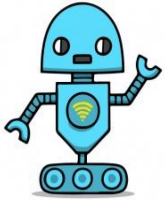


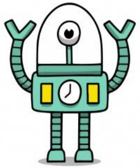
PhD Qualifying Exam Defense

Logic-based Reward Shaping for Multi-Agent Reinforcement Learning

Ingy ElSayed-Aly

Committee: Prof. Lu Feng, Prof. Haiying Shen, Prof. Haifeng Xu, Prof. Hongning Wang

06/10/23



Motivation

- **Problem:** Designing a reward function is still a manual and tricky process in Reinforcement Learning.
  - Exacerbated in multi-agent settings.
- **Potential Solution: Logic-based reward shaping** allows us to automatically construct a reward function based on the task.

What is logic-based reward shaping?

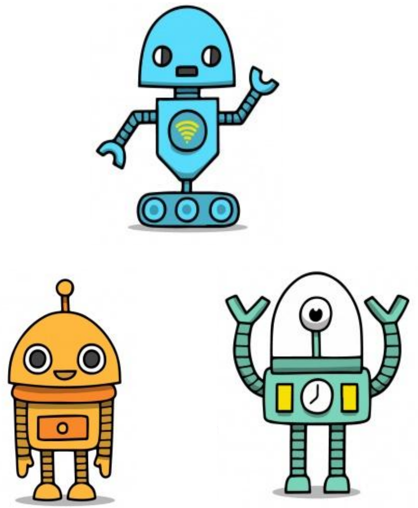
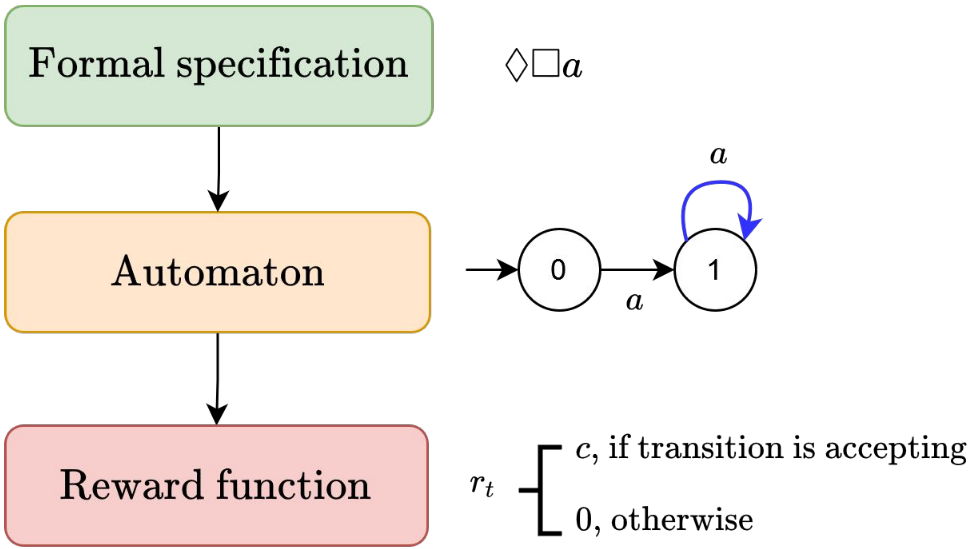


Photo Credit : Freepik

Logic Based Reward Shaping

- **Formal specification** to represent the desired behavior (ex. LTL).
- **Automaton** is automatically generated based on the specification.
- The **reward function** is defined based on the automaton states or transitions.



Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE, 2020.

Related Work

- Camacho et al. introduces Reward Machines (RM) on the basis of the **LTL** co-safe fragment.
- A reward machine is a **Mealy machine** that based on MDP states, actions and labels outputs a reward function.
- The authors compute a **potential function** over the reward machine to guide the reward shaping.
- Tailored Q-Learning algorithms are used.

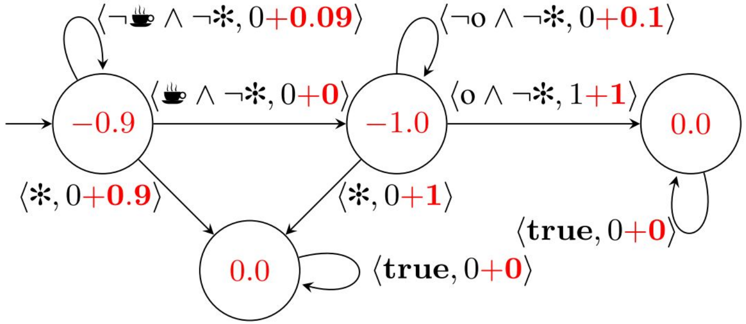
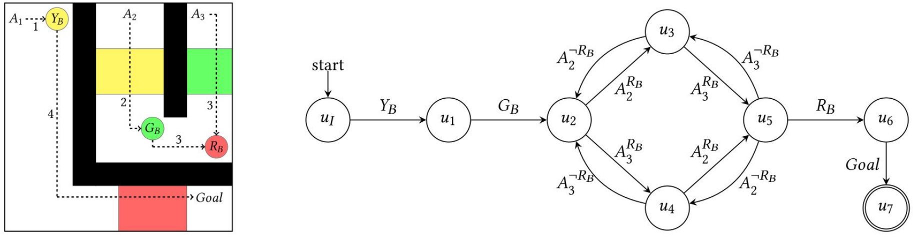


Figure 2: Reward shaping example with $\gamma = 0.9$.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073, 2019

Related Work

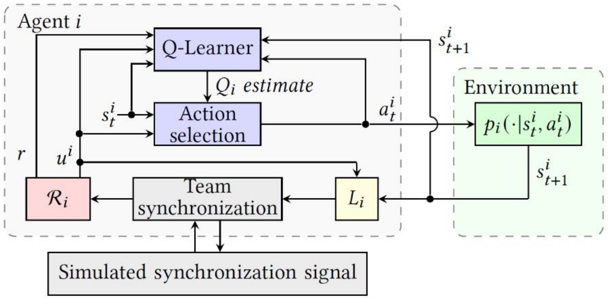
- Neary et al. decomposes MARL problems into collections of Reward Machines for single-agent RL tasks.
- The main innovation of this paper is using a cooperative **goal decomposition** to solve some multi-agent tasks.
- Once the multi-agent goals are reduced to **individual goals** a decentralized learning RL algorithm is used to learn the tasks.



(a) Cooperative buttons domain.

(b) Reward machine encoding the cooperative buttons task.

Figure 1: The multi-agent buttons task. In Figure (a), the colored circles denote the locations of the buttons, the thick black areas are walls the agents cannot cross, and the numbered dotted lines show the order of high-level steps necessary to complete the task. The set of events of the RM in (b) is $\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$.

