Generating Local Shields for Decentralised Partially Observable Markov Decision Processes

Haoran Yang
University of Oxford
haoran.yang@balliol.ox.ac.uk

Nobuko Yoshida
University of Oxford
nobuko.yoshida@cs.ox.ac.uk

Multi-agent systems under partial observation often struggle to maintain safety because each agent's locally chosen action does not, in general, determine the resulting joint action. Shielding addresses this by filtering actions based on the current state, but most existing techniques either assume access to a shared centralised global state or employ memoryless local filters that cannot consider interaction history.

We introduce a shield process algebra with guarded choice and recursion for specifying safe global behaviour in communication-free Decentralised Partially Observable Markov Decision Process (Dec-POMDP) settings. From a shield process, we compile a process automaton, then a global Mealy machine as a safe joint-action filter, and finally project it to local Mealy machines whose states are belief-style subsets of the global Mealy machine states consistent with each agent's observations, and which output per-agent safe action sets.

We implement the pipeline in Rust[1] and integrate PRISM, the Probabilistic Symbolic Model Checker[2], to compute best- and worst-case safety probabilities independently of the agents' policies. A multi-agent path-finding case study demonstrates how different shield processes substantially reduce collisions compared to the unshielded baseline while exhibiting varying levels of expressiveness and conservatism.

Keywords: Multi-agent systems, Dec-POMDP, Shield process algebra, Mealy machine, Deadlock-freedom, Collision-freedom, PRISM

1 Introduction

Multi-agent systems are prone to encountering collisions, deadlocks, or livelocks because an individual agent's local action does not, in general, uniquely determine the resulting joint action. Consequently, ensuring that each agent selects a concrete action that is guaranteed to be safe is non-trivial. This challenge is exacerbated in communication-free, Decentralised Partially Observable Markov Decision Process (Dec-POMDP) settings [10], where each agent must choose its action solely on the basis of its own local observations.

One widely used approach to enforcing safety is shielding, in which a shield restricts the agents' choices by providing, at each state, a set of safe actions. Existing approaches include temporal-logic based synthesis [1], probabilistic shields [4], learned shields [8], and decentralisation techniques [3, 6, 7] that derive local shields from a centralised safe-action set. While these methods have proved effective in many settings, they typically rely on assumptions such as access to a centralised global state, the availability of memoryless local filtering, or the sufficiency of constraints expressed purely at the action level. These assumptions limit their applicability in regimes where agents possess only local information and inter-agent communication is disallowed.

[1] https://gitlab.cs.ox.ac.uk/ug23hy/shield-process-compilation-pipeline/
[2] https://www.prismmodelchecker.org

Lorenzo Gheri & Kirstin Peters (Eds.): 17th International Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software 2026 (PLACES 2026)
EPTCS 444, 2026, pp. 1–10, doi:10.4204/EPTCS.444.1

Although shields can sometimes be generated automatically, for example, via DFA synthesis from LTL specifications [1], such constructions tend to be rigid and conservative. Moreover, they may fail to capture the informational complexity induced by partial observations, wherein each agent's local view reveals only a fragment of the underlying global state.

To solve these problems, we propose a succinct process algebra with guarded choice and recursion. This algebra admits an automata-theoretic interpretation and can be compiled into Mealy-machine-style shields [5]. The compilation pipeline comprises three main steps. First, we translate the process algebra into a process automaton. Second, we construct a global shield Mealy machine that takes the Dec-POMDP state as input and outputs a decentralised representation of safe joint actions consistent with the process automaton. Third, we derive local shield Mealy machines by replacing the Dec-POMDP state input with each agent's partial observation: the local shield Mealy machines' states are belief-style subsets of global Mealy machine states consistent with the agent's observation, thereby enumerating the possible individual safe action sets that remain feasible under decentralised local information. We implement this pipeline in Rust and analyse the resulting models using an integrated PRISM-based workflow.

In this paper, we present the complete pipeline, from motivation to implementation details, and illustrate it through a simplified multi-agent path-finding (MAPF) [11] case study. MAPF is used here as a case study because it is easy to visualise and has simple collision rules; however, the pipeline applies more generally to communication-free Dec-POMDPs whenever the relevant safety condition can be represented using the state together with a finite counter.

