Investigating Relational State Abstraction in Collaborative MARL

Sharlin Utke[1], **Jeremie Houssineau**[2], **Giovanni Montana**[1]

[1]University of Warwick, Coventry CV4 7AL, United Kingdom
[2]Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
sharlin.utke@warwick.ac.uk, jeremie.houssineau@ntu.edu.sg, g.montana@warwick.ac.uk

Abstract

This paper explores the impact of relational state abstraction on sample efficiency and performance in collaborative Multi-Agent Reinforcement Learning. The proposed abstraction is based on spatial relationships in environments where direct communication between agents is not allowed, leveraging the ubiquity of spatial reasoning in real-world multi-agent scenarios. We introduce MARC (Multi-Agent Relational Critic), a simple yet effective critic architecture incorporating spatial relational inductive biases by transforming the state into a spatial graph and processing it through a relational graph neural network. The performance of MARC is evaluated across six collaborative tasks, including a novel environment with heterogeneous agents. We conduct a comprehensive empirical analysis, comparing MARC against state-of-the-art MARL baselines, demonstrating improvements in both sample efficiency and asymptotic performance, as well as its potential for generalization. Our findings suggest that a minimal integration of spatial relational inductive biases as abstraction can yield substantial benefits without requiring complex designs or task-specific engineering. This work provides insights into the potential of relational state abstraction to address sample efficiency, a key challenge in MARL, offering a promising direction for developing more efficient algorithms in spatially complex environments.

Code & extended version with appendix — https://github.com/sharlinu/MARC

1 Introduction

Multi-Agent Reinforcement Learning (MARL) has emerged as an extension of single-agent RL, where multiple agents simultaneously interact with the environment to derive their optimal behavior by trial and error. Despite the complexity and dynamics of the learning environment, MARL holds significant promise for modeling real-world systems involving nuanced interactions between multiple entities, such as autonomous vehicle coordination (Shalev-Shwartz, Shammah, and Shashua 2016), traffic flow optimization (Agogino and Tumer 2012), and team robotics (Matignon, Jeanpierre, and Mouaddib 2012). These applications often involve collaborative or competitive dynamics that single-agent RL struggles to capture adequately. However, the increase in dimensionality over state and action spaces, the additional agent interactions, and the information influx this entails make sample efficiency a key challenge.

A critical aspect to sample efficiency is how agents represent the information of what they observe. The observation received by an agent can hold a lot of information, but not all of it is necessary to make an optimal decision. The ability to find abstract features allows for reasoning on a higher, conceptual level and adds robustness to small, task-irrelevant changes; a common principle for good representations (Bengio, Courville, and Vincent 2012). Abstraction leverages the underlying structure of the problem to focus on relevant information while reducing its complexity. That way, agents can learn optimal policies with fewer interactions, leading to improved sample efficiency and knowledge transfer to new situations (Mohan, Zhang, and Lindauer 2024).

A natural structure to present the environment is the decomposition into objects and their relations. Recent work in both deep single-agent RL and MARL has demonstrated the benefits of leveraging this relational structure as a graph-based representation in the learning architecture (Bapst et al. 2019; Jiang et al. 2021; Nayak et al. 2023; Agarwal et al. 2020), improving sample efficiency and generalization capabilities. This seems especially important in MARL, where the complexity often scales with the number of agents. The ability of graph convolutional networks (GCNs) (Scarselli et al. 2009; Welling and Kipf 2016; Gilmer et al. 2017) to model systems where the relationships between entities are critical has brought them to the forefront in many fields of multi-agent systems, ranging from modeling behavior and trajectories in multi-agent systems (e.g. Kipf et al. 2018; Tacchetti et al. 2019; Li et al. 2020; Kipf, van der Pol, and Welling 2020) to enhancing communication (e.g Niu, Paleja, and Gombolay 2021; Jiang et al. 2020; Zhang et al. 2021b).

In this study, we investigate the integration of a simple yet effective relational abstraction to MARL using the architectural flexibility of GCNs. We focus on collaborative tasks with high spatial complexity where direct communication between agents is not permitted due to cost or security constraints, e.g., in underwater robotics (Song, Stojanovic, and Chitre 2019). We specifically emphasize spatial relationships, as these are ubiquitous and readily available in real-world multi-agent scenarios. Spatial relations provide fundamental information about the relative positions

and orientations of agents and objects, which is crucial for navigation, coordination, and collaboration in physical environments. This focus allows us to explore how relational information can be leveraged through implicit information already present in the environment, without relying on explicit communication channels or complex architectural designs. Our research aims to address two key questions: (1) Can the incorporation of a state abstraction using spatial inductive biases improve sample efficiency and asymptotic performance in MARL? (2) How do different choices in the design impact the learning in such relational architectures?

To address these questions, we propose MARC (Multi-Agent Relational Critic), a simple multi-agent actor-critic architecture that abstracts the observation based on the relative positions of the entities into a graph-based representation. MARC utilizes a shared relational component within the critic architecture to efficiently learn a structured representation, further aiding sample efficiency. We conduct comprehensive empirical evaluation against state-of-the-art (SOTA) MARL baselines and across different tasks including a newly created collaborative multi-agent environment, designed to be a spatially demanding task between heterogeneous agents. Ultimately, we examine the impact of different design choices in our relational component in Section 5.

2 Related work

State Abstraction in RL Abstraction has been widely studied, with early work showing theoretical properties in single-agent RL (Li, Walsh, and Littman 2006; Abel 2022). Inspired by Zucker (2003) and focusing on state abstraction, we define abstraction as a mapping of the ground-truth representation of a state to a simpler, more compact representation by preserving desirable properties and removing less critical information. In other words, abstraction simplifies the information representation by dropping the information that is not essential to the task. Many methods have shown the success of embedding an abstract representation. For example, Kipf, van der Pol, and Welling (2020) factorize state inputs into objects and apply a relational, object-centric state abstraction to model a multi-object system. Zhang et al. (2021a) aim to learn an abstract state representation from high-dimensional observations based on the behavioral similarity between states to encode only information relevant to the task. Abdel-Aziz et al. (2024) reduce the computational complexity between communicating agents by learning a state abstraction based on quadtree decomposition (Samet 1984). Zhang et al. (2021b) use a state abstraction component in the MARL setting to reduce the high-dimensional observations into a more compact latent presentation using dense neural networks. While the abstraction method and assumption we take are different, we leverage these methods' underlying idea to discard any information irrelevant to the task to create a more compact and efficient representation.

Relational Representation in RL Before the integration of deep learning, traditional RL methods often falter in environments with relational structures or when generalization beyond initial training conditions is necessary. Relational RL addresses these challenges by learning the optimal policies over the objects and relations using a relational representation such as first-order logic. Whilst this approach has shown improved generalization and scalability, both in single-agent RL (Džeroski, De Raedt, and Driessens 2001; Sanner and Boutilier 2009; Driessens and Džeroski 2001) and in MARL (Croonenborghs et al. 2006; Ponsen et al. 2010; Li et al. 2022), the use of first-order logic comes with constraints, such as the need to hand-engineer features (Garnelo, Arulkumaran, and Shanahan 2016). Contemporary methods tackle this issue by learning the relations between objects using deep learning methods (Garnelo, Arulkumaran, and Shanahan 2016; Jiang et al. 2021; Zambaldi et al. 2019). These methods assume that observations comprise entities and the relationships between them while using deep learning methods as an inductive bias to learn over these structures. For example, Zambaldi et al. (2019) learns the importance of non-spatial relations between entities using attention mechanisms (Vaswani et al. 2017). They show superior performance and generalization capabilities compared to purely local relations in single-agent RL. Jiang et al. (2021) connect entities of a grid with a broader set of spatial relations, including remote relations, into a heterogeneous graph and passes them through a Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al. 2018). Their findings indicate that the imposition of structure by inducing a spatial bias can lead to improved asymptotic performance and generalization capabilities in single-agent tasks. However, they treat every cell in the grid as an entity, whether or not it contains an environment object. We extend this idea to MARL by employing a relational representation between agents and objects in the environment where the importance of the induced relations is implicitly learned using R-GCNs. Additionally, we only consider the environment objects, i.e. agents and other objects, as entities and use fewer relations, proposing a lean and abstract representation that better aligns with the computational complexity of MARL and works on continuous domains as well.

Relational Inductive Bias in MARL Relational inductive bias, loosely defined as the imposition of structural constraints on the learning process based on the relationship between objects (Battaglia et al. 2018), is a principle commonly embedded in MARL architectures.

A natural way to leverage structure in MARL is on the agent level. In fact, many of the commonly known MARL algorithms own an architecture that represents a form of relational inductive bias based on the structure between agents. For example, value decomposition methods such as QMIX (Rashid et al. 2018) assume conditional independence between the agents to decompose their value function. MAAC (Iqbal and Sha 2019) assumes that the influence of one agent's information on another can vary. Each critic can dynamically select which agents to focus on by assessing the relevance of the encoded information from other agents via a multi-head attention layer that is shared between critics. The actor-critic methods introduced by Liu et al. (2020) extend the constraint imposed in MAAC by using an additional hard-attention layer to strengthen the assumption that not all other agents' information is relevant to succeed, further re-

ducing the complexity of the game dynamics. Khan et al. (2019) leverage the underlying graph structure and symmetry between large numbers of homogeneous agents to parameterize the policies using a GCN framework.