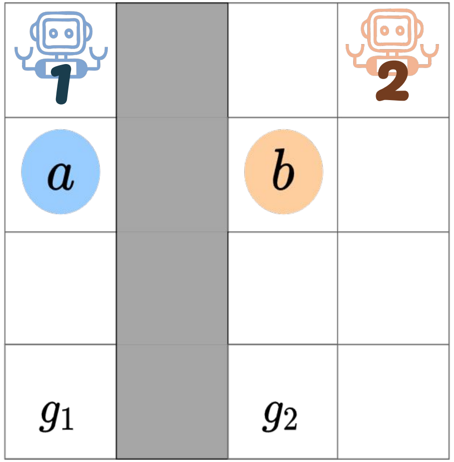


Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. Reward machines for cooperative multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 2021.

Problem Setting

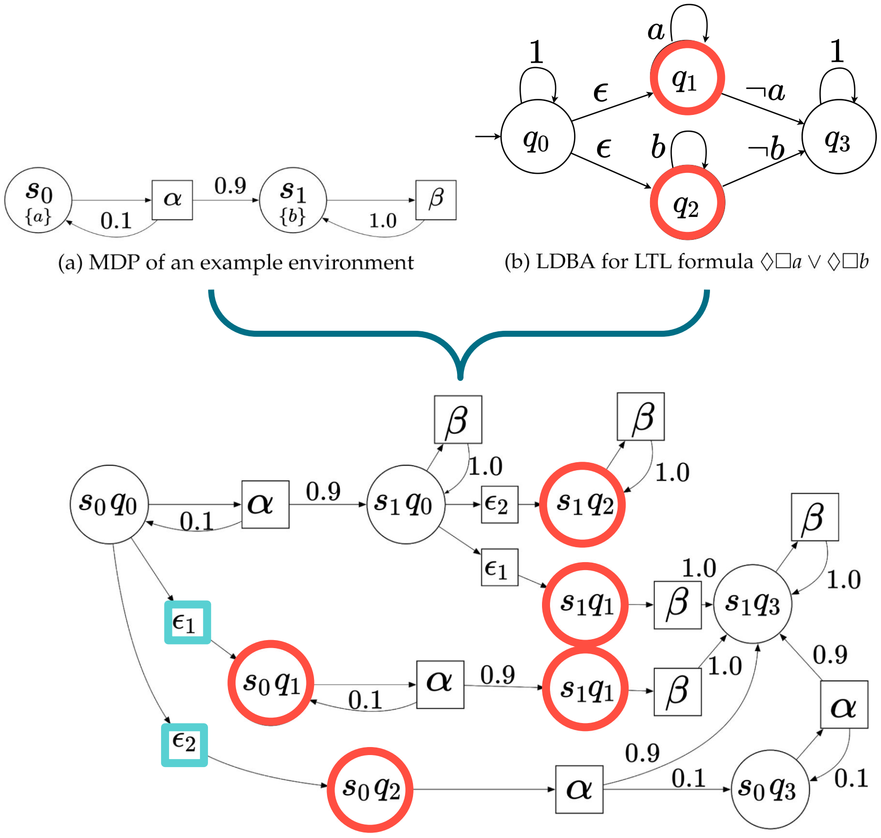
- Previous work requires explicit **task decomposition** and **tailored RL** algorithms.
- Multi-agent settings often suffer from an exponential increase in the number states.

- **Objective:** Create a more flexible algorithm agnostic **reward shaping framework** for Multi-Agent RL .



Reward Function

- **Limit-Deterministic Büchi Automaton** (LDBA) automatically constructed using the **LTL** specification.
- Product of the LDBA and the environment MDP.
- Increased state space and action space.
- Agents get a reward for being in an **accepting state** or **transition**.



(a) MDP of an example environment

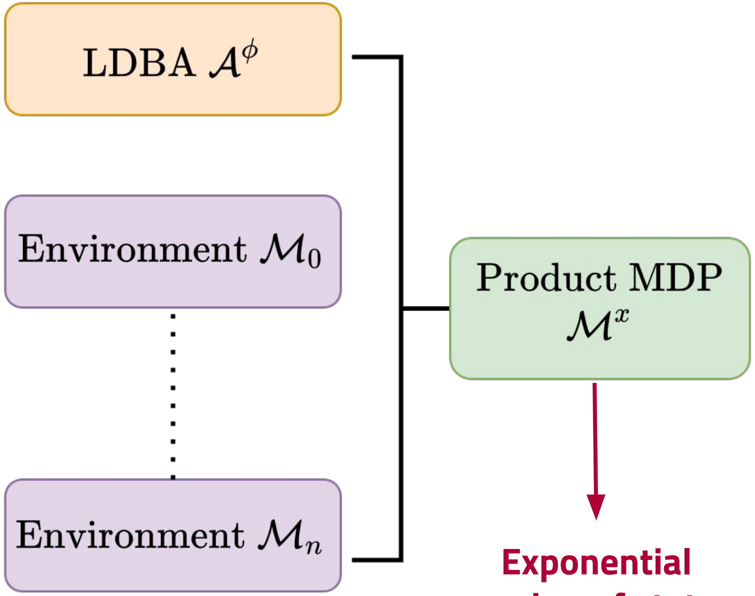
(b) LDBA for LTL formula $\lozenge\square a \vee \lozenge\square b$

(c) Resulting product MDP

Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE, 2020.

Reward Function: Multi-Agent

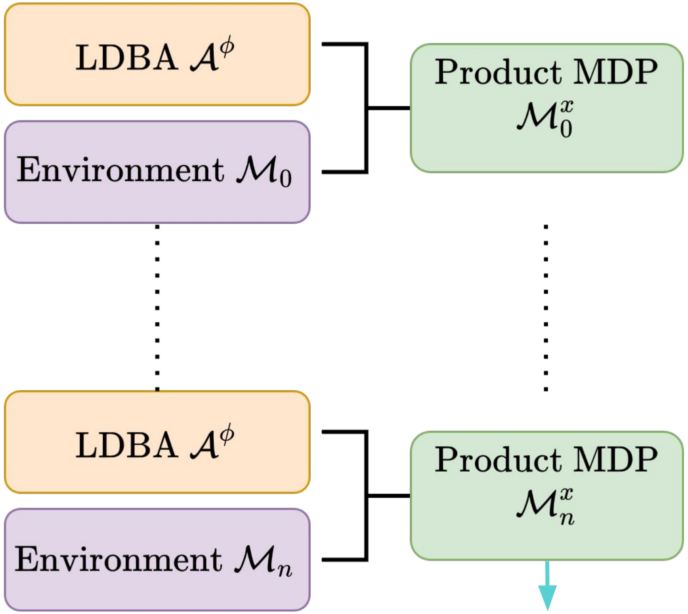
1. Straightforward Approach



Exponential number of states

$$\mathcal{M}^x = (\mathcal{A}^\phi \times \mathcal{M}_0 \times \ldots \times \mathcal{M}_n)$$

