Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 1061–1070

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Utilising Assured Multi-Agent Reinforcement Learning within Safety-Critical Scenarios

Joshua Riley*[a], Radu Calinescu[a], Colin Paterson[a], Daniel Kudenko[b], Alec Banks[c]

[a]*Dept. of Computer Science, University of York, York, UK*
[b]*L3S Research Centre, Leibniz Universität Hannover, Hanover, Germany*
[c]*Defence Science and Technology Laboratory, Ministry of Defence, Salisbury, UK*

Abstract

Multi-agent reinforcement learning allows a team of agents to learn how to work together to solve complex decision-making problems in a shared environment. However, this learning process utilises stochastic mechanisms, meaning that its use in safety-critical domains can be problematic. To overcome this issue, we propose an Assured Multi-Agent Reinforcement Learning (AMARL) approach that uses a model checking technique called quantitative verification to provide formal guarantees of agent compliance with safety, performance, and other non-functional requirements during and after the reinforcement learning process. We demonstrate the applicability of our AMARL approach in three different patrolling navigation domains in which multi-agent systems must learn to visit key areas by using different types of reinforcement learning algorithms (temporal difference learning, game theory, and direct policy search). Furthermore, we compare the effectiveness of these algorithms when used in combination with and without our approach. Our extensive experiments with both homogeneous and heterogeneous multi-agent systems of different sizes show that the use of AMARL leads to safety requirements being consistently satisfied and to better overall results than standard reinforcement learning.

Keywords: Reinforcement Learning; Multi-Agent Systems; Quantitative Verification; Assurance; Multi-Agent Reinforcement Learning; Safety-Critical Scenarios; Safe Multi-Agent Reinforcement Learning; Assured Multi-Agent Reinforcement Learning

1. Introduction

Multi-agent systems (MAS), in which a collection of agents pursue common goals in a shared environment, currently have a great deal of untapped potential within real-world applications [1] as diverse as search and rescue [2], security applications [3], and healthcare [4]. These systems could be particularly beneficial in *safety-critical scenarios* [5] where it may be unsafe for humans to operate, e.g. dealing with the Fukushima nuclear power plant disaster, where robotic agents have been used in highly irradiated areas [6]. Despite these potential advantages, MAS are very

* Corresponding author. *E-mail:* jpr538@york.ac.uk

10.1016/j.procs.2021.08.109

challenging to develop. One reason for this is that as each agent moves through the shared environment, its actions and behaviours modify the environment for all agents [7].

These challenges are further exacerbated in safety-critical scenarios, where humans are often removed, thus requiring the system to handle the problem autonomously. One way of overcoming these challenges is to make use of Multi-agent reinforcement learning (MARL). Reinforcement Learning (RL) [8] is a widely used technique with roots in behavioural psychology, allowing agents to learn how to meet objectives and goals efficiently through punishments and rewards [9]. MARL then extends RL into MAS [10]. However, the stochastic nature of the learning process used for RL and MARL limits its use in safety-critical scenarios.

Agents that operate within safety-critical scenarios must consider hazards within the environment, such as radiation exposure or unstable ground, which is counter-productive when paired with the unpredictability within RL. If these hazards are disregarded, the unpredictability can lead to unwanted outcomes, such as damage to the agents, the environment, or humans.

While techniques have been proposed to allow for hazard awareness in RL within safety-critical scenarios [11, 12, 13], most methods have limitations and rarely consider MARL. As such, these techniques do not allow for assurances to be made concerning risk or safety more generally.

In recent work [14], we introduced a preliminary variant of an Assured Multi-Agent Reinforcement Learning (AMARL) approach that provides assurances for safe MARL by using quantitative verification. This preliminary AMARL variant supports simple systems comprising two identical agents that employ Independent Q-learning [15]. In this paper, we significantly extend AMARL with the following main contributions:

1. support for agents (i.e., robots) with heterogeneous capabilities;
2. integration of quantitative verification with the additional MARL learning algorithms Team-Q [16] and WoLF-PHC [17];
3. ability to handle more than two agents;
4. an extensive evaluation that shows the effectiveness of our approach within three problem domains.

The rest of this paper is structured as follows. Section 2 defines concepts and terminology used to present our approach. Section 3 compares our solution with related work. Section 4 introduces our AMARL approach. Section 5 details three problem domains used for evaluating AMARL. Section 6 presents the evaluation and results of AMARL. Finally, section 7 concludes our findings and suggests potential future work.

2. Background

Markov Decision Process (MDP) [8] is a formalism used to model sequential decision-making problems that possess stochastic behaviour. An MDP is a tuple (S, A, P, R) where: S is a finite set of states representing the current environmental state. A is a finite set of actions representing the capabilities of the agents. $P : S \times A \times S$ is then a state transition function such that for any $s, s' \in S$ and $a \in A$, which is an allowed action in s, s'. $P(s, a, s')$ is the probability of transitioning to state s' when undertaking action a in state s. $R : S \times A \times S \rightarrow \mathbb{R}$ is a reward function such that $R(s, a, s') = r$ is the reward associated with taking an action a in state s which leads to s'.