Whilst the decomposition of the MARL architecture on the agent level is very intuitive, some methods leverage the structure already found within the observation. For instance, VMARL (Liu et al. 2021) transforms high-dimensional visual inputs to an object-centric intermediate state representation where environment objects are linked by their proximity, before being fed to policy and actor networks. MAGNet (Malysheva, Kudenko, and Shpilman 2019) considers all environment objects as entities when pre-training a static relevance graph with known node and edge types, which is then used to represent the observations. Agarwal et al. (2020) and Nayak et al. (2023) employ distance-based observation graphs with learned attention weights between agents and objects, demonstrating that graph representations of the environment allow for a framework invariant to permutation and the number of entities in the environment. They both assume shared rewards among homogeneous agents that allow for communication within their neighborhood. What most of these methods have in common is that they take advantage of the strong relational inductive bias posed by graph neural network architectures, enforcing learning over entities and their relations. We similarly leverage graph neural architectures to enforce a structured observation. However, we leverage inherent spatial symmetries to reduce the observation complexity and employ a MARL architecture also applicable to heterogeneous agents.

3 Methodology

3.1 Preliminaries

We work under the framework of partially observable Markov Games, with S being the state space in which each of the N agents has their own action space A_i, with $i = 1, \dots, N$, forming a joint action space $A = A_1 \times A_2 \times \cdots \times A_N$. After taking an action, each agent i receives an observation $o_i \in O_i \subset S$. Moreover, we assume individual reward functions, $R_i : S \times A \to \mathbb{R}$, which gives a reward signal after every step. At each time step, the agents simultaneously choose actions according to their respective policies, $\pi_i : O_i \mapsto P(A_i)$, which depend on the observation they receive. Consequently, the environment changes in line with the transition dynamics $T : S \times A \times S \to [0, 1]$ to a new state. The goal is that every agent finds the optimal policy that maximizes their expected cumulative return $J_i(\pi_i) = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in (0, 1]$ is the discount factor incorporating uncertainty about future returns, and where r_i^t is the individual reward received at time step t.

3.2 Abstract Observation Representation

Our objective is to design a sample-efficient multi-agent actor-critic architecture that decomposes the observations based on spatial inductive biases. We achieve this by employing a form of state abstraction: we simplify the observation representation by dropping information that is not essential to the task. This can also be described as domain reduction where we collapse observations into equivalent clusters, causing some observations to be indistinguishable and ultimately reducing observation complexity (Zucker 2003).

The abstraction assumption we make is that the relative positioning of entities is relevant, not their absolute positions. We are inducing an equivalence between observations, where we group observations with a similar spatial structure. This induces a translation invariance that applies to both remote and local relations in the observation space. For this to hold, we assume that the relative spatial relations can be extracted from the observation. This type of spatial information is inherent in many common environments and real-world scenarios and offers an intuitive example of using existing structures in the observations.

We hypothesize that this abstraction is particularly fruitful in discrete domains. Discrete states can be clearly separated from each other, which makes it easy to exactly define boundaries for any spatial relations. In contrast, continuous state spaces have a higher state complexity, as they are infinite expressions of the state and small changes can have a significant impact on the optimal action. Hence, clustering continuous state spaces can lead to a stronger loss of information in the representation that could impact performance (Li, Walsh, and Littman 2006). Whilst these challenges may affect the effectiveness of our abstraction, we test the robustness of our approach on continuous domains as well.

Effectively leveraging graph structures to impose such inductive bias in MARL poses the key challenges of (a) determining how entities are represented; (b) finding an informative yet efficient use of relations; (c) aggregating information across the graph to propagate relevant signals; (d) finding a computationally efficient way of incorporating the structural information into the MARL architecture. In the following, we address these challenges through specific design choices to create a structured, more compact observation representation that leverages the permutation invariance to the order of entities and the translation invariance to the absolute position between entities. An overview of the steps and our overall architecture can be seen in Figure 1.

Entity Representation In many of the discussed MARL methods, the focus lies on the interaction between agents, where the information they share is usually an encoding of their individual observation and action (e.g. Iqbal and Sha 2019; Liu et al. 2020; Jiang et al. 2020; Zhang et al. 2021b). We want to emphasize the structure already present within the observation itself. Hence, we aim to find a structured representation of the observation that does not only consider the agents but also all the other objects in the environment.

Typically, observations are given as fixed-sized vectors that contain the positions and attributes of agents and objects. This enforces an artificial ordering between the entities that is not desirable. The structure of such an observation is commonly a design choice and can be varied without great loss of generality. Consequently, we assume that the positions and attributes of the agents and environment objects can be extracted. In detail, we first construct an entity set $\mathcal{V}$ from all agents and objects. We then take the non-spatial information from all agents and objects, such as their level,

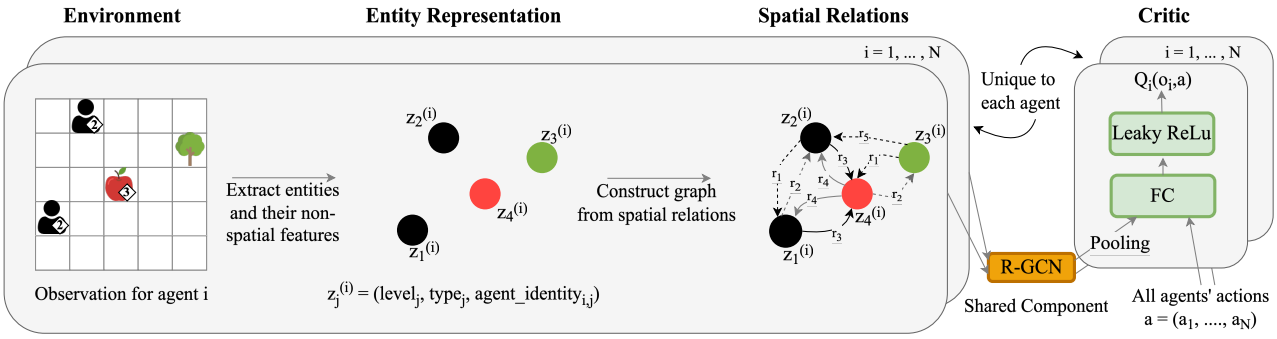


Figure 1: Overview of our MARC architecture on the example of level-based foraging. Without adding information, the observation is constructed as a graph, with objects and agents as entities and our chosen set of relations. We then pass this relational graph into a shared R-GCN component, followed by an individual head for each agent to estimate the state-action value.

type, if they are carrying objects, an identifier or other attributes. This results in a corresponding entity feature matrix $Z \in \mathbb{R}^{d \times |\mathcal{V}|}$ with d being the number of entity features.

Spatial Relations Our abstraction assumption is that only the relative spatial information is essential to solve the task. In line with that hypothesis, we do not consider the absolute position (x, y) of an entity as an entity feature but transform this information into relative spatial edges between all entities. These edges are established using spatial predicates $r(a, b) \leftarrow$ condition, such as $\mathit{left}(a, b) \leftarrow x_a < x_b$, which indicates that entity a is to the left of entity b. Our chosen set of relations $\mathcal{R} = \{\mathit{left}, \mathit{right}, \mathit{top}, \mathit{bottom}, \mathit{adjacent}, \mathit{aligned}\}$ are directed edges designed to balance sufficient expressiveness with computational efficiency. An evaluation of this selection along with the potential impact of additional relations, are discussed in Section 5. Since some entities, such as the agents, are dynamic, the spatial relations between entities change at every time step. Unlike in other methods (Nayak et al. 2023; Agarwal et al. 2020), the edges do not feature the distance between entities. This allows us to induce translation invariance, building a compact abstraction that treats observations with the same relative spatial structure as equivalent. We refer to the appendix for more details on the invariance of our abstraction.

Observation Encoding Having established the structure of the relational graph, our next objective is to obtain a higher-level representation of the observation, informed by the spatial relation between the entities. For this, we employ R-GCN (Schlichtkrull et al. 2018) updates, chosen for the ability to handle multiple relationship types. It updates entity representations by evaluating the entities' individual features and aggregating information from connecting entities depending on their relation type.

Formally, our graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, Z)$, where $\mathcal{V}$ is the set of all entities, $\mathcal{E}$ represents directed edges signifying relationships, $\mathcal{R}$ categorizes the types of these relationships and Z denotes the entity-feature matrix. The feature update for each entity $v \in \mathcal{V}$, initially represented by $z_v \in \mathbb{R}^d$, is governed by $z'_v = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} W_r z_u + W_0 z_v\right)$, where σ is a non-linear activation function. Each relation type r has an associated weight matrix $W_r \in \mathbb{R}^{d' \times d}$, customizing the update to the specific nature of the relationship. An auxiliary weight matrix $W_0 \in \mathbb{R}^{d' \times d}$ integrates the entity's original features. The term $\mathcal{N}_r(v)$ represents the neighboring entities of entity v for a given relation type r and the aggregation is normalized by $|\mathcal{N}_r(v)|$. After applying a number of R-GCN layers as specified, we obtain an updated feature matrix $Z' \in \mathbb{R}^{d' \times |\mathcal{V}|}$. To generate a fixed-size representation, we apply a feature-wise max-pooling operation, resulting in observation encodings $e(o_i) = \text{max-pool}(Z')$, where Z' implicitly depends on the original observation o_i.