2 Our approach

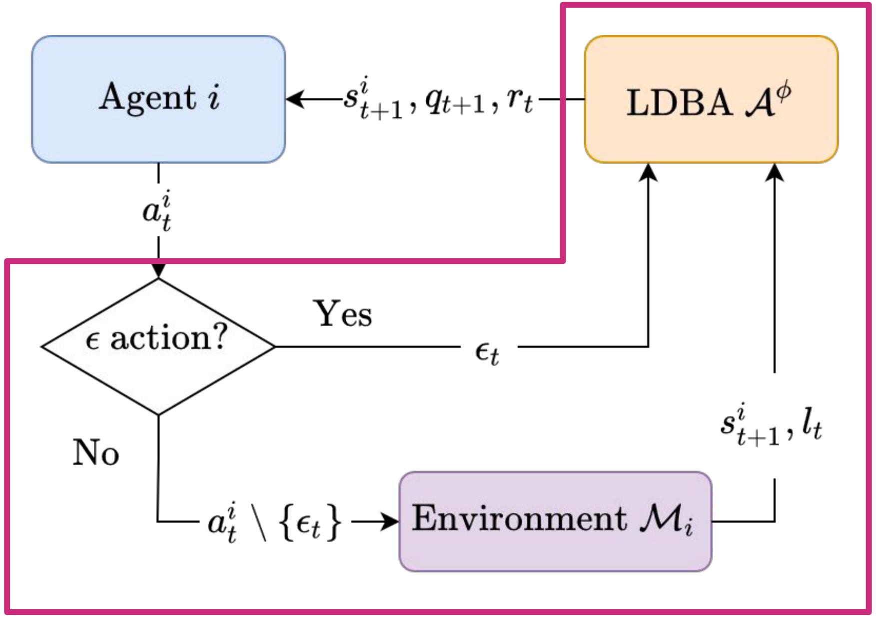


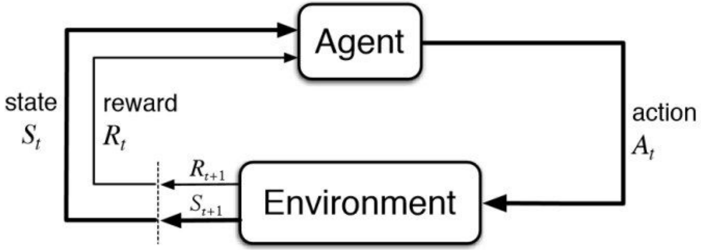
We can simulate each Product MDP

$$(\mathcal{A}^\phi \times \mathcal{M}_0) \ldots (\mathcal{A}^\phi \times \mathcal{M}_n)$$

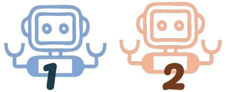
Semi-Centralized Multi-Agent Reward Shaping

- Simulates the full product MDP through simulation of multiple smaller product MDPs.
- Implementation only needs to take into account one agent's point of view.
- Shared LDBA states, epsilon actions and labeling function.



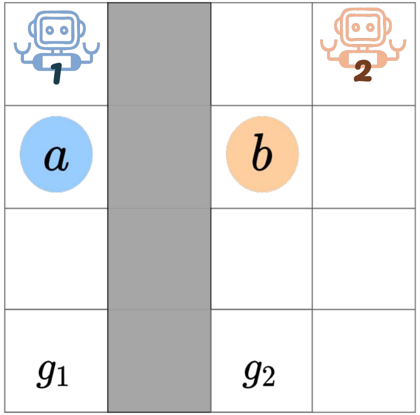


Motivating Example: Buttons

- Agents : 

- Buttons: 

- Goals: **g1, g2**



$$\phi_3 = ((\Diamond a \wedge \neg g_1) \cup (a \wedge b)) \wedge ((\Diamond b \wedge \neg g_2) \cup (a \wedge b)) \wedge (\Diamond \Box g_1) \wedge (\Diamond \Box g_2)$$

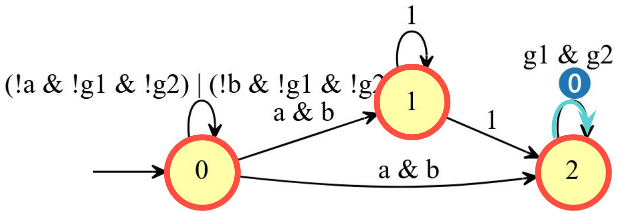
- Try going to **a** but don't go to **goal1** until **a&b.**
- Try going to **b** but don't go to **goal2** until **a&b.**
- Eventually go and stay at **goal1** and eventually go and stay at **goal2.**

Motivating Example: Buttons

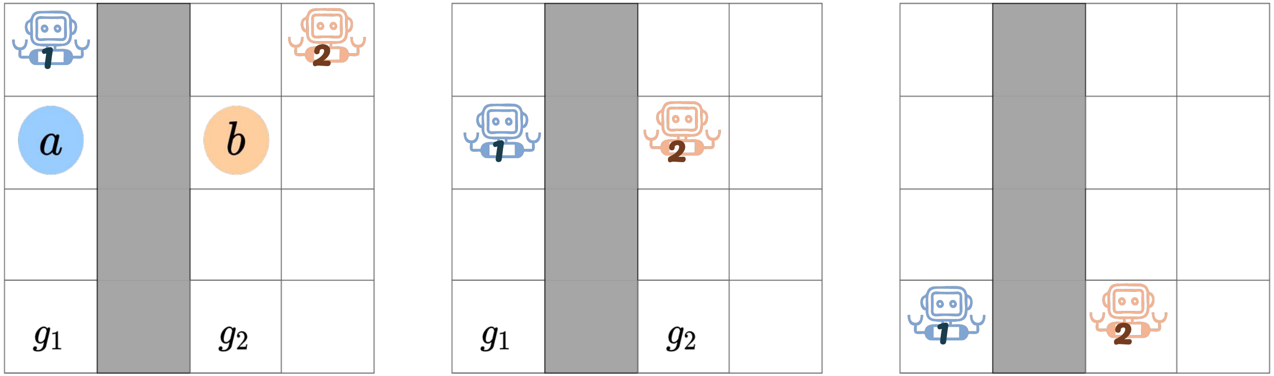
Specification:

$$\boxed{\phi_3 = ((\Diamond a \wedge \neg g_1) \cup (a \wedge b)) \wedge ((\Diamond b \wedge \neg g_2) \cup (a \wedge b)) \wedge (\Diamond \Box g_1) \wedge (\Diamond \Box g_2)}$$

Automaton:



Environment:



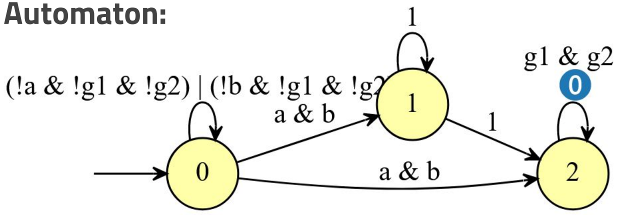
- Try going to **a** but don't go to **goal1** until **a&b.**
- Try going to **b** but don't go to **goal2** until **a&b.**
- Eventually go and stay at **goal1.**
- Eventually go and stay at **goal2.**

Motivating Example: Buttons

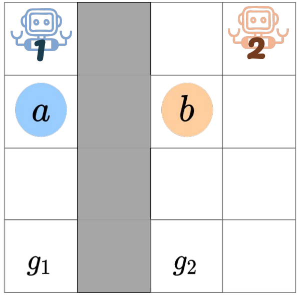
Specification:

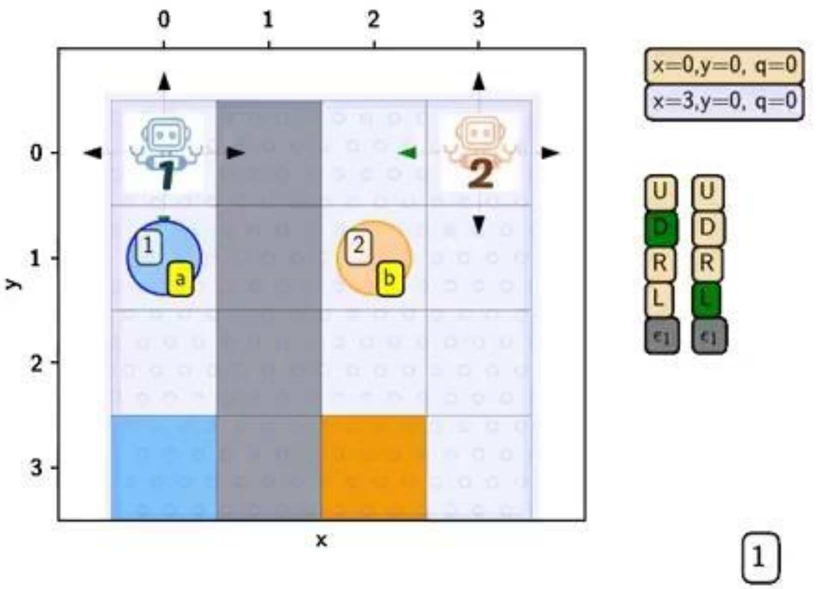
$$\boxed{\phi_3 = ((\Diamond a \wedge \neg g_1) \cup (a \wedge b)) \wedge ((\Diamond b \wedge \neg g_2) \cup (a \wedge b)) \wedge (\Diamond \Box g_1) \wedge (\Diamond \Box g_2)}$$

Automaton:



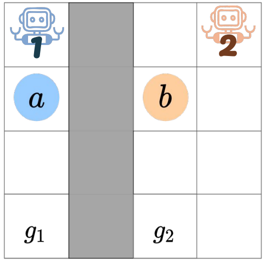
Environment:





Benchmark 1: Buttons

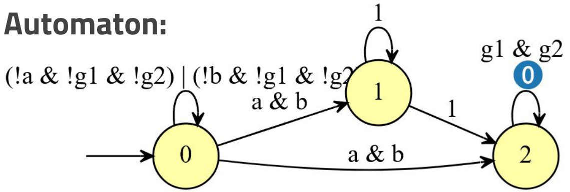
Environment:

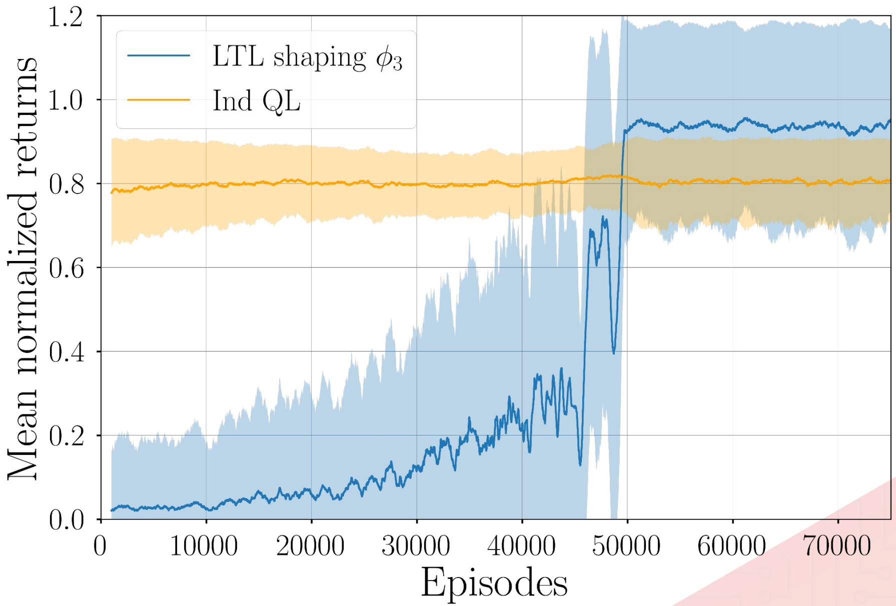


Specification:

$$\phi_3 = ((\Diamond a \wedge \neg g_1) \cup (a \wedge b)) \wedge ((\Diamond b \wedge \neg g_2) \cup (a \wedge b)) \wedge (\Diamond\Box g_1) \wedge (\Diamond\Box g_2)$$

Automaton:

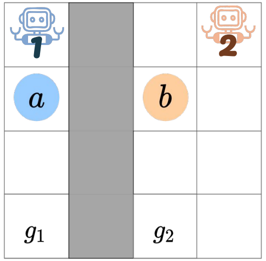




- The accumulated reward per episode is averaged between agents before being smoothed using a rolling window.
- Both use **Independent QL** as an underlying MARL algorithm.
- The **mean** for our method is better but the **standard deviation** is worse.