Abstract MDP (AMDPs) are high-level representations of an MDP in which multiple MDP states are aggregated based on their similarity [18] and temporally abstract options replace the MDP actions. For example, consider an agent which needs to navigate from room A to room B via a corridor. The low-level model may require many actions to avoid obstacles and negotiate doorways. An AMDP, by contrast, may represent the problem with two states representing the agent's presence in room A or B respectively and a single action representing the sequence of low-level actions necessary for navigation.

Single-Agent Reinforcement Learning (RL) is a technique that enables an agent to learn a deterministic policy $\pi : S \rightarrow A$ which maps each state to a permissible action when that state is encountered [8]. The learning process is inspired by behavioural psychology and uses past experiences to influence an agent's future behaviour. In this way, rewards are associated with actions executed in each state, and the agent gains these rewards as it explores the problem space.

The problem space is typically represented as a Markov Decision Process (MDP). While the agent navigates through the environment, it can select between using the action which is currently known to yield the highest reward (exploitation) or investigating other actions that may lead to higher rewards (exploration).

When an action is taken, a reward (or penalty) is received, and the agent updates the reward associated with the state action pair $Q : (s, a) \rightarrow \mathbb{R}$. Once the mapping of all state-action pairs is complete, a deterministic policy (π) may be extracted by selecting the action that returns the maximum reward for the state the agent currently resides within.

Multi-Agent Reinforcement Learning (MARL) is a natural extension to single-agent RL, where multiple agents learn how to work together towards a shared goal in order to improve efficiency and robustness through the division of labour. Categories of algorithms have formed commonly classified as independent learners, joint action learners, and direct policy search learners [10].

Independent learners learn entirely separately from other agents within the environment. These agents learn how to share a common workspace based on how their behaviours influence the environment. Although these techniques violate the Markov assumption, which underpins temporal difference techniques converging to an optimal policy, practically, these agents have been shown to converge [10].

Joint action learners allow agents to utilise game theory in their behaviour selection process. Agent behaviour will be directly influenced by the choices of other agents within the system and will learn which behaviour to pursue based on maximising the system's expected payoff. Thus, joint action learners are more sophisticated than independent learners. However, due to the nature of including all of the other agents' behaviour, these systems can easily suffer from state explosion [19].

Direct policy search learners allow exploration of a policy space through stochastic optimisation. This optimisation occurs through variable-based selection, describing all actions in a state as a probability distribution and adjusting these probabilities according to the agents' performance. Here the agent is searching the policy space directly without making use of action values.

These three groupings of MARL algorithms have often been combined, with algorithms using a mixture of temporal difference learning, game-theory, and direct policy search techniques, being commonly adapted [10].

Quantitative verification (QV) allows us to determine if quantitative properties, defined in formal logics, hold for a state transition model, as is the case with an MDP. For an MDP, these properties are expressed using probabilistic computation tree logic (PCTL) [20]. PCTL, a temporal logic, allows for the evaluation of probabilistic bounds on reachability within the model, e.g., the probability of reaching state s_f, where f is the end state, before time T. We may also associate reward structures with states and transitions in the model. In this way, we may evaluate bounds δ on cumulative rewards associated with actions taken within the problem space, e.g. battery usage $< \delta$ when the mission is complete. QV relies on efficient algorithms which examine the entire state-space of a given model. Probabilistic model checkers such as PRISM [21] and Storm [22] allow for such analysis.

Preliminary AMARL variant Our preliminary AMARL approach [14] considered safe multi-agent RL for simple MAS using two identical agents and solely Independent Q-learning reinforcement learning. Our initial approach consisted of a three-stage approach. In stage one, an AMDP and associated PCTL properties are derived, which represent the problem domain. In stage two, QV is employed to synthesise a safe policy that complies with the mandated safety requirements encoded in stage 1. Finally, stage three sees MARL applied to the system using the policies derived in stage two.

3. Related Work

Safe multi-agent reinforcement learning is a relatively new field of research, but it has gained much attention in recent years. Two main areas of research have emerged, deep safe MARL [23, 24], and safe MARL [25], which carries forward from more traditional reinforcement learning research. Our work falls into the latter category. However, our approach could be extended into deep safe MARL as future work.

Current work within what we will refer to as 'traditional' MARL has seen attempts to produce safe MARL policies through reward function tailoring [26] and multi-level reward function tailoring [27]. This work has seen interesting results but does not seem to consider safety during the learning process and is solely concerned about producing a