Contributions (1) A shield process algebra with recursion and guarded choice for decentralised, partially observable, communication-free settings. (2) A compilation pipeline that, given a shield process specification, generates the corresponding shield process automaton and, using transition and observation descriptions, synthesises a global Mealy shield and projects it to local Mealy shields executable from local observations, which in turn export safe action sets for each agent under partial observability. (3) A Rust implementation of this pipeline with PRISM integration, enabling the computation of best- and worst-case safety probabilities without fixing a policy.

2 Pipeline Explanation

This section is structured as follows. First, we introduce the preliminaries that underpin our pipeline. Second, we specify the configuration of a simplified multi-agent path-finding (MAPF) problem, which serves as a running example to illustrate the pipeline. Third, we present the process algebra used to describe a shield in terms of global states, thereby providing a global perspective on system behaviour. Fourth, we describe the pipeline itself using a special case of the MAPF problem with "blind" agents. In this final part, we construct a process automaton directly from the process algebra, derive a global shield Mealy machine from the process automaton together with the descriptions of initial states and the joint actions that induce state transitions, and subsequently synthesise a local shield Mealy machine for each agent by applying partial observation and belief-state construction to relate local observations to global states.

2.1 Preliminaries

A *decentralised partially observable Markov decision process* (*Dec-POMDP*) is a tuple [10] $\mathscr{M} = (\mathscr{I}, S, (A_i)_{i\in\mathscr{I}}, \delta, (\Omega_i)_{i\in\mathscr{I}}, O, r, \gamma, \rho_0)$, where $\mathscr{I} = \{1,\ldots,n\}$ is the set of agents, S is the global state

space, and A_i is the action space of agent i (with joint action space $A := \prod_{i \in \mathscr{I}} A_i$). The transition dynamics are given by $\delta(s' \mid s, a)$, the probability of transitioning from state s to state s' under joint action a. For each agent i, Ω_i is the observation space (with joint observation space $\Omega := \prod_{i \in \mathscr{I}} \Omega_i$), and $O(o_1, \ldots, o_n \mid s', a)$ denotes the probability of joint observation $(o_1, \ldots, o_n)$ after executing a and reaching s'. The shared reward function is $r(s,a)$, $\gamma \in (0,1]$ is the discount factor, and ρ_0 is the initial-state distribution.

A *global history* up to time t is $h^t = (s^0, a^0, \ldots, a^{t-1}, s^t)$, and a *local history* for agent i up to time t is $h_i^t = (o_i^0, a_i^0, \ldots, a_i^{t-1}, o_i^t)$. A *decentralised stochastic policy* is a tuple $\pi := (\pi_1, \ldots, \pi_n)$, where each $\pi_i(a_i \mid I_i^t)$ is a distribution over A_i given the local information I_i^t, typically the local history h_i^t or, in the memoryless case, the current observation o_i^t [7]. Bounded histories can be encoded into states or observations by forming the product of states or observations with actions. Consequently, in the remainder of the paper, we focus on states and observations without loss of generality.

A *shield* for an agent is a function $\mathscr{D} : S \to 2^A$ that returns the set of safe actions at each state [1, 8, 7]. Shields can be represented, for example, as deterministic finite automata (DFAs) or as Mealy machines [3]. In this work, we consider *pre-posed shields* that filter the available actions before the agents select them.

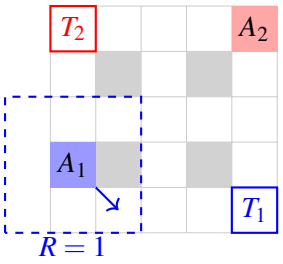
A *Mealy machine* is a tuple $(Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ_I is the input alphabet, Σ_O is the output alphabet, $\delta : Q \times \Sigma_I \to Q$ is the transition function, and $\lambda : Q \times \Sigma_I \to \Sigma_O$ is the output function. In this paper, Mealy machines represent shields whose inputs are either global states or local observations, and whose outputs are safe action sets.

2.2 A (Simplified) Multi-Agent Path-Finding Problem

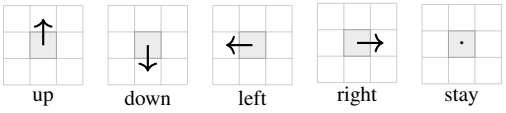
We consider a simplified multi-agent path-finding (MAPF) problem [11]. The environment is a fixed $H \times W$ grid populated by n agents. Each grid cell is either an obstacle or a free cell. A free cell may be empty, occupied by an agent, or designated as an agent's target.