Benchmark 1: Buttons

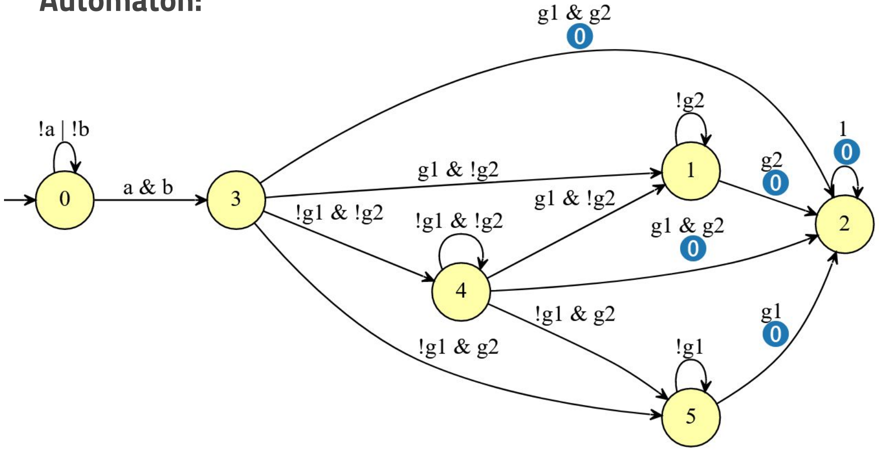
Environment:

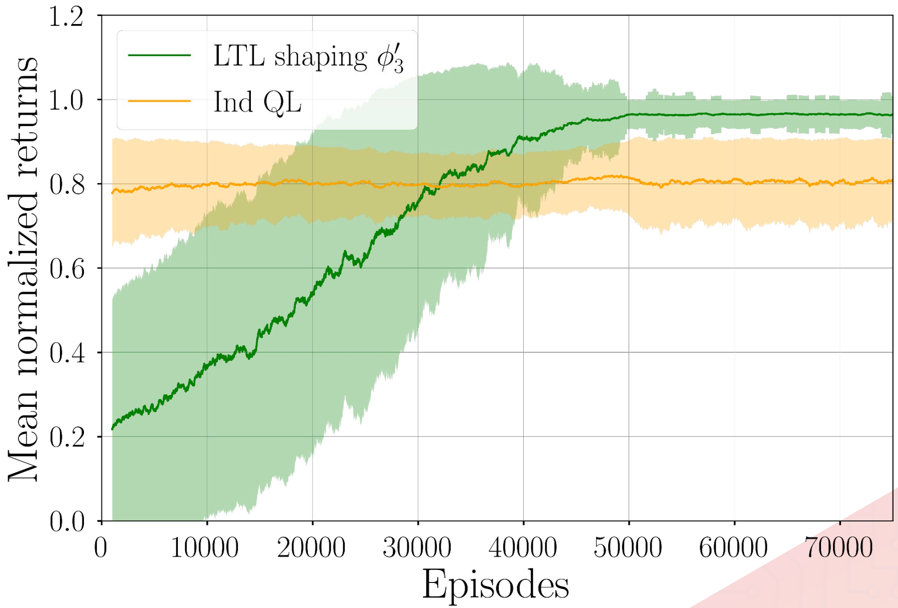


Specification:

$$\phi_3' = \Diamond((a \wedge b) \wedge \bigcirc\Diamond((g_1 \vee \bigcirc\Diamond g_1) \wedge (g_2 \vee \bigcirc\Diamond g_2)))$$

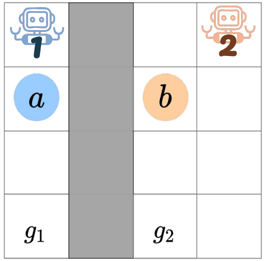
Automaton:





Benchmark 1: Buttons

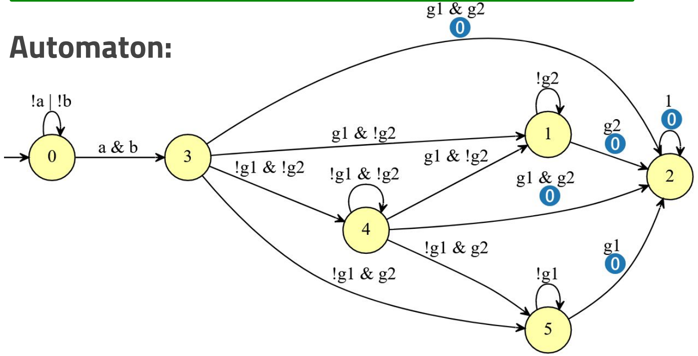
Environment:

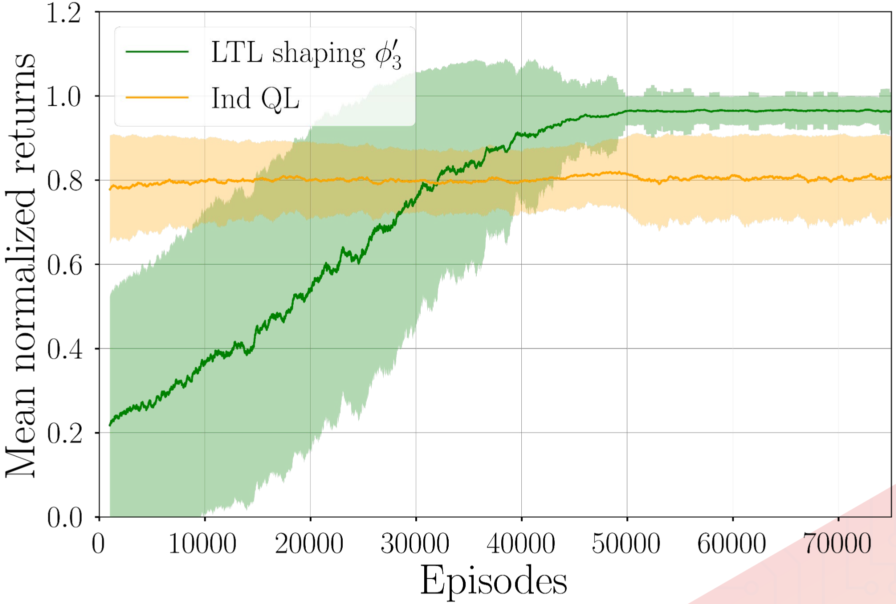


Specification:

$$\boxed{\phi_3' = \Diamond((a \wedge b) \wedge \bigcirc\Diamond((g_1 \vee \bigcirc\Diamond g_1) \wedge (g_2 \vee \bigcirc\Diamond g_2)))}$$

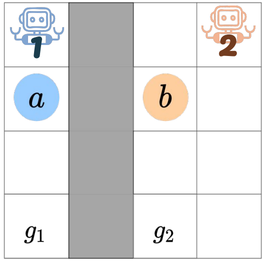
Automaton:



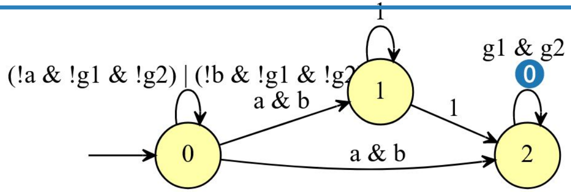


- Same environment and method, different formatting of the **specification**.
- Both use **Independent QL** as an underlying MARL algorithm.
- Both the **mean** and the **standard deviation** for our method is better.