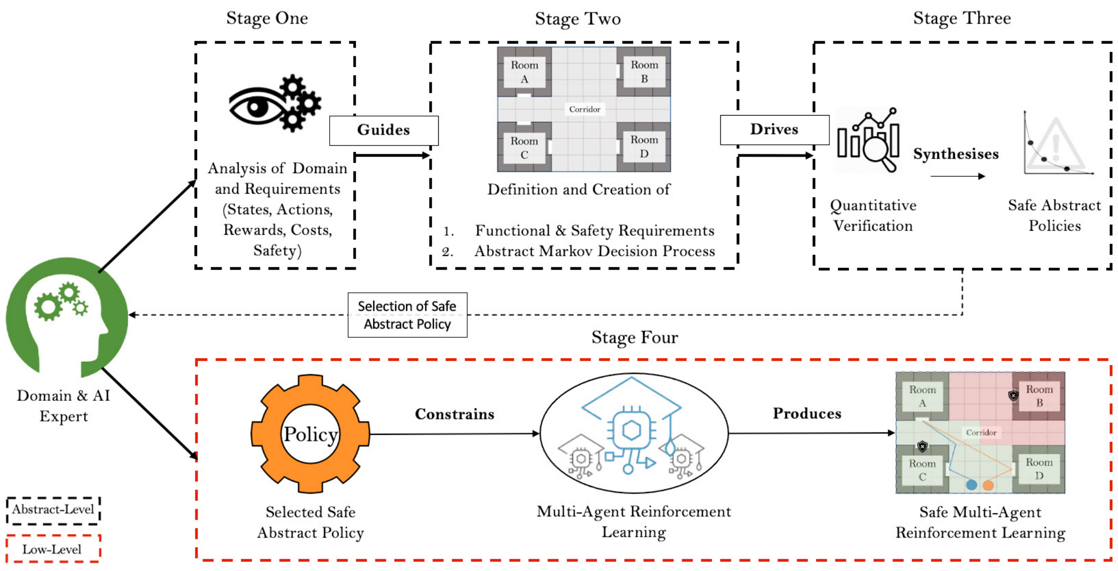

Stage One
Stage Two
Stage Three
Guides
Drives
Analysis of Domain and Requirements (States, Actions, Rewards, Costs, Safety)
Room A
Room B
Room C
Room D
Corridor
Definition and Creation of
1. Functional & Safety Requirements
2. Abstract Markov Decision Process
Synthesises
Quantitative Verification
Safe Abstract Policies
Selection of Safe Abstract Policy
Stage Four
Domain & AI Expert
Policy
Constrains
Produces
Selected Safe Abstract Policy
Multi-Agent Reinforcement Learning
Safe Multi-Agent Reinforcement Learning
Abstract-Level
Low-Level

Fig. 1. Assured Multi-Agent Reinforcement Learning

safe policy that can be used after learning. As a result, agents using this technique will repeatedly put themselves in a position of risk while learning what behaviours to avoid for their final policy.

These approaches also struggle to supply a formal guarantee of safety. The lack of formal guarantees is due to the reward function tailoring process, which is hard to tune appropriately [28]. There have also been attempts to reduce the risk of agent collision within safe MARL [29] and other particular safety approaches which may not apply to all problems, such as humans interfering to reduce unsafe behaviour [30]. These methods consider safety during and after the learning process but do not look at the more general MARL problem and only focus on specific issues.

The research that most closely relates to our work is the use of quantitative analysis to constrain agent behaviour. This technique has been used successfully in multi-agent systems [31, 32] and single-agent reinforcement learning [13], but was not applied to safe MARL until our recent work [14].

This agent behaviour constraint technique is well documented in safe RL [11], but this can be overly constraining, unnecessarily reducing the potential optimality of agents or removing behaviours necessary for mission completion. In [13] a technique that avoided this issue was proposed while still producing formal guarantees that agent learning would be deemed an acceptable level of safety.

While this approach was not the first instance of model checking being used within reinforcement learning [33], it is the first to do so in this manner and as a general-purpose formal approach.

4. Assured Multi-Agent Reinforcement Learning

As shown in Figure 1, our assured multi-agent reinforcement learning approach comprises four stages. These stages are presented in detail below.

Stage One involves the acquisition and analysis of information concerning the environment, the capabilities of the agents within the environment, the constraints on performance and the safety concerns to be addressed by the AMARL process. Since stage two requires us to construct an AMDP, the data-gathering exercise requires the identification of:

- all potential states for agents within the environment S. e.g. agent locations, failure modes, all possible transitions between states and more.
- the capabilities of agents and the states within which capabilities may be exercised A. These will be mapped to actions in the next stage.

- reward structures necessary for the evaluation of system performance and safety R. For example, how much energy is likely to be consumed by an agent undertaking action a in state s.
- goals and constraints defined as time, or probability, bounds on reaching states within the model, or limits on cumulative rewards. Goals and constraints derived in this way are amenable to encoding as PCTL.

It is important at this stage to abstract away all unnecessary detail for the problem to avoid the state space explosion problem. For example, consider two rooms linked by two separate corridors. If taking corridor A leads to a significant increase in the amount of energy required to transition between locations then this should be modelled. If no such variation exists, then the model may model both routes as a single transition.

We also note that the agents' capabilities, probabilities, and rewards may vary by individual agents in a heterogeneous system. For example, whilst agent one may be able to transition from room A to room B, agent two may find the route impossible, and while agent three is capable of moving from room A to B, the cost of doing so is higher than that of agent one. Such differences should be identified in this stage of the approach.