This entire process - from initial graph construction through to the final pooling operation - acts as a unified observation encoder that aligns with common MARL environments. It transforms individual agent observations into a compact, relational representation, emphasizing the understanding of entity relationships while excluding nonessential details. Reducing the details in the representation enhances computational efficiency without sacrificing essential information for decision-making.

By transforming the observation into a graph representation and employing R-GCN updates followed by max-pooling, we obtain observation encodings that are invariant to the order of the input elements. This is a very desirable property, as there is no natural ordering between the objects in an environment. Furthermore, by fully removing the absolute information on positions and distance between entities, we not only reduce the observation complexity but also induce a translation-invariant representation.

Learning Algorithm Having encoded the observations into a relational graph, our next step involves feeding the relational representation into the MARL framework. In principle, the observation encoder is agnostic to the backbone MARL algorithm and we include a supporting experiment along with a discussion of this aspect in the appendix. Known for effectively balancing SOTA performance with scalability, we employ the popular centralized training with a decentralized execution regime. This framework enables agents to share information during training while maintaining individual decision-making during execution. To en-

hance scalability and efficiency even further, we depart from the common practice of feeding observation information from all agents into the critic architecture (e.g. Iqbal and Sha 2019; Nayak et al. 2023). Instead, the individual critic only receives the observation information from its own agent and exchanges information implicitly by collectively learning the parameters of the observation encoder, significantly reducing the input dimensionality to the critic. This shared module is complemented by individual heads for each agent, facilitating efficient learning while preserving the capacity for learning individualized behavior.

In detail, each agent, indexed by i, maintains its own critic and policy, allowing for distinct reward structures and action spaces. Formally, the critic for each agent is defined as $Q_{\psi_i}(o_i, a) = f_i(e(o_i), a)$, where f_i is a dense neural network that processes the encoded observation $e(o_i)$ together with the collective actions $a = (a_1, \ldots, a_N)$. Here, ψ_i includes the parameters from both the shared observation encoder e and the agent-specific dense layers f_i. The critics are jointly optimized to minimize the following regression loss:

$$\mathcal{L}_Q(\psi) = \sum_{i=1}^{N} \mathbb{E}_{(o_i,a,r_i,o_i')\sim D} \left[\left(Q_{\psi_i}(o_i, a) - y_i \right)^2 \right],$$

where the target is defined as $y_i = r_i + \gamma \mathbb{E}_{a'\sim\pi_{\bar{\theta}}} \left[Q_{\bar{\psi}_i}(o_i', a') - \alpha \log \pi_{\bar{\theta}_i}(a_i'|o_i') \right]$. Here, γ is the discount factor and D represents the replay buffer. Following the soft actor-critic updates (Haarnoja et al. 2018), we define $\bar{\psi}_i$ and $\bar{\theta}_i$ as the target critic and policy parameters for each agent, respectively, and α as the temperature parameter balancing entropy and reward maximization. The joint target policy vector $\pi_{\bar{\theta}} = (\pi_{\bar{\theta}_1}, \ldots, \pi_{\bar{\theta}_N})$ comprises policies, each a dense neural network. The individual policies are learned via gradient ascent as in the SAC (Haarnoja et al. 2018) framework and as described in (Iqbal and Sha 2019) without major modification. The implementation details and hyperparameters can be found in the appendix.

4 Experiments

In this section, we detail the experimental setup designed to evaluate the performance and capabilities of our proposed algorithm. We first describe the environments chosen, which are tailored to challenge and showcase the algorithm's spatial reasoning and collaborative capabilities in relationally complex tasks. We then outline the baseline algorithms against which we compare our approach to understand the added value of the relational inductive bias, followed by a comparative analysis of our results.

4.1 Environments

We hypothesize that the introduced abstraction learns effectively in spatially complex coordination tasks with sparse rewards. On this basis, we designed a new highly collaborative environment that requires coordination between different types of agents and several object types. Furthermore, we select other collaborative grid environments as they naturally challenge the algorithm's capabilities in spatial reasoning and cooperation under sparse rewards.

Whilst the employed state abstraction is particularly well suited to the discrete nature of these environments, we also evaluate our method on a continuous domain. Following is an overview of the chosen environment and for further details, we refer to the appendix.

Collaborative Pick and Place (CPP) is a new, collaborative environment with two types of agents that need to pick up and drop off a box at a designated goal, entailing heterogeneous, collaborative agents. Only the picker agents can collect a box whereas the delivery agents can only receive a box and drop it at at a goal location. Once a box is dropped at the goal location, no other box can be placed there. At the beginning of each episode, the boxes, agents and goals are randomly spawned on the grid. Depending on their role, agents receive a reward for successful pick-ups, passes, and drop-offs, as well as for prompt completion of the task. In our experiments, we test the challenging setting of a 10×10 grid, 2 picker agents, 2 delivery agents and 3 objects [1]. .

Level-based Foraging (LBF) (Christianos, Schäfer, and Albrecht 2020) situates agents in a grid world where they are rewarded for collecting fruits. As opposed to the original LBF environment, we assume that fruits are on trees that remain on the grid after the fruit has been collected, with a value of -1. This alteration demands a higher relational reasoning capability from agents, as they must now navigate around the trees, recognizing them as noncollectable obstacles. For testing high cooperation, our experiments run on a 10×10 grid with 4 agents and 4 foods, enforcing cooperation (denoted as 10x10-4a-4f-coop). To assess scalability, we extend the environment to a 15×15 grid with 8 agents and 1 fruit (denoted as 15x15-8a-1f-coop).

In **Wolfpack** (Rahman et al. 2023), 3 agents are placed in a 10×10 grid to capture 2 prey. In a departure from the original setup, we have introduced sparse rewards by removing additional rewards based on the proximity to prey, significantly weakening the learning signal.

The **Target** task, based on the multi-agent particle environment (Lowe et al. 2017) and modified by Nayak et al. (2023), is a continuous domain environment where a number of agents try to reach their target landmarks while avoiding collision with obstacles and other agents.

4.2 Baselines

In our study, we choose the baselines based on the following criteria: performance, reproducibility, ability to handle discrete action spaces and similarity to our approach. Following is an overview of the selected baselines; implementation details and hyper-parameter selection can be found in the appendix.

MAAC (Iqbal and Sha 2019) also uses SAC as the base RL algorithm. The use of attention between agents represents a different form of relational inductive bias on agent interaction rather than our object-centric representation.

GA-AC is the AC algorithm that makes use of the G2ANet mechanism (Liu et al. 2020). It builds on MAAC with an ad-

[1] We made the environment code available at https://github.com/gmontana/CollaborativePickAndPlaceEnv

ditional hard attention layer, which allows for an even more nuanced differentiation of information from other agents and represents an even stronger inductive bias than MAAC.

InforMARL (Nayak et al. 2023) introduces a distance-based graph representation of objects and agents that informs policy and critic networks, yielding a similarly structured observation encoding. Tailored to the multi-agent particle environment, it provides a relevant baseline for our experiments in the continuous domain.

QMIX (Rashid et al. 2018) leverages the structural assumption of conditional independence between agents' value functions to factorize it, yielding a rigorously implemented and strong baseline for comparison.

MAA2C (Papoudakis et al. 2020) is an on-policy approach that learns a centralized critic from joint observations without other agents' actions. It serves as a fast and strong baseline due to its absence of relational inductive bias, meaning it does not explicitly consider relationships between agents or entities in the environment.

MAPPO (Yu et al. 2022) is an extension of single-agent PPO (Schulman et al. 2017), noted for its performance and, similar to MAA2C, does not incorporate a relational inductive bias. It enhances sample efficiency through multiple updates on batches of training data.

4.3 Asymptotic Performance and Sample Efficiency in Discrete Domains

In this section, we present a comparative analysis of the asymptotic performance and sample efficiency as illustrated in Figure 2. Asymptotically, MARC is competitive and outperforms all baselines across the implemented tasks. Additionally, MARC demonstrates superior sample efficiency, learning all the tasks the fastest. In the LBF-15x15-8a-1f-coop task, MARC reaches an average performance of 99% after 5.9e5 environment steps, whereas the second-best algorithm, MAAC, takes 7.3 times the number of steps to reach the same performance.

The most significant margins in asymptotic performance are achieved in CPP and LBF-10x10-4a-4f-coop, where MARC achieves a performance gain of 69.9% and 35.2% respectively, as displayed in Figure 2a and Figure 2b. They require a high level of coordination and spatial understanding between entities to succeed in the task. In the LBF-10x10-4a-4f-coop setting, MARC reaches 26% of the maximum returns, on average, in 1e6 steps, while the second-best algorithm, MAPPO, reaches the same performance in 5.6 times the number of steps. MAAC performs relatively well in tasks that highly depend on coordination between agents, such as LBF-15x15-8a-1f-coop and Wolfpack, as visualized in Figure 2c and Figure 2d, respectively. However, MAAC's performance deteriorates in CPP and LBF-10x10-4a-4f-coop, where information about objects is essential to gain a good understanding of the environment. GA-AC and MAAC do not have a significant performance difference, indicating that the additional hard-attention layer on the agent interactions does not dramatically impact performance in spatially demanding tasks involving reasoning over environment objects. MAA2C and MAPPO perform reasonably well in LBF and Wolfpack. As both share the critic and policy network across agents, one hypothesis is that this proves beneficial in highly collaborative and homogeneous tasks.

