# AdventumRL: A Quest Based Reinforcement Learning API

Kyle Xiao, Kushagr Singh, Mark Riedl

October 2019

**Abstract**

We propose AdventumRL, an API framework for complex mission-based reinforcement learning in a Minecraft setting. In this research, a set of encodings are defined on top of tools in Malmo, an open source Minecraft reinforcement learning framework. We propose a framework grammar for encoding states, actions, transitions, and goals that can represent these challenges for a hierarchical RL scenario, The described methods define a logical grammar and interaction encoding schema that can support quests with logical dependencies. Proof of concept reinforcement learning agents are constructed using a tabular agent and deep Q learner.

## 1 Introduction

Minecraft is an open-world sandbox construction game that provides users with the capability to explore and interact with entities. It provides a continuous state representation as well as a sufficiently expansive world where we can define complex hierarchical constructions and objectives [2]. Minecraft has proven to be a popular platform to teach agents via reinforcement learning. Reinforcement learning is the problem of learning a control policy $\pi(s_t, a_{t-1}) = a_t$ that provides an action $a_t$ that, if taken and the policy is followed henceforth, results in the maximum expected future reward. The learner is referred to as the agent, which selects actions in the environment and possibly receives a reward while a transition is made into a new state. Example of reinforcment learning agents in Minecraft include navigation-based tasks [8] [9] [5] and block placing [1] , which for human players are typically primitive actions that are performed to achieve a higher goal.

Project Malmo [6] provides an API for reinforcement learning agents to control virtual characters within a Minecraft map. Although Minecraft—and Malmo in particular—offers an expansive environment with many flexible features, the game itself has seen limited use for reinforcement learning involving sequences of subgoals (outside of building). Other computer games—such as role-playing games—and also the real world often involve the interleaving of navigation and interactions with objects and entities in order to execute a plan and achieve a goal. In computer games, these sequences of sub-goals are often referred to as *quests*. Quests can be simple, such as picking up and delivering an object to another entity, or complex, such as locating and using a set of keys to open doors in a particular order. In general, learning to solve quests in games is a precursor for more complex sequential tasks in the real world.

For the scope of this project, a quest is a series of discrete high-level actions the agent must perform to complete a game. A corresponding goal is a configuration of facts or propositions about the world that the agent must achieve through its actions. Quest-based reinforcement learning refers to a scenario of reinforcement learning that structures learning as a quest. This differs from general reinforcement learning in that multiple low level actions may be needed to make a single transition in the propositional configuration space, while general reinforcement learning techniques have no inbuilt structure separating

nascent, higher level facts and transitions about the world from the general state space.

We present a framework—*AdventumRL*[1]—that extends Project Malmo by providing a grammar for encoding states, actions, transitions, and goals that can be used to express quests in terms of high-level state abstractions (e.g., "the non-player character has the key") that are easy to read and write. This framework can help produce richer reinforcement learning tasks in Minecdraft by allowing developers to reduce joint-action spaces by keeping only high-level information, facilitate game theoretic payoff search scenarios through a context-free grammar, grant users the ability to assign individual rewards for accomplishing subgoals in cooperative tasks, and promote an inbuilt method for describing complicated propositional structures. Current state of the art reinforcement learning tools such as Malmo encode this information via MissionSpec, which defines a map, reward signal, consumable goods, and set of observations and actions [6]. However, these tools have difficulty communicating complex signals among multiple agents as well as high level objectives outside of navigation-based goals.

In AdventumRL, a set of encodings are defined on top of the current Malmo tools. States are represented via first-order logic on entities relevant to the problem space. The described methods define a logical grammar and encoding schema. A proof of concept reinforcement learning agent is then constructed.

Multi-agent reinforcement learning refers to multiple autonomous agents which can learn policies to interact with both with the environment and amongst themselves [7]. Current challenges within multi-agent reinforcement learning include joint-action spaces, game-theoretic events, credit assignment, and complex state representation [7].

## 2 Quest Grammar

The questing scenario is a proposed structure to reinforcement learning missions whereby high level representations are formed from more primitive state space information, inspired by Textworld [4]. This high level representation is referred to as the propositional space, which contain facts and propositions about the world inspired by first-order logic definitions. [10] Transitions in the propositional space must be achieved through actions satisfying a precondition and transitioning the propositional configuration space into a postconditions. These conditions thus form subgoals out of more primitive actions in the general state space.

This propositional encoding includes information on objects, relations, and functions. For example, we may want to encode a mission where the objective is to eat an apple and a carrot. The initial world propositional space may be

$$in(apple, chest) \ \wedge \ in(carrot, house)$$

With the action of taking the apple given as:

$$in(apple, chest) \quad \wedge \quad by(agent, chest) \quad \rightarrow \quad in(apple, inventory)$$

Actions apply transitions via a context free grammar on the current propositional space. Goals are represented as a set of facts or propositions in the propositional space (e.g. $\forall food | in(food, belly)$). From the logical encoding, we construct facts, statements, and actions using a grammar derived from first-order logic as well as the themes of the quest.

The construct of the grammar includes facts, propositions, actions, and rules as defined by first-order logic constructs. [10] A fact defines a relation among elements that is true in the context. A proposition defines a relation among subsets that is true in the context. An action defines a transition between a set of precondition facts to a set of postcondition facts. A rule defines a similar transition from a set of precondition propositions to a set of postcondition propositions.

We define several pre-built relations, which are automatically tracked by the API.