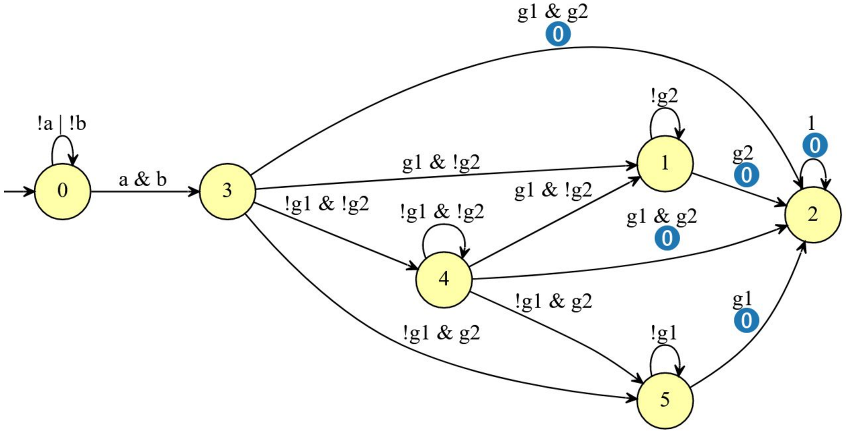
Benchmark 1: Buttons

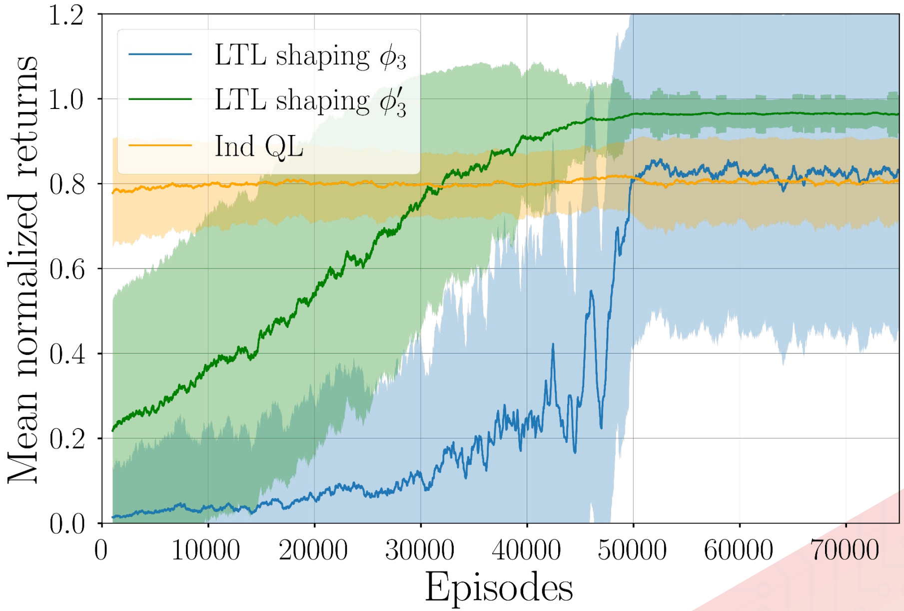


$$\phi_3 = ((\Diamond a \wedge \neg g_1) \cup (a \wedge b)) \wedge ((\Diamond b \wedge \neg g_2) \cup (a \wedge b)) \wedge (\Diamond\Box g_1) \wedge (\Diamond\Box g_2)$$



$$\phi_3' = \Diamond((a \wedge b) \wedge \bigcirc\Diamond((g_1 \vee \bigcirc\Diamond g_1) \wedge (g_2 \vee \bigcirc\Diamond g_2)))$$



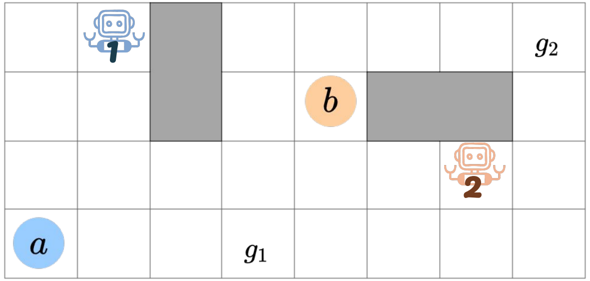


Benchmark 2: Flag Collection

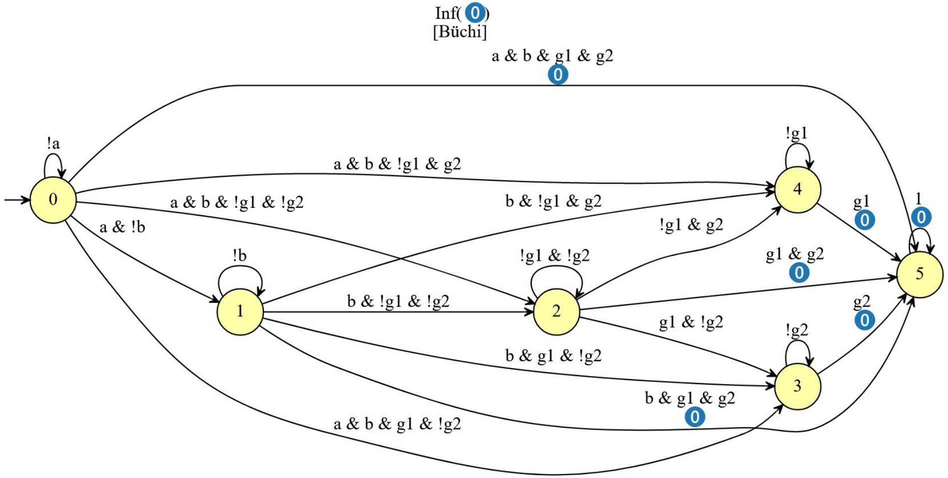
Specification:

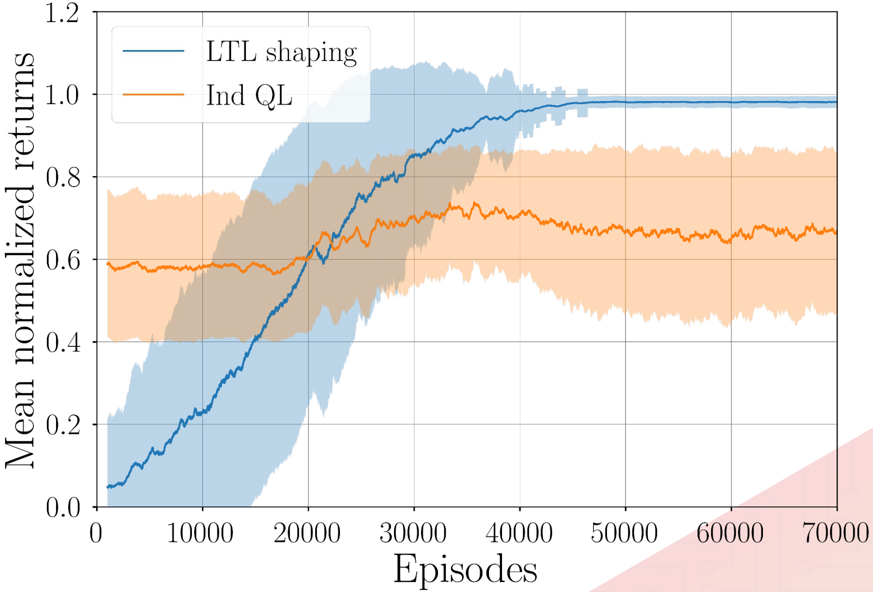
$$\phi = \Diamond(a \wedge \Diamond(b \wedge (\Diamond(g_1 \vee \bigcirc\Diamond g_1) \wedge \Diamond(g_2 \vee \bigcirc\Diamond g_2))))$$