Stage Two makes use of the information derived in stage one to construct a model of the AMDP using the PRISM modelling language [21]. A command in PRISM takes the form:

```
[action] guard -> prob_1 : update_1 + ... + prob_n : update_n;
```

where action may be used as an annotation or, where another command has a matching value, for synchronisation, guard. is a predicate over all variables in the model and update describes a transition the model may take, with probability prob, if the guard evaluates to true.

In the following code fragment we show how two agents, which move between rooms in a navigation task, may be modelled.

```
// In room Zero and making a movement choice
[visit0_1_1]  !done & r1=0 & visits1<N -> 1:(r1'=1)&(visits1'=visits1+1); \\robot 1 visits room 1
[visit0_1_2]  !done & r2=0 & visits1<N -> 1:(r2'=1)&(visits1'=visits1+1); \\robot 2 visits room 1
```

In this case, for each command line, the action serves as an annotation to aid readability as well as a transition label which allows reward structures to be associated with the state transition. While done is a model variable used to indicate whether the functional goals of the MAS have been completed. Furthermore, r1 and r2 are variables that indicate the room in which robot 1 and 2, respectively, are in; visits1 is a counter for the number of times room 1 has been visited and; N is the maximum number of times we wish a room to be visited.

Reward structures in PRISM may be associated with states or transitions. The following reward structure fragment shows how the energy consumed by agent 1 is allocated as it moves between states:

```
rewards "energy"
[visit0_1_1]  true : 3;
[visit0_2_1]  true : 1.5;
[visit0_3_1]  true : 2;
```

Stage Three uses the abstract model created in the previous stage. As defined in stage one, the domain expert must also supply functional and safety constraints for analysis in the QV tool. These requirements and constraints may be expressed using PCTL.

Below we show two examples of PCTL used as possible safety constraints:

1. $(R\{\text{“}energy\text{”}\}_{\geq 0.3}$ [F finished])
2. $R\{\text{“}damage\text{”}\}_{\leq 0.01}$ [F finished]

Property 1 addresses the problem of excessive energy consumption. The property is read as the reward associated with energy should be greater than or equal to 0.3 when the system reaches the finished state. Similarly, property 2 represents a constraint on the amount of damage an agent may sustain before completing all tasks. We can constrain the possibility of damage to agents by stating that the risk of damage does not exceed the domain-specific requirement by the end of the process.

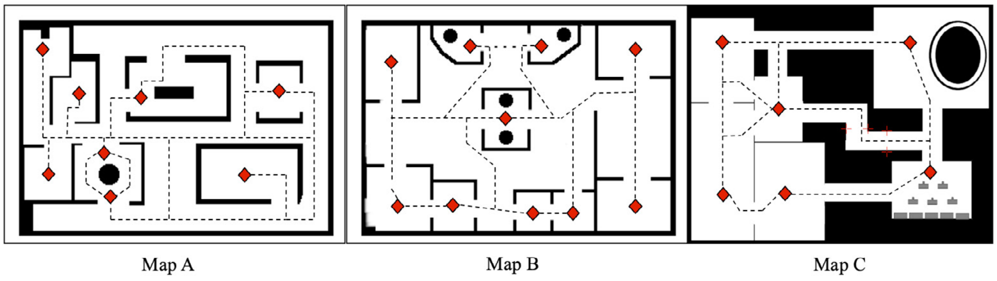


Fig. 2. Navigation Environment Maps with Patrolling Points Shown as red diamonds.

With both the AMDP and the functional and safety constraints defined, we may now use the QV tool to synthesise policies that meet the defined PCTL requirements. These policies are guaranteed to meet functional and safety requirements, and indeed, these synthesised policies are where the assurance of our approach is supplied. However, where constraints are overly constricting, it may be necessary to return to stage one to review the level of abstraction used and modify the model and constraints appropriately.

Once this stage is complete, we obtain a set of safe abstract policies.

Stage Four is the final stage and involves learning policies within the non-abstracted domain while under the constraint of the policy retrieved from stage three. The synthesised policy is used to partition and constraint agents' tasks and also the domain space. These constraints allow the robots to explore and learn about the problem domain without violating the constraints of the guaranteed policy.

While under these constraints, an action may be taken that holds a quantified level of risk, but with the QV verified policy, we can guarantee that the cumulative risk and the probability of risky events happening are bounded. This approach does not aim for optimality, as the most optimal approach may be disallowed during the QV process. However, it guarantees a level of safety and quality while typically increasing the learning process's speed.

5. Learning Domains

To evaluate AMARL, we consider a problem in which a team of robots must inspect a nuclear power plant where each location in the plant must be inspected multiple times. However, several regions within the plant are highly irradiated and pose a risk to the agents should they spend too long in these regions. In addition, the battery life of the agents is limited and, should an agent run out of energy, retrieving the agent would pose a potential risk to human operators. Using agents to inspect nuclear power plants has been suggested as a potential application of MAS [34, 6] and patrolling problems more generally are prevalent in MARL research [3]. Figure 2 illustrates three different maps (domains) in which black circles indicate areas of risk while red diamonds indicate key areas that the robots must visit (patrol points). Irradiated areas are the primary safety concern within these domains. Radiation exposure is hazardous to humans and robots alike. Due to this reason, robotic agents must limit their time within these irritated areas by making intelligent navigation choices to reduce exposure; these choices are displayed in Figure 3.