Overall, the comparative analysis demonstrates the effectiveness of MARC in achieving superior asymptotic performance and sample efficiency in the selected multi-agent environments. The spatial inductive bias introduced in MARC proves to be beneficial in understanding the relationships between agents and objects, leading to faster learning and better asymptotic performance compared to the baselines.

4.4 Generalisation Performance

To assess the ability of our method to generalize to out-of-distribution settings, we evaluate our model trained on the most difficult scenario of LBF, 10x10-4a-4f-coop, where MARC achieves 81% of the maximum performance, on a varying number of agents and fruits. We then compare our algorithm by training the best-performing algorithm on this task, MAPPO, with the same varied number of fruits and agents. When reducing the number of agents available to collect fruits to 3, MARC still achieves 38% of the performance, whilst MAPPO's performance fully deteriorates to 0%. Increasing the number of agents by 1 makes the task easier and yields an improved performance of 93% vs. 88% for MAPPO. This indicates that MARC learns an invariance to the number of agents. The performance decreases to 59% with an increase in fruits (from 4 to 6 fruits), but given that the number of environment steps remains fixed it generally becomes more difficult to fulfill in time and can still be considered robust. In comparison, MAPPO's performance decreases by 40% down to 19%. An overview of all generalization results can be found in the appendix.

4.5 Extension to Continuous Domain

Algorithm	3 Agents (2×10^6 steps)	7 Agents (4×10^6 steps)
MARC	212.7 ± 5.7	468.2 ± 4.2
InforMARL	193.5 ± 4.3	426.1 ± 81.2
MAAC	236.1 ± 2.9	527.9 ± 5.4
GA-AC	$\mathbf{236.6 \pm 3.5}$	$\mathbf{530.6 \pm 3.4}$
MAA2C	233.5 ± 2.1	68.8 ± 393.4
MAPPO	109.0 ± 16.2	304.7 ± 6.0
QMIX	21.5 ± 14.8	-90.0 ± 79.6

Table 1: Asymptotic performance and standard deviation for the Target task, averaged across 3 seeds.

As seen in Table 1, MARC performs stronger than the SOTA graph-based algorithm InforMARL, underlining the strength of our graph design also in continuous domains. It is also competitive with the best-performing baselines, MAAC, GA-AC and MAA2C. Deeper analysis shows that the performance margin comes from MARC taking, on average, 1-2 steps longer to reach the target. There is a trade-off

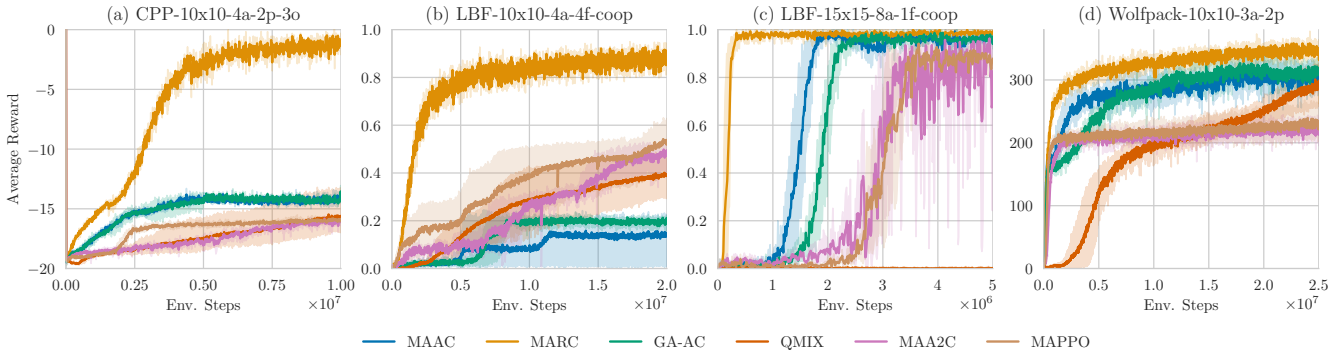


Figure 2: Mean average performance and 95% confidence interval for all discrete tasks. For each model, we run 3 random seeds.

in abstraction between a simple and efficient abstraction and removing too much information. For example, as environment objects also have velocity in the target task, the agents ideally have a more fine-grained understanding of the proximity to other objects, rather than just knowing once they are adjacent. This could lead to collisions that cannot be avoided or initially going passed their target due to their accumulated velocity. Further experiments, which can be found in the appendix, have confirmed this trade-off, indicating that a coarser abstraction yields improvements in sample efficiency with a decrease in asymptotic performance. Nevertheless, our algorithm demonstrates competitive and robust performance even for the continuous case.

5 Ablation Studies

Given the immense flexibility of graph architectures, we aim to shed light on how different design choices affect performance by systematically varying the following aspects:

Choice of Relations To understand how the choice of relations impacts performance, we evaluate our experiments with 3 different groups: our *default relations*, *local relations* representing a convolutional kernel, and *all relations* as the union of the two, detailed in the appendix. We found that purely local relations are not sufficient to learn the task, achieving only 10.6% of the performance achieved by the chosen architecture. This seems intuitive, as the agent gains a deeper spatial understanding if they can infer information from all entities, even if they are further away. Additionally, adding local relations to our default set does not elevate the performance, indicating that our default relations offer a sufficient and strong enough spatial bias.

Number of Entities We compared our approach of considering only agents and objects in the graph to using all grid elements as entities. Learning over the full grid compared to our choice of compact representation is, despite being more informative, not as sample efficient, reaching only 46.9% of the performance achieved by the chosen architecture in $8e6$ environment steps, along with a higher computational cost.

Choice of Graph Architecture We explore alternative choices of aggregating information from connecting entities in the graph. Whilst the choice is vast, we focus on previous work in the single-agent literature, where relational inductive bias between the entities is introduced via multi-head attention (Zambaldi et al. 2019). For this, we construct a binary graph and pass it through a Graph Attention Network (GAT) Veličković et al. (2018). Furthermore, we combine the approaches of spatial relations and varying importance between entities as in GATs by using an R-GAT layer (Busbridge et al. 2019) on the graph constructed in Section 3. For a detailed display of these alternative implementations, we refer to the appendix. Our R-GAT and R-GCN implementation yield indistinguishable performance, indicating that implicitly specifying different importance between entities does not yield a more expressive representation and its computation is therefore not required. In contrast, the use of a GAT layer yields suboptimal performance, asymptotically reaching only 23.4% of the chosen architecture's performance. The non-spatial, weighted interactions among entities might not serve as a robust inductive bias to effectively reason about the inherent structure of the task.

6 Conclusion and Future Work

In this work, we presented a relational state abstraction approach for MARL and demonstrated its effectiveness in environments requiring spatial reasoning and coordination among agents. By incorporating spatial inductive biases into our abstraction, we achieved significant improvements in sample efficiency and asymptotic performance compared to SOTA MARL algorithms. Our findings provide strong evidence for the potential of leveraging relational inductive biases to address the challenges of sample efficiency and generalization in MARL.

To further enhance our method, future research could explore the incorporation of inductive biases beyond spatial reasoning, an even stronger incorporation of structured representations, for example into the policy network as well, and the fine-tuning to more complex, high-dimensional environments. Investigating the interpretability and transparency of the structured representation could also facilitate the deployment into real-world scenarios.

Acknowledgments

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC EP/W523793/1), through the Statistics Centre for Doctoral Training at the University of Warwick and from a UKRI Turing AI Acceleration Fellowship (EPSRC EP/V024868/1).

References

Abdel-Aziz, M. K.; Elbamby, M. S.; Samarakoon, S.; and Bennis, M. 2024. Cooperative Multi-Agent Learning for Navigation via Structured State Abstraction. In *IEEE Transactions on Communications*.

Abel, D. 2022. *A Theory of Abstraction in Reinforcement Learning*. Ph.D. thesis, Brown University.

Agarwal, A.; Kumar, S.; Sycara, K.; and Lewis, M. 2020. Learning Transferable Cooperative Behavior in Multi-Agent Teams. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1741–1743.

Agogino, A. K.; and Tumer, K. 2012. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, 24: 1–25.

Bapst, V.; Sanchez-Gonzalez, A.; Doersch, C.; Stachenfeld, K. L.; Kohli, P.; Battaglia, P. W.; and Hamrick, J. B. 2019. Structured agents for physical construction. In *Proceedings of the 36th International Conference on Machine Learning*.

Battaglia, P.; Hamrick, J. B. C.; Bapst, V.; Sanchez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V. J.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261.

Bengio, Y.; Courville, A. C.; and Vincent, P. 2012. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1798–1828.

Busbridge, D.; Sherburn, D.; Cavallo, P.; and Hammerla, N. Y. 2019. Relational Graph Attention Networks. arXiv:1904.05811.

Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 10707–10717.

Croonenborghs, T.; Tuyls, K.; Ramon, J.; and Bruynooghe, M. 2006. Multi-agent Relational Reinforcement Learning. In *Learning and Adaption in Multi-Agent Systems*, 192–206.

