned policy to generalize to a wide array of tasks. Tasks are specified in the formal language *linear temporal logic (LTL)*.

Linear Temporal Logic (LTL)

Next Eventually

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc\varphi \mid \varphi \,\mathsf{U}\, \psi \mid \Diamond\varphi \mid \Box\varphi$$

Until Always

LTL is an expressive language with desirable properties for RL.

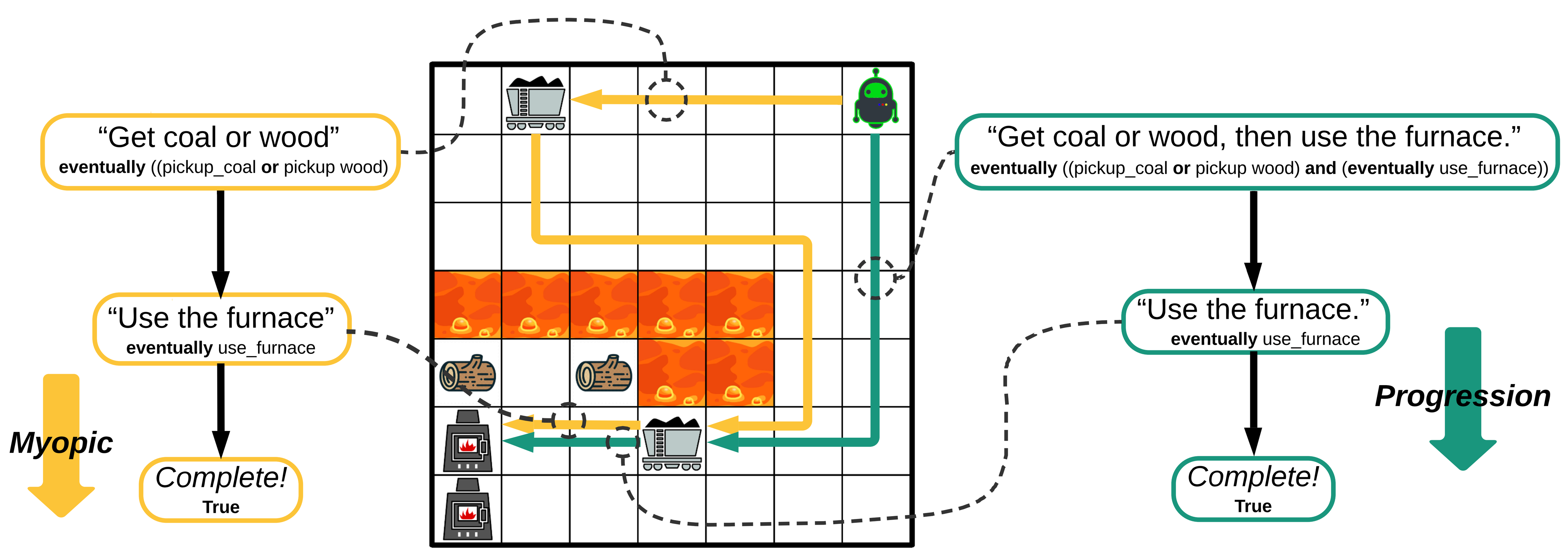
- **Temporal patterns** can be specified with modalities like *eventually, until, always* together with event predicates (i.e., propositions p).
- **Compositional syntax** allows us to procedurally sample diverse, meaningful tasks for training (over 10^{39} tasks, in our experiments).
- **Unambiguous semantics** allow us to automatically determine task completion, unlike natural language. We don't rely on manually labelled data.

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

Task	LTL	English
Single Goal	$\Diamond$ `get_coal`	"Get coal"
Ordered Goals	$\Diamond$ (`get_coal` $\wedge$ $\Diamond$ `use_furnace`)	"Get coal then use the furnace"
Unordered Goals	$\Diamond$ `get_coal` $\wedge$ $\Diamond$ `get_wood`	"Get coal and wood, in any order"
Disjunctive Goals	$\Diamond$ `get_coal` $\vee$ $\Diamond$ `get_wood`	"Get coal or get wood"
Safety	$\Diamond$ `get_wood` $\wedge$ $\Box\neg$ `on_lava`	"Get wood while avoiding lava"