We set a system-level goal for all domains that each patrol point should visit three times without the batteries being exhausted or the agents becoming damaged due to excessive exposure to radiation in high-risk areas.

- C1: Visit each room a minimum of three times (Encoded as an abstract state within the AMDP).
- C2: Complete all tasks while attempting to minimise battery usage (PCTL Property : R"energy"min=?).
- C3: Minimise time in irradiated areas (PCTL Property :R"risk"min=?).

For each map, we then define the permissible level of risk of exposure and minimum safe battery levels as follows:

- Map A: maximum acceptable risk - 0.2; minimum battery level - 20.
- Map B: maximum acceptable risk - 0.25; minimum battery level - 25.
- Map C: maximum acceptable risk - 0.1; minimum battery level - 30.

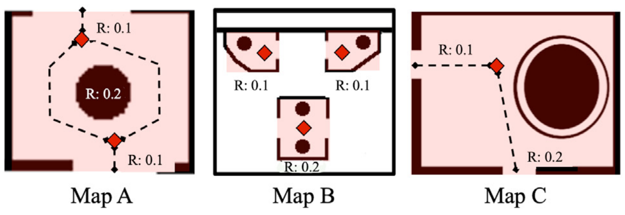


Fig. 3. Areas of risk within maps

However, despite having a secondary safety constraint for battery usage, due to the binary nature of the domain problems, there must be leniency during the early learning episodes to allow the systems time to learn a successful policy that does not run until complete battery depletion. During later stages of the learning process, this leniency is removed entirely.

6. Evaluation

6.1. Experimental Setup

We demonstrated our approach using an openly available ROS simulator [35] within which we implemented the three domains shown in Figure 2. Reinforcement learning is applied to these patrolling problems by giving each patrol point four potential states in which the agent can make choices. These states are based on how often these patrol points have been visited. State 0 - Not Visited; State 1 - Visited Once; State 2 - Visited Twice; State 3 - Visited Three Times.

A series of experiments were then undertaken in which the MARL Algorithm, number and type of agents were varied.

MARL Algorithms: Three separate algorithms were considered: Independent Q-learning (Temporal Difference Learning) [15], Team-Q (Game Theory) [16], WoLF-PHC (Direct Policy Search) [17]. These algorithms fall into the main three categories of MARL techniques, showing AMARL's plug-in styled flexibility.

Agents: The number of agents used within the environment was set to either two agents or three agents. For the two agent scenario, the agents were homogeneous. However, when three agents were employed, we also considered heterogeneous agents. Specifically, we introduced an agent with additional radiation shielding. We name this robot the Scutum (shielded) robot, which consumes more energy than the standard 'Runner' agent seen in the homogenous systems, but the Scutum agent accumulates risk more slowly in highly radiated areas of the map.

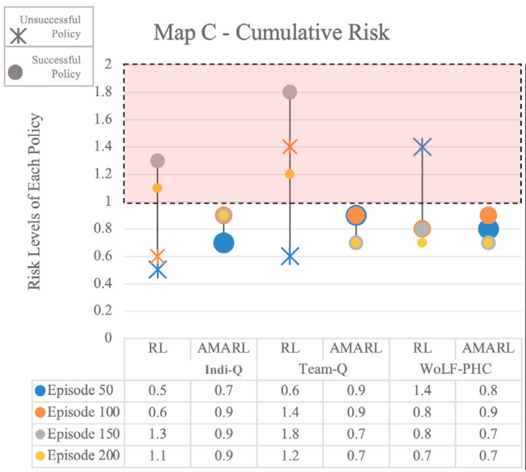
For each scenario, we compare AMARL to a baseline using standard MARL, listed as simply RL in graphs and tables to allow a more straightforward distinction between the two.

6.2. Results

We begin our experimentation by considering Map C. All three algorithms were evaluated using standard RL and our AMARL approach, and the results are shown in Figures 4 and 5. We note that the standard RL is often unable to find a successful policy, and when one is found, the risk is in the red zone, i.e. it violates our safety limits. Our AMARL approach, in contrast, finds a successful policy for all three learning algorithms, which respects the safety constraint imposed for risk of damage to the agents.