Environment:



Automaton:



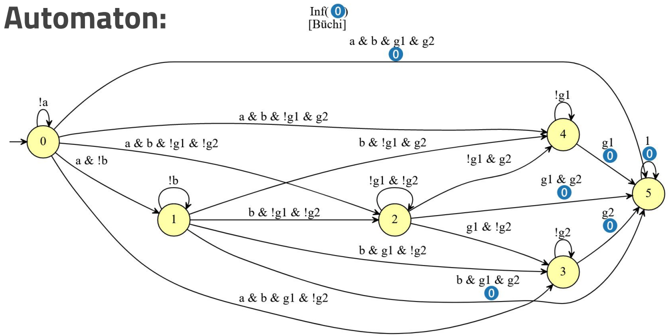


Benchmark 2: Flag Collection

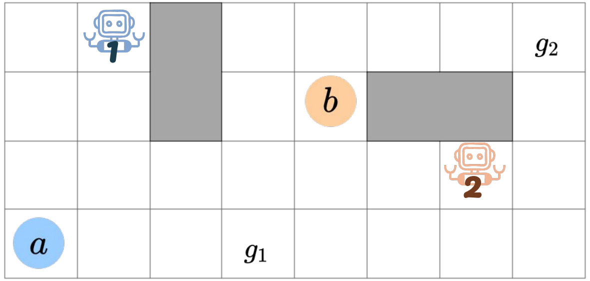
Specification:

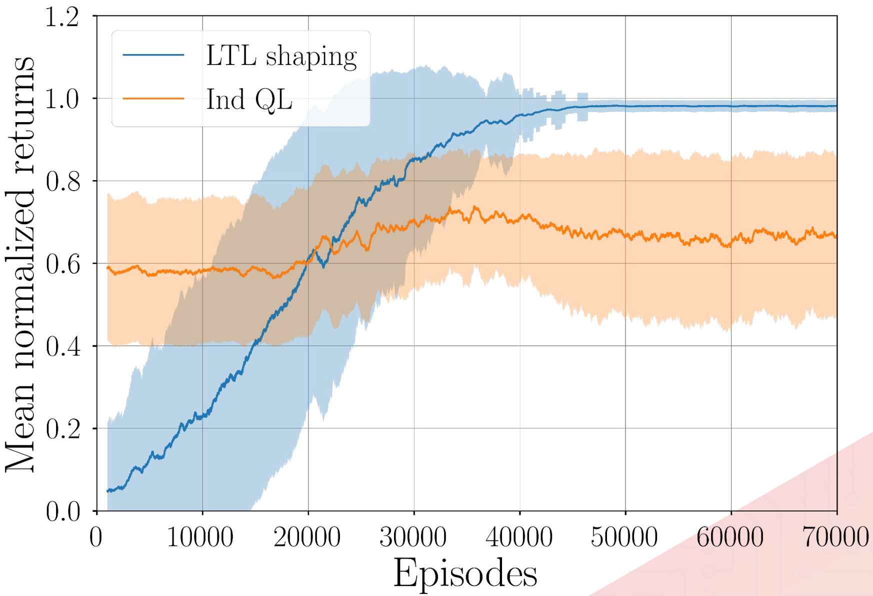
$$\phi = \Diamond(a \wedge \Diamond(b \wedge (\Diamond(g_1 \vee \bigcirc\Diamond g_1) \wedge \Diamond(g_2 \vee \bigcirc\Diamond g_2))))$$

Automaton:



Environment:

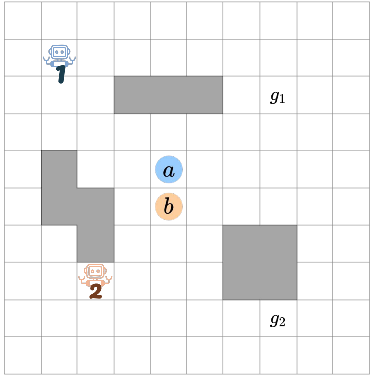




- Agents are **not assigned** a specific flag or goal.
- Both use **Independent QL** as an underlying MARL algorithm.
- Both the **mean** and the **standard deviation** for our method is much better.

Benchmark 3: Rendez-Vous

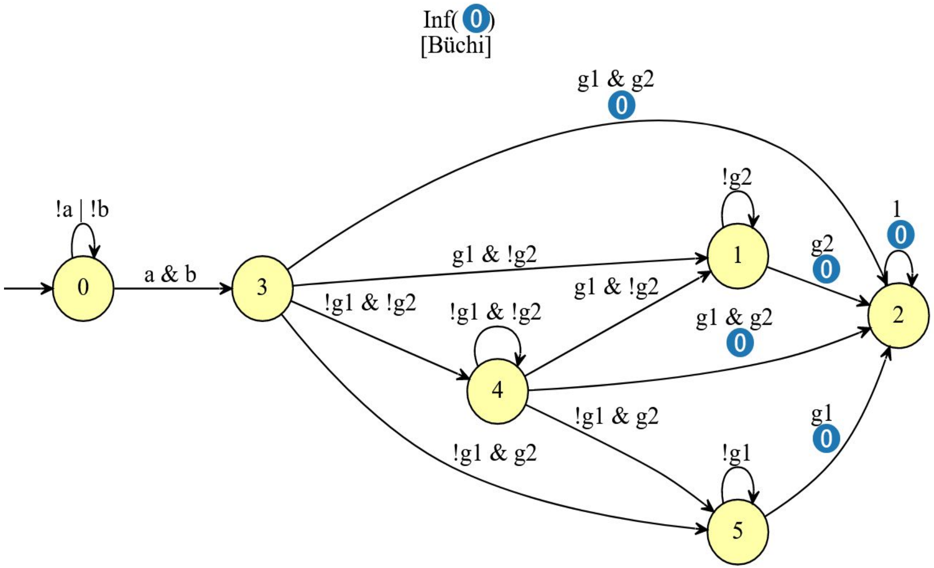
Environment:

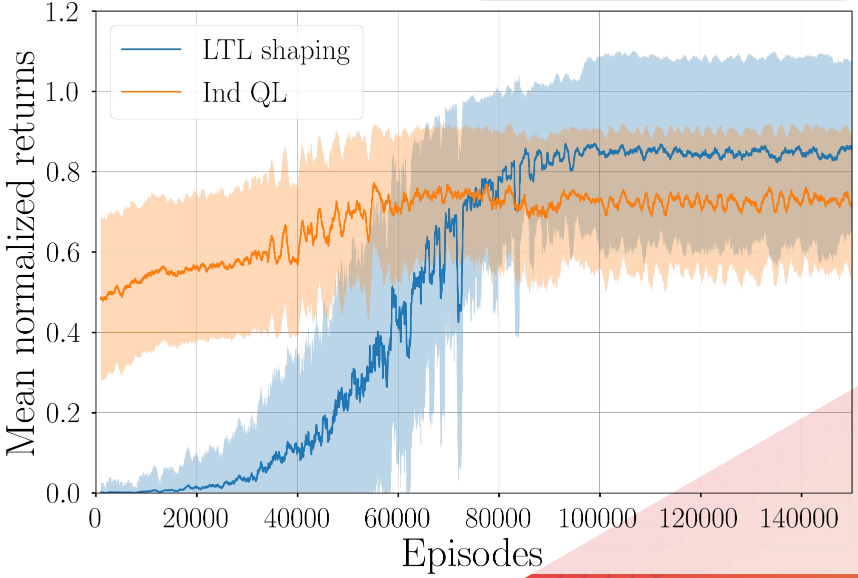


Specification:

$$\phi'_3 = \Diamond((a \wedge b) \wedge \bigcirc\Diamond((g_1 \vee \bigcirc\Diamond g_1) \wedge (g_2 \vee \bigcirc\Diamond g_2)))$$

Automaton:



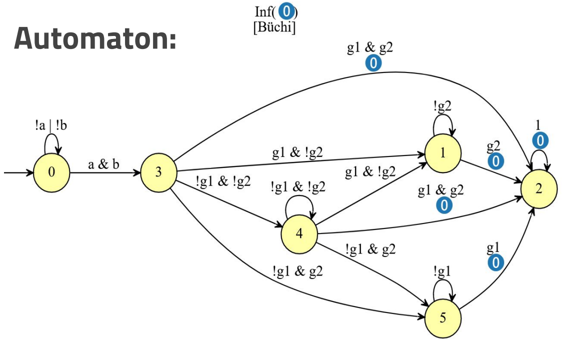


Benchmark 3: Rendez-Vous

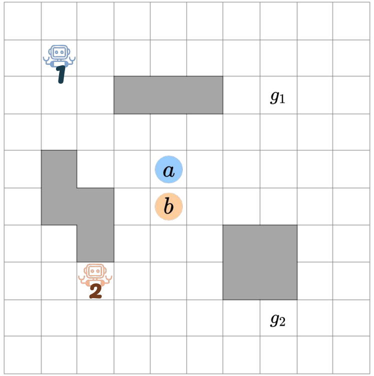
Specification:

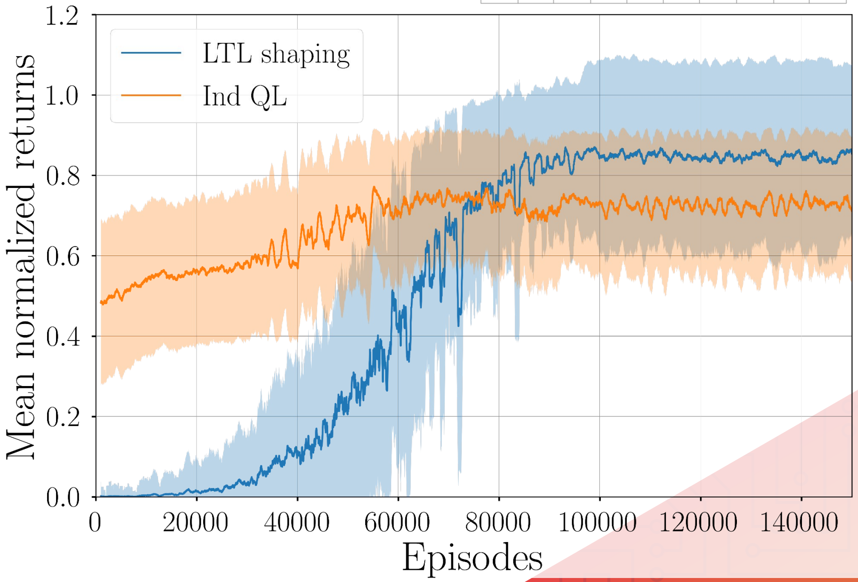
$$\phi_3' = \Diamond((a \wedge b) \wedge \bigcirc\Diamond((g_1 \vee \bigcirc\Diamond g_1) \wedge (g_2 \vee \bigcirc\Diamond g_2)))$$

Automaton:



Environment:

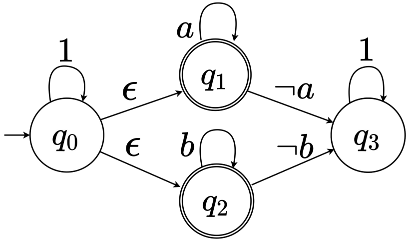


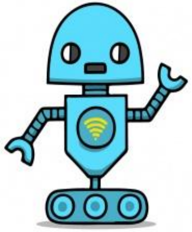


- Larger environment, both agent must **meet at a&b** before going to a goal location.
- Both use **Independent QL** as an underlying MARL algorithm.
- The **mean** for our method is better.

Conclusion

- Our approach has performed well with Independent Q-Learning.
- Different LTL specification formats can lead to different performance in shaping.
- Potential Future Directions:
  - Extending this approach to a more general Markov Game case.
  - Negotiation or better priority management for epsilon-actions.
  - Explore more MARL algorithms





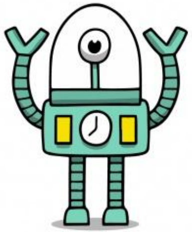
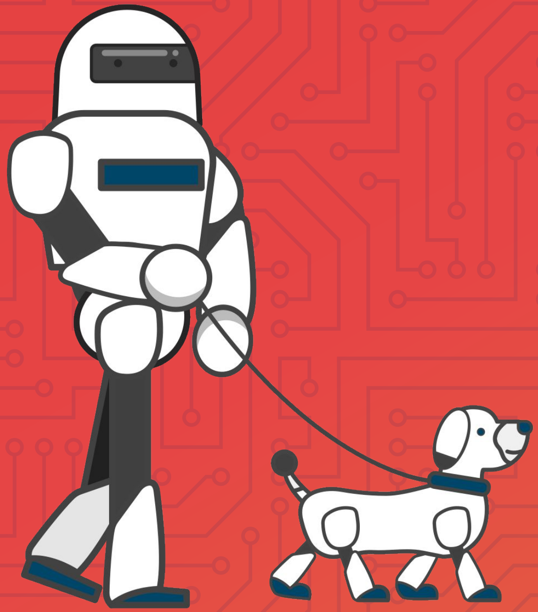


Photo Credit : Freepik



Thank you for your attention. Questions?

Image credit: rawpixel