Challenges

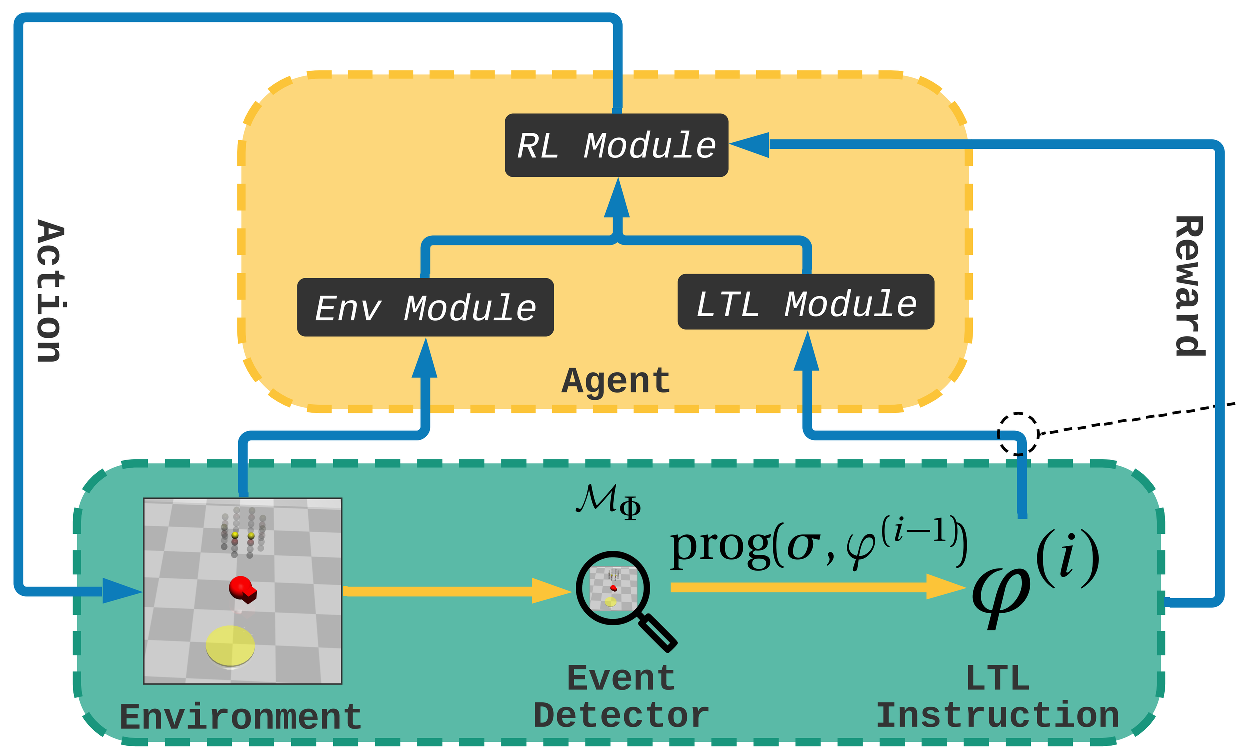
- **Non-Markovian Reward**: Some LTL tasks require memory with respect to the state (see table above).
- **Myopia**: Standard techniques for decomposing tasks into sequential subtasks are sub-optimal (see example below, left).
- **Generalization**: Most work on LTL+RL does not generalize to unseen tasks.