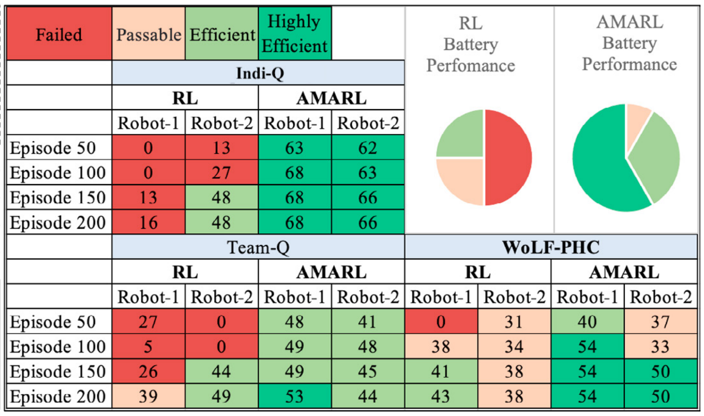
When we consider battery performance, shown in Figure 5, we see that RL takes significantly longer to find a successful policy than AMARL. This is due to AMARL's constrained search space, allowing more efficient choices to be made. Looking at the overall performance of the battery conservation, which can be seen displayed in the pie charts within Figure 5, AMARL's battery conservation outperforms RL. However, it can be seen that in some cases, RL performance better than AMARL. This can be viewed as AMARL finding a trade-off between battery conservation performance and risking damage to the agents.

All three algorithms show a similar trend, with Team-Q and WoLF-PHC showing more concern for safety than Independent-Q due to their ability to make more informed decisions. However, this did allow Independent-Q improved battery conservation. Overall, AMARL is shown to incorporate these different algorithms successfully.



	RL	AMARL	RL	AMARL	RL	AMARL
	Indi-Q		Team-Q		WoLF-PHC	
● Episode 50	0.5	0.7	0.6	0.9	1.4	0.8
● Episode 100	0.6	0.9	1.4	0.9	0.8	0.9
● Episode 150	1.3	0.9	1.8	0.7	0.8	0.7
● Episode 200	1.1	0.9	1.2	0.7	0.7	0.7

Fig. 4. Cumulative Risk of Policies in Map C.

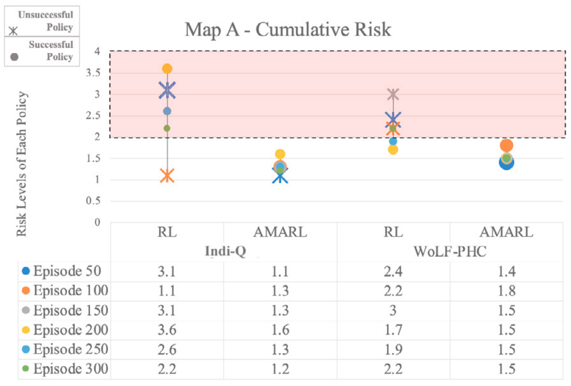

RL Battery Perfomance
AMARL Battery Performance

Failed	Passable	Efficient	Highly Efficient

	Indi-Q			
	RL		AMARL	
	Robot-1	Robot-2	Robot-1	Robot-2
Episode 50	0	13	63	62
Episode 100	0	27	68	63
Episode 150	13	48	68	66
Episode 200	16	48	68	66

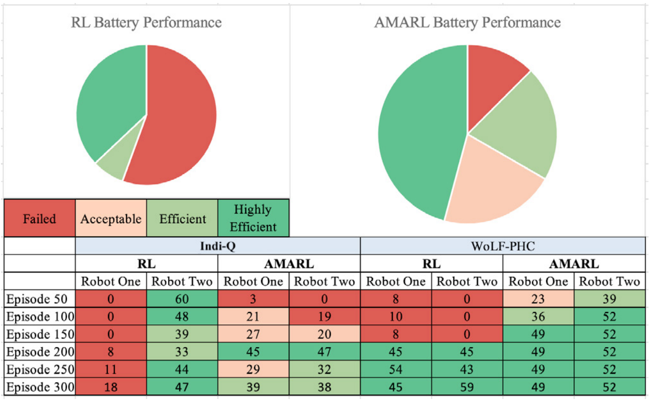
	Team-Q				WoLF-PHC			
	RL		AMARL		RL		AMARL	
	Robot-1	Robot-2	Robot-1	Robot-2	Robot-1	Robot-2	Robot-1	Robot-2
Episode 50	27	0	48	41	0	31	40	37
Episode 100	5	0	49	48	38	34	54	33
Episode 150	26	44	49	45	41	38	54	50
Episode 200	39	49	53	44	43	38	54	50

Fig. 5. Battery Conservation of Policies in Map C.



	RL	AMARL	RL	AMARL
	Indi-Q		WoLF-PHC	
● Episode 50	3.1	1.1	2.4	1.4
● Episode 100	1.1	1.3	2.2	1.8
● Episode 150	3.1	1.3	3	1.5
● Episode 200	3.6	1.6	1.7	1.5
● Episode 250	2.6	1.3	1.9	1.5
● Episode 300	2.2	1.2	2.2	1.5

Fig. 6. Cumulative Risk of Policies in Map A.


RL Battery Performance
AMARL Battery Performance

Failed	Acceptable	Efficient	Highly Efficient

	Indi-Q				WoLF-PHC			
	RL		AMARL		RL		AMARL	
	Robot One	Robot Two	Robot One	Robot Two	Robot One	Robot Two	Robot One	Robot Two
Episode 50	0	60	3	0	8	0	23	39
Episode 100	0	48	21	19	10	0	36	52
Episode 150	0	39	27	20	8	0	49	52
Episode 200	8	33	45	47	45	45	49	52
Episode 250	11	44	29	32	54	43	49	52
Episode 300	18	47	39	38	45	59	49	52

Fig. 7. Battery Conservation of Policies in Map A.