Driessens, K.; and Džeroski, S. 2001. Integrating guidance into relational reinforcement learning. *Machine Learning*, 116–127.

Džeroski, S.; De Raedt, L.; and Driessens, K. 2001. Relational reinforcement learning. *Machine Learning*, 7–52.

Garnelo, M.; Arulkumaran, K.; and Shanahan, M. 2016. Towards Deep Symbolic Reinforcement Learning. arXiv:1609.05518.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, 1263–1272.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, 1861–1870.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Iqbal, S.; and Sha, F. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2961–2970.

Jiang, J.; Dun, C.; Huang, T.; and Lu, Z. 2020. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations*.

Jiang, Z.; Minervini, P.; Jiang, M.; and Rocktäschel, T. 2021. Grid-to-Graph: Flexible Spatial Relational Inductive Biases for Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 674–682.

Khan, A.; Tolstaya, E. V.; Ribeiro, A.; and Kumar, V. R. 2019. Graph Policy Gradients for Large Scale Robot Control. In *Conference on Robot Learning*, 823–834.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural Relational Inference for Interacting Systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2688–2697.

Kipf, T.; van der Pol, E.; and Welling, M. 2020. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations*.

Li, G.; Xiao, G.; Zhang, J.; Liu, J.; and Shen, Y. 2022. Towards Relational Multi-Agent Reinforcement Learning via Inductive Logic Programming. In *Artificial Neural Networks and Machine Learning*, 99–110.

Li, J.; Yang, F.; Tomizuka, M.; and Choi, C. 2020. EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning. In *Advances in Neural Information Processing Systems*, 19783–19794.

Li, L.; Walsh, T.; and Littman, M. 2006. Towards a Unified Theory of State Abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*.

Liu, I.-J.; Ren, Z.; Yeh, R. A.; and Schwing, A. G. 2021. Semantic tracklets: An object-centric representation for visual multi-agent reinforcement learning. In *International Conference on Intelligent Robots and Systems*, 5603–5610.

Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; and Gao, Y. 2020. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI conference on artificial intelligence*, 7211–7218.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*.

Malysheva, A.; Kudenko, D.; and Shpilman, A. 2019. MAGNet: Multi-agent Graph Network for Deep Multi-agent Reinforcement Learning. In *XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, 171–176.

Matignon, L.; Jeanpierre, L.; and Mouaddib, A.-I. 2012. Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017–2023.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533.

Mohan, A.; Zhang, A.; and Lindauer, M. 2024. Structure in Deep Reinforcement Learning: A Survey and Open Problems. *Journal of Artificial Intelligence Research*, 79.

Nayak, S.; Choi, K.; Ding, W.; Dolan, S.; Gopalakrishnan, K.; and Balakrishnan, H. 2023. Scalable Multi-Agent Reinforcement Learning through Intelligent Information Aggregation. In *Proceedings of the 40th International Conference on Machine Learning*, 25817–25833.

Niu, Y.; Paleja, R.; and Gombolay, M. 2021. Multi-Agent Graph-Attention Communication and Teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 964–973.

Papoudakis, G.; Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2020. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS Datasets and Benchmarks*.

Ponsen, M.; Croonenborghs, T.; Tuyls, K.; Ramon, J.; Driessens, K.; Herik, H.; and Postma, E. 2010. *Learning with Whom to Communicate Using Relational Reinforcement Learning*, 45–63. Studies in Computational Intelligence. Springer.

Rahman, A.; Carlucho, I.; Höpner, N.; and Albrecht, S. V. 2023. A general learning framework for open ad hoc teamwork using graph-based policy learning. *Journal of Machine Learning Research*, 24: 1–74.

Rashid, T.; Samvelyan, M.; Witt, C. S. D.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*.

Samet, H. 1984. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16: 187–260.

Sanner, S.; and Boutilier, C. 2009. Practical solution techniques for first-order MDPs. *Artificial Intelligence*, 173: 748–788.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20: 61–80.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, 593–607.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. arXiv:1610.03295.

Song, A.; Stojanovic, M.; and Chitre, M. 2019. Underwater Acoustic Communications: Where we Stand and What is Next? *IEEE Journal of Oceanic Engineering*, 44.

Tacchetti, A.; Song, H. F.; Mediano, P. A. M.; Zambaldi, V.; Kramár, J.; Rabinowitz, N. C.; Graepel, T.; Botvinick, M.; and Battaglia, P. W. 2019. Relational Forward Models for Multi-Agent Learning. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Welling, M.; and Kipf, T. N. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and WU, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, 24611–24624.

Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; Shanahan, M.; Langston, V.; Pascanu, R.; Botvinick, M.; Vinyals, O.; and Battaglia, P. 2019. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*.

Zhang, A.; McAllister, R. T.; Calandra, R.; Gal, Y.; and Levine, S. 2021a. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *International Conference on Learning Representations*.

Zhang, X.; Liu, Y.; Xu, X.; Huang, Q.; Mao, H.; and Carie, A. 2021b. Structural relational inference actor-critic for multi-agent reinforcement learning. *Neurocomputing*, 459: 383–394.

Zucker, J. 2003. A grounded theory of Abstraction in Artificial Intelligence. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 358: 1293–309.

7 Data Appendix

7.1 Environments

To explore the effectiveness of learning a relational state representation in MARL, we have selected a diverse set of environments that offer a suitable testbed for examining the learning of spatial relationships between pairs of entities. The chosen environments all involve multiple agents interacting with each other and their surroundings in ways that require them to reason about the relative positions, distances, and spatial configurations of entities in the environment.

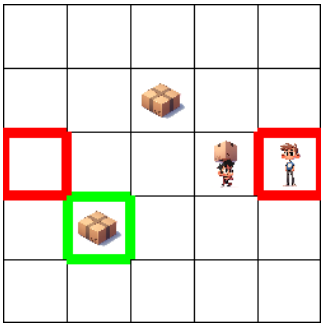


Figure 3: Collaborative pick and place environment on a 5x5 grid with 1 picker agent, 1 delivery agent, 3 boxes and 3 goal locations.

Collaborative Pick and Place The Collaborative Pick and Place (CPP) environment introduces a novel multi-agent challenge that involves heterogeneous agent roles working together to complete a task. In this environment, picker agents and delivery agents must cooperate to pick up boxes and drop them off at designated goal locations. The picker agents are responsible for collecting the boxes, while the delivery agents take the boxes from the picker agents and deliver them to the goal locations. Once a box is placed at a goal, it secures the spot, preventing any other boxes from being placed there. The agents operate within a grid world and, at the beginning of each episode, the environment is initialized with agents, boxes, and goals randomly distributed across the grid. An example of this environment involving two agents and two goals is illustrated in Figure 3.

Our experiments are conducted in a more complex 10×10 grid setup that includes 2 picker agents, 2 delivery agents, and 3 boxes. The agents have a set of six actions at their disposal: move up, down, left, right, pass a box, or wait. Rewards are assigned based on successful interactions between the agents and the environment. Picker agents receive rewards for picking up boxes, while both types of agents earn rewards for successful passes and drop-offs. To encourage cooperation and discourage redundancy, the reward structure is designed as follows: the first pass between a picker agent and a delivery agent grants each agent a reward of 0.5, while repeated passes of the same box result in a penalty of -1. To promote efficient task completion, agents receive a step penalty of -0.1 at each time step, incentivizing them to finish the task quickly. Additionally, if the agents complete the task within the 50-step limit per episode, they receive a completion bonus of 1.

The CPP environment offers flexibility in terms of grid size, the number of agents, and the number of boxes, making it adaptable to various experimental setups. This versatility allows researchers to investigate different aspects of multi-agent coordination and task allocation strategies among heterogeneous agents.

The observations entail information about the position, entity type, agent type, and whether or not an agent is carrying an object of all entities. The code is available at https://github.com/gmontana/CollaborativePickAndPlaceEnv.

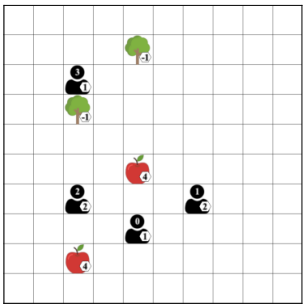


Figure 4: Level-based Foraging environment with 4 agents and fruits that become trees once picked.

Level-based Foraging The Level-based Foraging (LBF) environment, originally introduced by Christianos, Schäfer, and Albrecht (2020), places agents in a grid world where they are tasked with collecting fruits to receive rewards. The environment incorporates a level-based system that determines an agent's ability to collect a fruit. An agent can only collect a fruit if their level is equal to or higher than the fruit's level. This mechanic introduces a collaborative aspect to the task, as fruit levels can be set higher than the level of individual agents, requiring them to work together. This is a sparse environment, as the agents only receive a reward when they successfully collect a fruit, where the reward is proportional to the agent's level.

We have made a notable alteration to the original LBF environment as seen in Figure 4 by introducing the concept of trees. In our version, fruits are assumed to be on trees that remain on the grid even after the fruit has been collected. These trees have a value of -1 and serve as obstacles that the agents must navigate around. This modification adds a layer of complexity to the task, requiring agents to possess higher relational reasoning capabilities to recognize and avoid the noncollectable tree obstacles while searching for fruits [2].