- **in**—Whether entity $A$ in the coordinate

[1] *Adventum* is latin for "adventure".

space is encapsulated by entity $B$, that is $Coord(A) \subset Coord(B)$ where $Coord()$ is the set of coordinates occupied by a given entity. This can be used, for example, to describe the state of the agent being inside of a building or other region that has been specifically marked out and given an identifier.

- **at**—Whether entity $A$ is in the coordinate space of another entity $(Coord(A) \cap Coord(B) \neq \varnothing)$.

- **by**—Whether entity $A$ is in within a range to entity $B$ for cases of interacting with an object (e.g. being in range to grab an apple off from a chest) $(\exists \delta < \epsilon | (Coord(A) + \delta) \cap Coord(B) \neq \varnothing)$.

- **unlocked**—A theme specific attribute relation for unlockable items.

- **inhand**—Whether an entity is currently at the top of the inventory and is usable in the world.

We allow users to specify constructs using these relations in a grammar JSON file. In addition, we also allow users to specify physical entities in the Minecraft world through the quest file, which are then tracked for the previous relations. These are useful for specifying artificial bounding-box boundaries.

Finally, we allow the user to specify *triggers*, which are facts or propositions that are made observable to the general state space. The user can choose what information to share with the agent that may be useful for decisions on low level tasks such as navigation. For example, consider the scenario where a task requires you to ride a boat across a river. Since the navigation challenges of navigating a ship are different than walking, it would be useful to have a feature that distinguishes these forms of locomotion. In this case, the user could set a trigger that distinguishes states where the agent is steering or not to facilitate training.

# 3 Muti-Agent RL

Multi-agent RL has multiple challenges that we propose the provided API can help solve. Current multi-agent reinforcement learning either utilizes independent agent observations of other actors or interpreting the joint action space of all agents together on the environment [3]. The former method limits the ability of autonomous agents to collaborate, while the latter is not a valid model under certain partially observable MDPs since it assumes all agent policies are fully observable. An encoding is needed to "communicate" between agents, or to add limited shared information among agents. This way, nascent behavior from individually trained agents interacting can be obtained.

Through the described propositional and triggering schema, an encoding can be made about high level information to communicate among agents. This can, for instance, communicate if a single agent has obtained a key item in the quest or learned some propositional information about previously unexplored areas. We posit that our API provides the building blocks for the representation and tracking of high level information that would be useful for representing a more natural, hierarchical agent cooperation model.

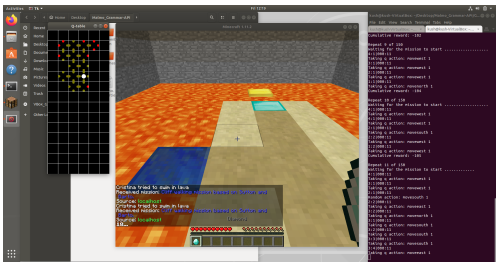
# 4 Content of Demo

For the purposes of the demo, we have provided a map whereby an agent must obtain a key and unlock a door to exit a room filled with lava. Therefore, the action leading to the goal configuration is:

$$locked(door) \hat{} by(player, door) \hat{} in(key, inventory) \hat{} inhand(key) \rightarrow\sim locked(door) \hat{} by(player, door)$$

We provide two premade agents in our API which can train on general questing missions. The TabQAgent provides a discrete q-learning agent which takes in the tabular position state space Q and a set of artificial boundary triggers J to help with convergence (i.e. the state space is $Q \times J$). The DQNAgent takes a naive one-hot embedding for our proposition space and state

space and similarly outputs a discrete action.



## 5 Related Work

The work in this paper is inspired by the Malmo API [6] as well as the Textworld platform [4]. The Malmo platform is a reinforcement learning platform which provides an interface into the Minecraft game engine, while the textworld platform inspired the described first-order logic definitions and context-free grammar parsing structure.

The construct described can also be related co-creativity [11], which is defined as the human and AI collaboration on creative artifacts. The message passing interface from the multi-agent scenario can similarly be used to interface with humans, as the propositional configuration space is explainable and interpretable.

## 6 Conclusion

In this paper, we described a novel API for the construction of questing-based reinforcement learning. We gave a set of value propositions and use cases for this API, and defined a structure for learning high-level constructs. We provided a set of proof-of-concept models and gave an interface for building quests and agents. We then proposed scenarios for novel exploration into the reinforcement learning space.

## References


[1] Stephan Alaniz. Deep reinforcement learning with model learning and monte carlo tree search in minecraft. *CoRR*, abs/1803.08456, 2018.

[2] Marc Brittain and Peng Wei. Hierarchical reinforcement learning with deep nested agents. *CoRR*, abs/1805.07008, 2018.

[3] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, March 2008.

[4] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

[5] Spencer Frazier and Mark Riedl. Improving deep reinforcement learning in minecraft with action advice, 2019.

[6] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 4246–4247. AAAI Press, 2016.

[7] Sanyam Kapoor. Multi-agent reinforcement learning: A report on challenges and approaches. *CoRR*, abs/1807.09427, 2018.

[8] Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *CoRR*, abs/1605.09128, 2016.

[9] Diego Pérez-Liébana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, André Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The multi-agent reinforcement learning in malmö (marlö) competition. *CoRR*, abs/1901.08129, 2019.

[10] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[11] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (PCGML). *CoRR*, abs/1702.00539, 2017.