For Map A, which is slightly larger, the two best-performing algorithms in terms of relative safety and battery conservation are allowed to learn, again, both using RL and AMARL. This shows that the trends found using Map C are still present, with AMARL always remaining within the restrictions of its primary safety objective and RL continuing to place itself within unnecessary risky areas as seen in Figure 6. In terms of the behaviours of the algorithms, they are as predicted based on previous studies that utilise these techniques, with WoLF-PHC finding more optimal policies at a faster rate due to its direct search method than independent-Q learning.

The battery conservation results in Figure 7 show for the first time AMARL not succeeding to find a policy that meets the battery conservation requirements before episode fifty. This delay is to be expected with the binary problem given to the MARL agents; either the mission is successful, or the mission is failed, and the robots deplete their batteries, there is no grace period between these outcomes. This binary problem is why battery conservation limits are not enforced rigidly within the early stages of the learning process due to the nature of MARL and become crucial at the later stages of learning and after the learning has concluded.

With this in mind, AMARL met both the functional requirements of these domains efficiently in comparison to RL, and unlike RL, AMARL met its primary safety constraint consistently while following the less rigid guidelines for safety in regards to battery conservation reliably.

Finally, we considered the use of heterogeneous agents within Map B, the largest of the domains used in this study. The AMDP involved in these experiments was adjusted by tailoring reward structures for the specific costs of the different agents performing actions. Such as the Runner robots using less battery when making transitions between

	Indi-Q	
	RL	AMARL
Episode 50	3.2	1.2
Episode 100	3.7	1.2
Episode 150	2.6	1.3
Episode 200	3.2	1.4

Map B – Cumulative Risk

Failed	Acceptable	Efficient	Highly Efficient

	Indi-Q					
	RL			AMARL		
	Runner One	Runner Two	Scutum One	Runner One	Runner Two	Scutum One
Episode 50	24	20	0	40	58	56
Episode 100	33	44	0	42	57	60
Episode 150	52	49	44	56	67	64
Episode 200	63	66	50	60	64	67

Map B – Battery Conservation

Fig. 8. Cumulative Risk and Battery of Policies in Map B.

states but accumulating more risk from hazardous areas, and the Scutum robot using more battery when making transitions, but significantly reducing the risk that it accumulates.

The policy synthesis allowed the selection of three safe abstract policies, and the policy which prioritised limited risk was selected for use. This can be seen in Figure 8, where the runner robots do not enter risky areas and focus on efficiently completing safe tasks, and the Scutum robot focuses primarily on the risky areas due to it being much more equipped to deal with this radiation exposure.

As can be seen in Figure 8, the battery was balanced relatively well given the differing consumption rates, and the system under the guidance of AMARL met all functional and safety requirements.

7. Conclusion

As seen by the evaluation of AMARL, this approach successfully allows MARL to be utilised within safety-critical scenarios both during and after the learning process reliably. Furthermore, the contributions mentioned in the introduction of this research have also been successfully delivered through the experimental evaluation of AMARL in varied scenarios. Namely, the use of homogeneous and heterogeneous systems, varied system sizes, various MARL algorithms, and multiple problem domains demonstrates AMARL's flexibility.

Clear trends have formed that support the continued use and development of AMARL, such as consistently meeting primary safety requirements during and after learning, efficiently balancing conflicting objectives, and increasing the learning speed of systems with a constrained search space. AMARL outperformed the standard RL systems regarding all of these trends mentioned above and demonstrated a level of trust that can be placed in successfully implementing AMARL.

Our approaches potential future work could focus on increased robustness to the model's abstract process and correctness checking by evaluating states and transitions during runtime and adjusting the abstraction accordingly. This work would be very valuable due to the difficulty that is involved with accurately abstracting a problem. There could also be an extension from this work focusing on showing Deep MARL techniques utilised with AMARL. While there is no sign that Deep MARL would not apply to AMARL, there would be increased confidence in the approach with this further work.

References

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[2] M. Frasheri, B. Cürüklü, M. Esktröm, and A. V. Papadopoulos, "Adaptive autonomy in a search and rescue scenario," in *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 150–155, IEEE, 2018.

[3] M. Othmani-Guibourg, A. El Fallah-Seghrouchni, J.-L. Farges, and M. Potop-Butucaru, "Multi-agent patrolling in dynamic environments," in *2017 IEEE International Conference on Agents (ICA)*, pp. 72–77, IEEE, 2017.

[4] C. Hurtado, M. R. Ramirez, A. Alanis, S. O. Vazquez, B. Ramirez, and E. Manrique, "Towards a multi-agent system for an informative healthcare mobile application," in *KES International Symposium on Agent and Multi-agent Systems: Technologies and Applications*, pp. 215–219, Springer, 2018.