To evaluate the agents' ability to cooperate under challenging conditions, we conduct experiments on a 10×10 grid with 4 agents and 4 fruits, enforcing cooperation (denoted as 10x10-4a-4f-coop). This setup presents a scenario with sparse rewards, demanding effective coordination among the agents to succeed. Furthermore, to assess the

[2] Our fork including our modifications for this environment can be found under https://github.com:sharlinu/lb-foraging

scalability of the agents' strategies, we extend the environment to a larger 15×15 grid with 8 agents and 1 fruit, still enforcing cooperation (denoted as 15x15-8a-1f-coop). This expanded setup tests the agents' ability to coordinate and adapt their strategies when working with a larger number of agents in a more complex environment.

The observations entail information about the position, entity type and level of all entities.

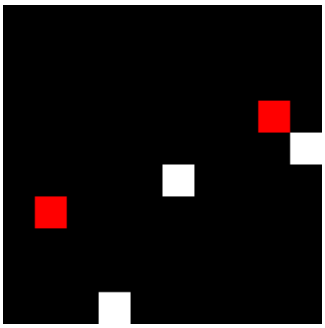


Figure 5: Wolfpack environment with 3 predator agents coordinating to catch 2 moving prey targets.

Wolfpack Wolfpack is a MARL environment inspired by the implementation of (Rahman et al. 2023). In this environment, a team of predator agents is tasked with capturing prey within a 2D grid world. The predators must learn to coordinate their actions and form packs to successfully surround and capture the prey. The objective of the predator agents is to capture the prey as efficiently as possible. To capture prey, at least two predator agents must surround it by occupying adjacent grid cells. When a prey is successfully captured, the predator agents involved in the capture are rewarded based on the size of the pack. The captured prey is then removed from the grid and respawned at a random location.

In our specific implementation, we place 3 predator agents and 2 prey in a 10×10 grid. The predator agents have full observability, meaning they can see the positions of all objects within the grid. However, the prey agents, which are trained using Deep Q-Networks (DQN) (Mnih et al. 2015), operate under partial observability and can only perceive a 3x3 grid centered on their position. The predator agents are allowed to move in any direction (up, down, left, right) or choose to remain stationary at each time step. The prey agents, on the other hand, follow their own learned policy based on DQN.

In a departure from the original setup, we have modified the reward structure to introduce sparse rewards. We have removed the additional rewards based on the proximity of predator agents to the prey, which were present in the original implementation. This change significantly weakens the learning signal, making the task more challenging for the predator agents to learn optimal coordination strategies [3].

The observations entail information about the position and agent type of all entities.

[3]Our fork including our modifications for this environment can be found under https://github.com:sharlinu/wolfpack

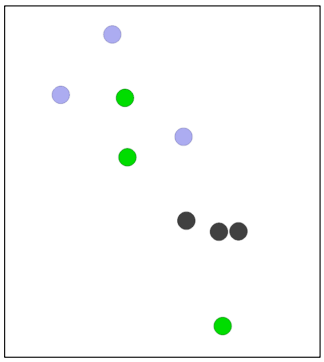


Figure 6: Navigation task with 3 agents aiming to reach target landmarks whilst avoiding obscuring obstacles.

Target In the target task, agents try to minimize the distance to specific target landmarks while navigating through moving obstacles and other agents. The environment rewards efficient pathfinding and penalizes collisions, forcing agents to balance speed and caution. This setup requires agents to handle various movements and interactions, similar to real-world scenarios. In our experiment, we test this setting with 3 and 7 agents, that need to reach their assigned markets (3 and 7 targets respectively) whilst avoiding collision with 3 obstacles and the other agents. The observations entail information about the position, velocity and entity type of all entities.

7.2 Additional Plots

We provide the performance curves for the continuous tasks in Figure 7 and for the ablation studies in Figure 8.

8 Technical Appendix

8.1 Constructing Edges for the Observation Graph

To model interactions and proximities in the observation graph, we define relationships between entities based on their spatial arrangements. These relationships are categorized into three distinct groups: Remote Relations, Contiguous Relations, and Local Relations. Each group serves a specific purpose and represents different levels of proximity and interaction potential between entities.

Remote Relations Remote relations identify long-range interactions where entities do not need to be immediately adjacent:

$$\begin{aligned} \text{left}(a,b) &\leftarrow x_a < x_b, \\ \text{right}(a,b) &\leftarrow x_a > x_b, \\ \text{down}(a,b) &\leftarrow y_a < y_b, \\ \text{top}(a,b) &\leftarrow y_a > y_b, \end{aligned}$$

Contiguous Relations Contiguous relations define direct adjacency or alignment, suitable for modeling interactions within immediate reach:

$$\begin{aligned} \text{aligned}(a,b) &\leftarrow (x_a = x_b) \wedge (y_a = y_b), \\ \text{adjacent}(a,b) &\leftarrow (|x_a - x_b| \leq 1) \vee (|y_a - y_b| \leq 1), \end{aligned}$$

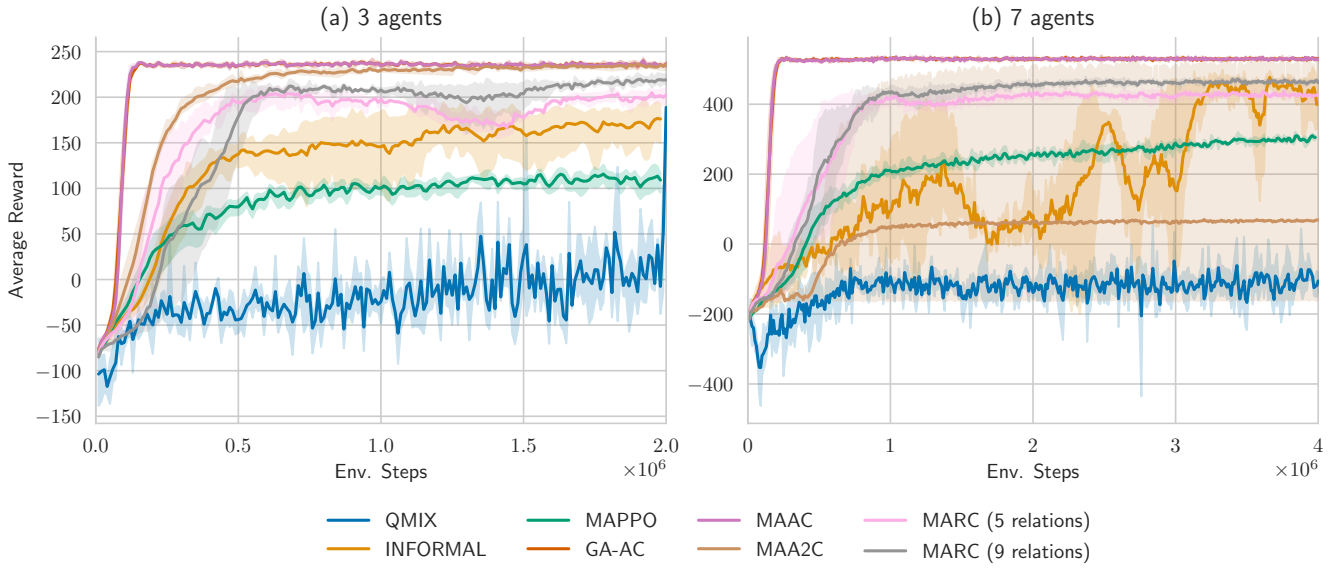


Figure 7: Mean average performance and 95% confidence interval for the continuous target tasks. For each model, we run 3 random seeds.

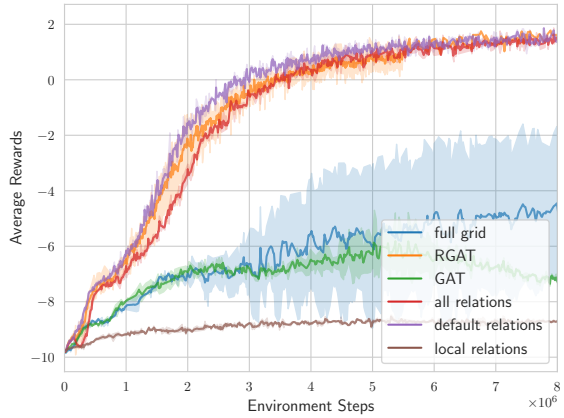


Figure 8: Training curves for a 10x10 CPP environment with 1 picker agent, 1 dropper agent and 2 objects, showing performance for varying graph architectures, sets of relations and number of entities, averaged across 2 seeds.