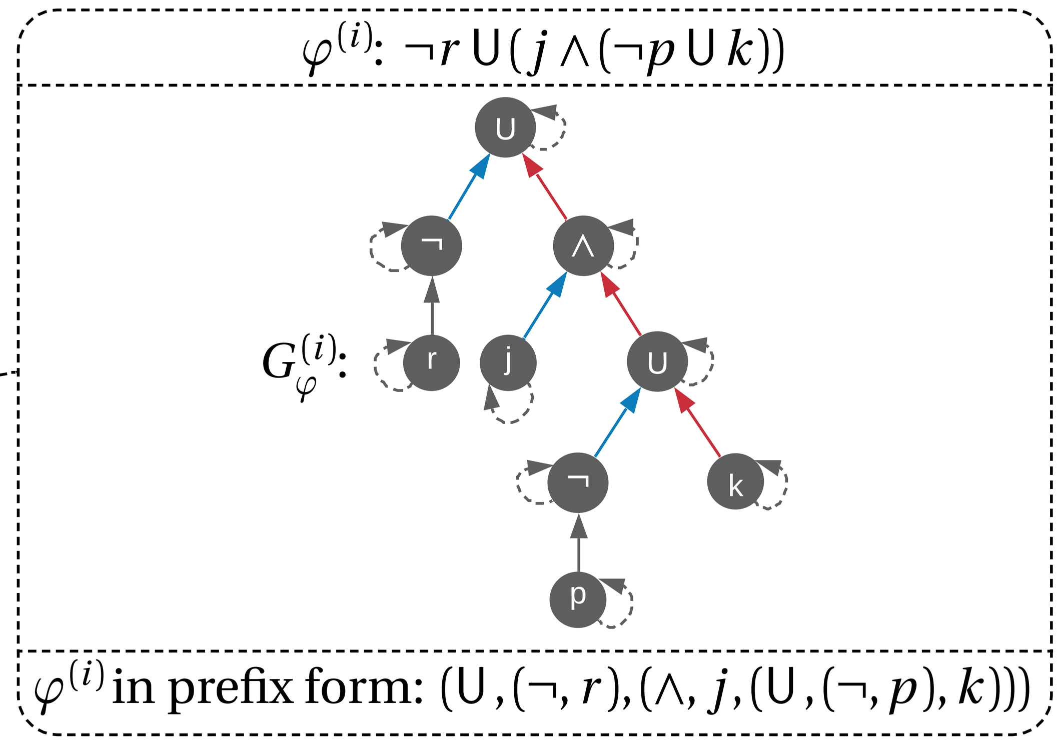


LTL2Action

Neural Encodings of LTL Formulas

We encode the LTL instructions with a neural network to enable generalization to unseen tasks. We considered encoding the syntax as a sequence of tokens (**GRU**, **LSTM**) or the abstract syntax tree (**GNN**).





LTL Progression

LTL Progression (Bacchus & Kabanza, 2000) is a formal method for simplifying instructions over time as parts of the task are solved (example in bottom left Figure). We show the following guarantees:

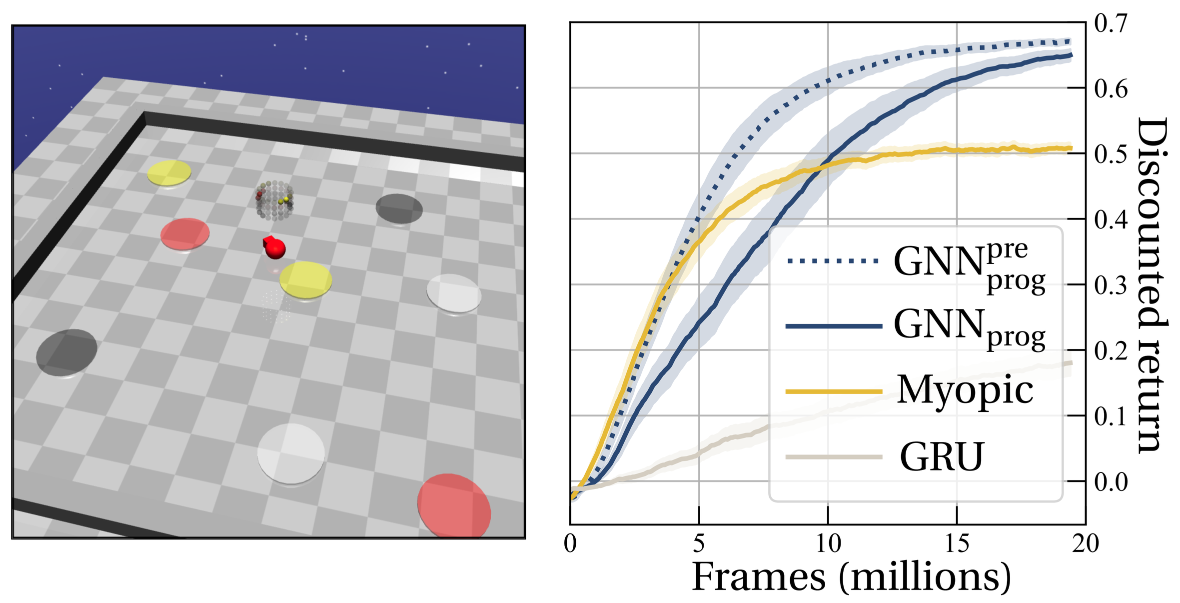
Theorem: For LTL tasks, there exists an optimal policy that is Markovian when the instructions are updated via LTL progression.

✓ Standard Markov RL can be applied

✓ Non-myopic

Pretraining

As LTL syntax and semantics are environment-agnostic, we propose to pretrain encodings of LTL *without interacting with any physical environment.*



The results on ZoneEnv, a MuJoCo-based continuous control environment with coloured zones as LTL propositions. Tasks involve reaching zones of certain colours in the correct order (while avoiding zones of the incorrect colour).

Pretraining Task: Given an LTL formula φ, satisfy φ as quickly as possible, choosing one proposition to be true per step.

Experiments

We conducted experiments on diverse, procedurally-generated LTL tasks, and across Gridworld and MuJoCo environments.

Key Results

- **Performance**: LTL2Action outperforms other approaches which do not use LTL progression or are myopic.
- **Architecture**: Compositional architectures (GNN) encode LTL formulas better than sequence models (LSTM, GRU).
- **Pretraining**: Pretraining LTL encodings results in more rapid convergence in novel downstream environments.
- **Upward Generalization**: Our approach robustly generalizes to instructions up to 3× larger than those in training.

Upward Generalization