[5] K. Houliotis, P. Oikonomidis, P. Charchalakis, and E. Stipidis, "Mission-critical systems design framework," *Advances in Science, Technology and Engineering Systems Journal*, vol. 3, no. 2, pp. 128–137, 2018.

[6] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Robotics research*, pp. 455–472, Springer, 2017.

[7] H. A. Abbas, S. I. Shaheen, and M. H. Amin, "Organization of multi-agent systems: an overview," *arXiv preprint arXiv:1506.09032*, 2015.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] P. G. Patel, N. Carver, and S. Rahimi, "Tuning computer gaming agents using q-learning," in *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 581–588, IEEE, 2011.

[10] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in Multi-agent Systems and Applications-1*, pp. 183–221, 2010.

[11] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[12] G. Mason, R. Calinescu, D. Kudenko, and A. Banks, "Assured reinforcement learning with formally verified abstract policies," in *9th International Conference on Agents and Artificial Intelligence (ICAART)*, pp. 105–117, 2017.

[13] G. Mason, R. Calinescu, D. Kudenko, and A. Banks, "Assurance in reinforcement learning using quantitative verification," in *Advances in Hybridization of Intelligent Methods*, pp. 71–96, Springer, 2018.

[14] J. Riley, R. Calinescu, C. Paterson, D. Kudenko, and A. Banks, "Reinforcement learning with quantitative verification for assured multi-agent policies," in *13th International Conference on Agents and Artificial Intelligence*, York, 2021.

[15] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[16] B. Ghazanfari and N. Mozayani, "Enhancing Nash Q-learning and Team Q-learning mechanisms by using bottlenecks," *Journal of Intelligent & Fuzzy Systems*, vol. 26, no. 6, pp. 2771–2783, 2014.

[17] W. Zhao, Z. Fang, and Z. Yang, "Four-dimensional trajectory generation for UAVs based on multi-agent Q learning," *The Journal of Navigation*, vol. 73, no. 4, pp. 874–891, 2020.

[18] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," *ISAIM*, vol. 4, p. 5, 2006.

[19] S. S. Mousavi, B. Ghazanfari, N. Mozayani, and M. R. Jahed-Motlagh, "Automatic abstraction controller in reinforcement learning agent via automata," *Applied Soft Computing*, vol. 25, pp. 118–128, 2014.

[20] F. Ciesinski and M. Größer, "On probabilistic computation tree logic," in *Validation of Stochastic Systems*, pp. 147–188, 2004.

[21] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Int. Conf. Computer Aided Verification*, pp. 585–591, 2011.

[22] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A Storm is coming: A modern probabilistic model checker," in *Int. Conf. Computer Aided Verification*, pp. 592–600, 2017.

[23] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[24] S. Lu, K. Zhang, T. Chen, T. Basar, and L. Horesh, "Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning," 2021.

[25] N. Jansen, B. Könighofer, S. Junges, and R. Bloem, "Shielded decision-making in MDPs," *arXiv preprint arXiv:1807.06096*, 2018.

[26] Y. Hu, Y. Gao, and B. An, "Multiagent reinforcement learning with unshared value functions," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 647–662, 2014.

[27] F. Ritz, T. Phan, R. Müller, T. Gabor, A. Sedlmeier, M. Zeller, J. Wieghardt, R. Schmid, H. Sauer, C. Klein, *et al.*, "SAT-MARL: Specification aware training in multi-agent reinforcement learning," *arXiv preprint arXiv:2012.07949*, 2020.

[28] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "LTL and beyond: Formal languages for reward function specification in reinforcement learning.," in *IJCAI*, vol. 19, pp. 6065–6073, 2019.

[29] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," *arXiv preprint arXiv:2101.05436*, 2021.

[30] E. M. E. Mhamdi, R. Guerraoui, H. Hendrikx, and A. Maurer, "Dynamic safe interruptibility for decentralized multi-agent reinforcement learning," *arXiv preprint arXiv:1704.02882*, 2017.

[31] B. Herd, S. Miles, P. McBurney, and M. Luck, "Quantitative analysis of multi-agent systems through statistical verification of simulation traces," *International Journal of Agent-Oriented Software Engineering*, vol. 6, no. 2, pp. 156–186, 2018.

[32] A. Tarasyuk, I. Pereverzeva, E. Troubitsyna, and L. Laibinis, "Formal development and quantitative assessment of a resilient multi-robotic system," in *International Workshop on Software Engineering for Resilient Systems*, pp. 109–124, Springer, 2013.

[33] S. Junges, N. Jansen, J.-P. Katoen, U. Topcu, R. Zhang, and M. Hayhoe, "Model checking for safe navigation among humans," in *International Conference on Quantitative Evaluation of Systems*, pp. 207–222, Springer, 2018.

[34] R. Bogue, "Robots in the nuclear industry: a review of technologies and applications," *Industrial Robot: An International Journal*, 2011.

[35] D. Portugal, L. Iocchi, and A. Farinelli, "A ROS-based framework for simulation and benchmarking of multi-robot patrolling algorithms," in *Robot Operating System (ROS)*, pp. 3–28, 2019.