Local Relations Local relations are more granular, detailing the specific neighboring positions around an entity, and are crucial for detailed spatial reasoning:

$$\begin{aligned}
\text{rightAdj}(a,b) &\leftarrow (x_a = x_b + 1) \wedge (y_a = y_b),\\
\text{leftAdj}(a,b) &\leftarrow (x_a = x_b - 1) \wedge (y_a = y_b),\\
\text{topAdj}(a,b) &\leftarrow (x_a = x_b) \wedge (y_a = y_b + 1),\\
\text{bottomAdj}(a,b) &\leftarrow (x_a = x_b) \wedge (y_a = y_b - 1),\\
\text{bottomLeftAdj}(a,b) &\leftarrow (x_a = x_b - 1) \wedge (y_a = y_b - 1),\\
\text{bottomRightAdj}(a,b) &\leftarrow (x_a = x_b + 1) \wedge (y_a = y_b - 1),\\
\text{topLeftAdj}(a,b) &\leftarrow (x_a = x_b - 1) \wedge (y_a = y_b + 1),\\
\text{topRightAdj}(a,b) &\leftarrow (x_a = x_b + 1) \wedge (y_a = y_b + 1),
\end{aligned}$$

An illustrative example of a few of these relations can be seen in Figure 9. For our ablation studies, we categorize the relations into specific groups based on their use:

- **Default set of relations:** These include the most commonly used spatial relationships which cover basic proximity and directional interactions. The set comprises:

$$\{\text{adjacent}, \text{aligned}, \text{left}, \text{right}, \text{top}, \text{bottom}\}$$

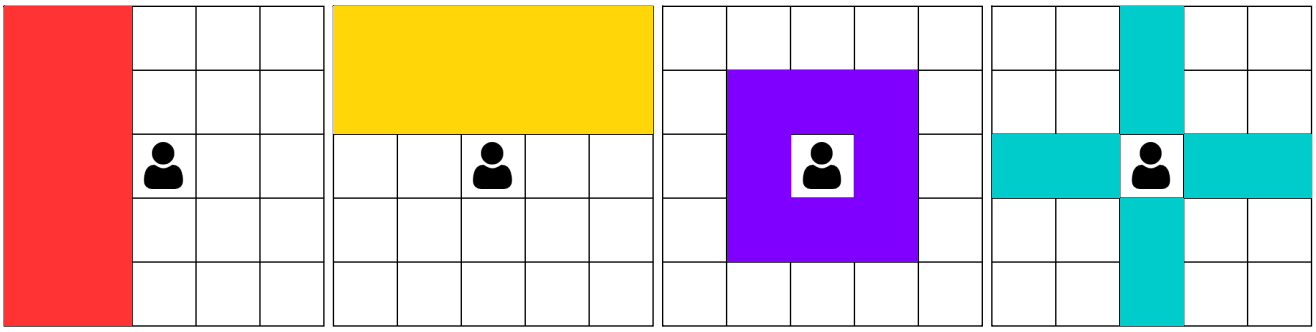
- **Local set of relations:** This set includes more detailed and localized spatial relations, providing finer control and specificity for modeling interactions:

$$\{\text{leftAdj}, \text{rightAdj}, \text{topAdj}, \text{topLeftAdj}, \text{topRightAdj}, \text{bottomAdj}, \text{bottomLeftAdj}, \text{bottomRightAdj}\}$$

- **Set of all relations:** Combines both default and local relations for comprehensive coverage:

$$\{\text{default relations} \cup \text{local relations}\}$$

Default relations are applied in the discrete task unless specified otherwise, offering a balance between computational efficiency and the resolution of spatial relationships.



(a) left (b) top (c) adjacent (d) aligned

Figure 9: Examples of the spatial determination rules employed in our methodology: given an entity and its position on a grid, the colored areas represent the areas under which a specific relational rule would hold if another entity is positioned there.

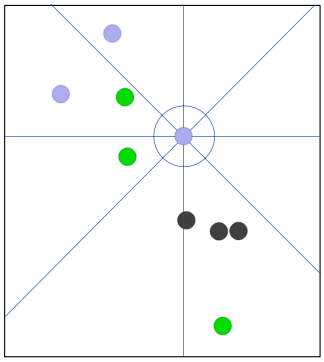


Figure 10: Example of the spatial clusters created by the 9 relations chosen for the continuous target task. Relations are shown with respect to one entity.

Relations for Continuous Domain For the continuous domain, we ran our experiment with default relations as described above, apart from the aligned relation, as the determination of that relation is not concise in the continuous domain. Similarly to our ablation studies on the discrete domain, we tested the addition of more detailed relations on the continuous domain. In this case, we added relations describing octagonal directions, plus adjacency, as depicted in Figure 10. We found that the addition decomposes the state in more fine-grained areas and improves asymptotic performance as shown in Figure 7. This again highlights the trade-off between having a compact, more efficient representation and optimal performance.

8.2 Training Procedure and Hyperparameter

The experiments were conducted on a workstation equipped with four NVIDIA RTX A6000 GPUs (46 GB each) and an AMD EPYC 7452 32-core Processor CPU (64 cores) with 251 GB RAM. The software environment consisted of Ubuntu 20.04.6, Python 3.9 and CUDA 12.4. The Python packages needed can be found in the `requirements.txt` file in our code base at https://github.com/sharlinu/MARC.

Our implementation of MARC follows Algorithms 1 and 2. To find the optimal hyperparameters, we engaged in a random search for MARC on the complex task of LBF-10x10-4a-4f-coop, selecting the hyperparameters that perform best in terms of asymptotic performance. These hyperparameters, documented in Table 2, are then applied consistently across all scenarios and are summarized in the following training procedure: We add each environment transition to a buffer of 1e5 transitions. The performance of MARC is robust to changes in the buffer length, which is why we chose a reduced buffer length to save memory. After every 100 environment steps, we perform 4 updates for both critics and policies using a batch size of 1024 samples. These updates are conducted using the Adam optimizer (Kingma and Ba 2015) with a learning rate of 0.001 for both networks. The hidden dimension for all networks is uniformly set to 128, and the discount factor γ is set at 0.99. Subsequently, we update the target networks using soft actor-critic updates, employing an update rate τ of 0.001. Additionally, the temperature parameter α is set to 0.05. For the policy network, we employ a simple 3-layer dense network. For the critic, we use a single-layered encoder to generate a dense feature representation out of the sparse entity features. These entity embeddings are then passed through a single-layered, shared R-GCN module. To output Q-values, we pass these entity embeddings through a max-pooling layer, generating a single vector representation that can be fed into a dense layer. Post concatenation with all agents' actions, these outputs traverse a final 2-layer dense network, unique to each agent.

8.3 Baseline Implementations

In the following, we list the code bases used to reproduce and test the baseline algorithms:

- MAA2C, MAPP0 (discrete), QMIX (discrete): https://github.com/uoe-agents/epymarl
- InforMARL, MAPPO (continuous), QMIX (continuous): https://github.com/nsidn98/InforMARL
- GA-AC and MAAC : https://github.com/shariqiqbal2810/MAAC

Epymarl (Papoudakis et al. 2020) is a thoroughly tuned and benchmarked codebase for MAPPO, MAA2C and

Parameter	MAAC/GA-AC	MARC
Dense layers in policy network	2 | **3**	2 | **3**
Dense layers in critic head	**2**	**2**
R-GCN layers	-	1 | **2** | 3
Entity embedding hidden dimension	-	8 | 16 | 32 | **48** | 64 | 128
Discount (γ)	**0.99**	**0.99**
Replay buffer length	10^5 | $\mathbf{10^6}$	$\mathbf{10^5}$ | 10^6
Critic learning rate	**0.001** | 0.005	**0.001** | 0.005
Policy learning rate	**0.001** | 0.005	**0.001** | 0.005
Critic hidden dimension	64 | **128** | 256	64 | **128** | 256
Policy hidden dimension	64 | **128** | 256	64 | **128** | 256
Attention heads	1 | 2 | **4** | 6	-
Batch size	512 | **1024**	512 | **1024**
Entropy coefficient (α)	**0.01**	**0.01**
Nonlinearity	**LeakyReLU**	**LeakyReLU**
Soft actor update rate (τ)	0.001 | **0.005** | 0.01	**0.001** | 0.005 | 0.01
Update interval in steps	**100**	**100**
Number of updates	**4**	**4**
Reward normalization	False | **True**	False | **True**

Table 2: Hyperparameter selection for MAAC, GA-AC and MARC. The table shows the range of evaluated hyperparameters, with the selected ones in bold.

Algorithm 1: Pseudocode for Multi-Agent Relational Actor-Critic

Initialize the environment and replay buffer D
$T_{\text{update}} \leftarrow 0$
for $t = 1$ to num episodes **do**
  Reset environment and get initial observation o_i for each agent $i = 1, \ldots, N$
  while $\text{num}_{\text{steps}} < \text{max}_{\text{steps}}$ or episode $\neq$ terminated **do**
    for each agent i **do**
      Select action $a_i \sim \pi_{\theta_i}(\cdot|o_i)$
      Do action and receive next observation o_i' and reward r_i
      Store transitions (o_i, a_i, r_i, o_i') in D
      $o_i \leftarrow o_i'$
    end for
    $T_{\text{update}} \leftarrow T_{\text{update}} + 1$
    if $T_{\text{update}} \geq \text{min}_{\text{update\_steps}}$ **then**
      Sample a subset B random transitions from D
      for $j = 1$ to num network updates **do**
        UpdateCritics(B)
        UpdatePolicies(B)
      end for
      Soft update target parameters for all agents i:
      $\bar{\psi}_i \leftarrow \tau\bar{\psi}_i + (1-\tau)\psi_i$
      $\bar{\theta}_i \leftarrow \tau\bar{\theta}_i + (1-\tau)\theta_i$
      $T_{\text{update}} \leftarrow 0$
    end if
  end while
end for

Algorithm 2: Update Functions for Critic and Policies

Function UpdateCritics (B)
for all agents $i = 1, \ldots, N$ and all transitions $b \in B$, in parallel **do**
  Calculate $Q_{\psi_i}(o_i^b, a_i^b)$
  Calculate $a_i'^b \sim \pi_{\bar{\theta}_i}(o_i'^b)$ using target policy
  Calculate $Q_{\bar{\psi}_i}(o_i'^b, a'^b)$ using target critic
end for
Update critics by minimizing the joint regression loss $\mathcal{L}_Q(\psi)$

Function UpdatePolicies (B)
for all agents $i = 1, \ldots, N$ and all transitions $b \in B$, in parallel **do**
  Keep o_i^b and discard the rest of the transition
  Sample new actions $a_i^b \sim \pi_{\theta_i}(o_i^b)$ for each agent
  Calculate $Q_{\psi_i}(o_i^b, a^b)$ using newly sampled actions
  Update policies using $\nabla_{\theta_i} J(\pi_{\theta_i})$
end for

QMIX that we used as baseline implementation for discrete tasks. We also leverage the comprehensive hyperparameter search conducted by Papoudakis et al. (2020), adopting the optimal hyperparameters and architectures identified by the authors for the LBF environment and applying them consistently across our discrete tasks. Equally, InforMARL (Nayak et al. 2023) was designed for and thoroughly tested on the continuous target task, using QMIX and MAPPO as their baselines. We therefore took their implementation and optimized hyperparameters for the continuous target task. Hence, for MAA2C, QMIX, MAPPO, and InforMARL we refer to paper and code base, where selected hyperparameters along with the matching training procedure are clearly and thoroughly documented. MAAC and GA-AC were not tested on the tasks that we used in our experiments so we conducted a random hyperparameter search for MAAC/GA-AC on the LBF-10x10-4a-4f-coop task, evaluated on asymptotic performance, and report the range and selection along with MARC in Table 2. We note that GA-AC is based on the MAAC architecture with an additional hard-attention layer, so the implementation details for MAAC and GA-AC are equivalent.

8.4 Alternative MARL Backbone

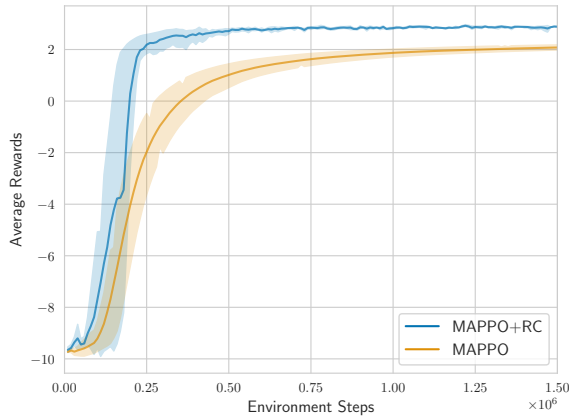


Figure 11: Mean average performance and 95% confidence interval for a test version of CPP on a 7x7 grid with 1 picker agent, 1 delivery agent, 1 box and 1 goal locations. MAPPO denotes the original algorithm and MAPPO+RC the combination of MAPPO with a relational critic. For each model, we run 3 random seeds.

In principle, relational abstraction is implemented as an observation encoder independent of the MARL algorithm itself. We therefore test this hypothesis by combining the relational observation encoder detailed in this paper with MAPPO (Yu et al. 2022), a strong and popular baseline algorithm. We compare our implementation with the original MAPPO implementation, setting the hyperparameters to be the same for a fair comparison. As seen in Figure 11, combining the observation encoder in the MAPPO critic architecture, we find an improvement in sample efficiency and asymptotic performance for the considered task.

8.5 Invariances of the State Abstraction

By using graph neural network computations, we inherently benefit from its invariance to the order of the input elements (Hamilton, Ying, and Leskovec 2017). This is a very desirable property, as there is no natural ordering between the objects in an environment. Our constructed state mapping aggregates states together based on their relative spatial similarity. By fully removing the absolute information on positions and distance between entities, we not only reduce the state complexity but also induce a translation-invariant representation of the state. Translation shifts of the absolute positions of the environment objects do not influence the relative positioning of the environment objects, leaving the final structured representation unchanged. To demonstrate this, we give an example of what observations would be considered equal in the critic network in Figure 12.

Formally, we consider the translation operator $T_{(a,b)}$ that shifts the position of elements of a function f by (a, b), i.e. $(T_{a,b}f)(x, y) = f(x - a, y - b)$. Translation invariance means that

$$T_{(a,b)}f(x, y) = f(x, y),$$

i.e. the translation of the input leaves its output unchanged. In our case, the only dependence on the position of an environment object (x, y) lies in the construction of our edge set via our relational rules. More specifically, the relational rules are binary functions that depend on the absolute position of the entities u and v they are comparing. To demonstrate this in an example, we can rewrite the left relation as:

$$r(u(x, y), v(x, y)) = \begin{cases} 1 & \text{if } x_u < x_v \\ 0 & \text{otherwise.} \end{cases}$$

Applying the translation operator shifts the position of entities u and v. So the translated relational rule can be written as:

$$\begin{aligned} r(T(u), T(v)) &= \begin{cases} 1 & \text{if } x_u - a < x_v - a \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } x_u < x_v \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

This ultimately implies translation invariance $r(u, v) = r(T(u), T(v))$, and can be shown for all spatial relational rules we employ.

8.6 Different Graph Architectures

To introduce a GAT-layer (Veličković et al. 2018), we construct a complete binary graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', Z')$, where, as before, $\mathcal{V}$ consists of N agents and M objects. However, for the GAT implementation, we drop all relation types and fully connect all entities with each other. To add back the spatial information, we enrich the feature matrix $Z' \in \mathbb{R}^{(d+2)\times|\mathcal{V}|}$ with two additional dimensions for the coordinates of the

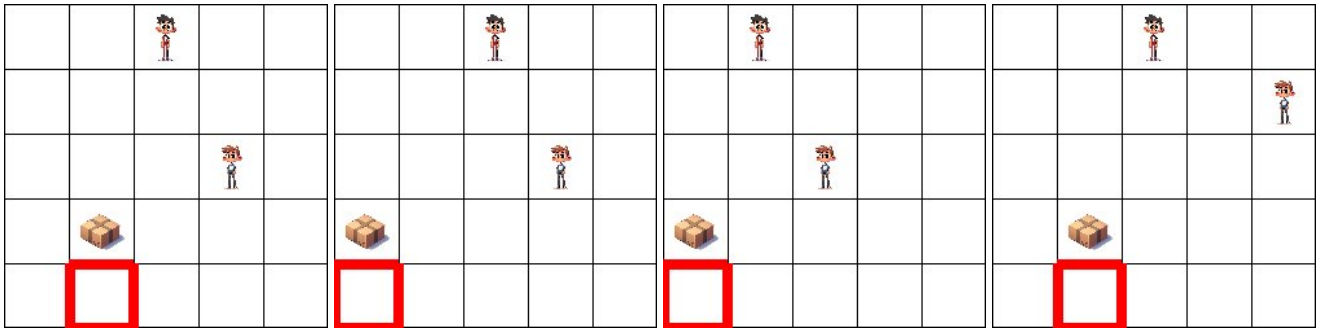


Figure 12: Examples of states that are considered to be equivalent in our critic architecture: translation shifts do not affect the representation if the relative spatial structure remains the same.

entities. The update of the entity features is as follows:

$$a(z_i, z_j) = \mathrm{softmax}\left(\mathrm{LeakyReLU}\left(q^T W z_i + k^T W z_j\right)\right),$$

$$z_i' = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} a(z_i, z_j) W z_j\right)$$

where $q, k \in \mathbb{R}^{d'}$ and $W \in \mathbb{R}^{d' \times d}$ are weight vectors and matrix, respectively. $a(z_i, z_j)$ is a learnable, self-attention weight that implicitly computes the importance of entity j to the representation of entity i. That way, we can learn the importance of connection between the agents and other objects in the environment.

One can combine the GAT architecture with an R-GCN by following Busbridge et al. (2019) in applying attention weights to edges of a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, Z)$, as it is defined earlier. This yields the following update:

$$a_r(z_i, z_j) = \mathrm{softmax}\left(\mathrm{LeakyReLU}\left(q_r^T W_r z_i + k_r^T W_r z_j\right)\right),$$

$$z_i' = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} a_r(z_i, z_j) W_r z_j\right).$$

where $q_r, k_r \in \mathbb{R}^{d'}$ and $W_r \in \mathbb{R}^{d' \times d}$ are relation-specific weight vectors and matrix, respectively. $a_r(z_i, z_j)$ are, as before, learnable self-attention weights but now depend on the specific relation type. The $\mathrm{softmax}(\cdot)$ is now computed across all connecting entities irrespective of their relations. That means that $a_r(z_i, z_j)$ implicitly computes the importance of entity j to the representation of entity i under relation type r, compared to all incoming connections to entity i. The difference between the R-GAT and R-GCN update is that in R-GCN, each neighboring entity has equal importance and is simply weighted with a relation-specific normalizing constant, i.e. $a_r(z_i, z_j) = |\mathcal{N}_r(i)|^{-1}$.