Each agent can perform one of five actions: $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \cdot\}$, which correspond to moving left, right, up, down, or remaining stationary. Agents operate under partial observability; each agent's observation Ω_i includes its local neighbourhood within radius R and a coarse direction-to-target signal in $\{-1,0,1\} \times \{-1,0,1\}$.

For simplicity, we consider only *vertex conflicts*, meaning that no two agents may occupy the same cell after a transition. Figure 1(a) shows a representative grid configuration with two agents, their targets, obstacles, and an example observation radius and direction-to-target-signal. Figure 1(b) depicts the action space for each agent, and Figure 1(c) illustrates a vertex conflict.



(a) Grid with agents (filled cells), obstacles (grey cells), targets (bordered cells), and the current observation region of A_1 for $R = 1$.



(b) Agent action space.

(c) Vertex conflict example.

Figure 1: Example MAPF grids.

2.3 Shield Process Algebra

Shield Process We define a *global state shield* as a set of states $Sh \subseteq S$ that are considered safe to move next. We also define a *shield process* using the following grammar:

$$P ::= \mathsf{idle} \mid \mathsf{fail} \mid \mu X.P \mid X \mid Sh.P \mid P_1 \|_g P_2,$$

where idle denotes successful termination of the shield process, and fail denotes unsuccessful termination. The construction $\mu X.P$ introduces recursion (which is assumed to be guarded), with X as the corresponding recursion variable. The term $Sh.P$ denotes a continuation process that asserts that the next state lies in the global state shield $Sh \subseteq S$ and then continues as P. For a guarded choice $P_1 \|_g P_2$ with $g \subseteq S$, the process behaves as P_1 when the current state $s \in g$, and as P_2 otherwise.

2.4 Vertex Conflict Avoidance Example with the Pipeline

An interesting motivating example arises when the agents are blind ($R = 0$), a scenario we refer to as the "Blind Agents", in the environment shown in Figure 2.

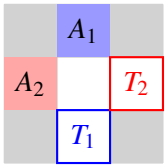


Figure 2: Initial state for the "Blind Agents" ($R = 0$).

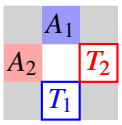
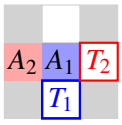
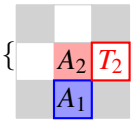
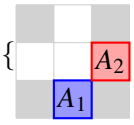
Since each agent's local observation remains constant over time, it is impossible to construct a shield that depends solely on local observations [7]. Likewise, DFA-based constructions that attempt to avoid unsafe states [3] are ineffective in this setting, because no action available to any agent can be guaranteed to lead, in combination, to a safe joint action.

By contrast, our method enriches the available information through the process syntax, hence it is able to address this problem using the pipeline as shown below:

Pipeline Initiation To instantiate the pipeline, we provide a transition description $SAS : S \times S \to 2^A$, an observation description $O'_i : S \to 2^{\Omega_i}$, and an initial set description $S_0 \subseteq S$. These descriptions may be instantiated as the supports of δ, O and ρ_0, or as conservative abstractions thereof.

Shield Process Description We begin with the following process specification:

$$P = (Sh_1.Sh_2.Sh_3.\mathsf{idle}) \|_g \mathsf{fail}$$

where $g = \{$  $\}$, $Sh_1 = \{$  $\}$, $Sh_2 = \{$  $\}$, $Sh_3 = \{$  $\}$
demonstrates a sequence of safe states that ensures both agents eventually reach their respective targets without encountering vertex conflicts.

Shield Process Automaton We generate a shield process automaton from the process specification, obtaining a deterministic automaton that interprets the shield process:

$$\mathscr{A}_{P_0} = (Q_P, \Sigma, \delta_P, q_0), \qquad \Sigma := S, \quad q_0 := \mathsf{start}$$

Its states are either idle, fail, a continuation process (e.g., $Sh_2.Sh_3.\mathsf{idle}$), or the dummy initial state start. Transitions follow the guard set g and consume the global state shield Sh in sequence. The resulting automaton for the "Blind Agents" is shown in Figure 3.

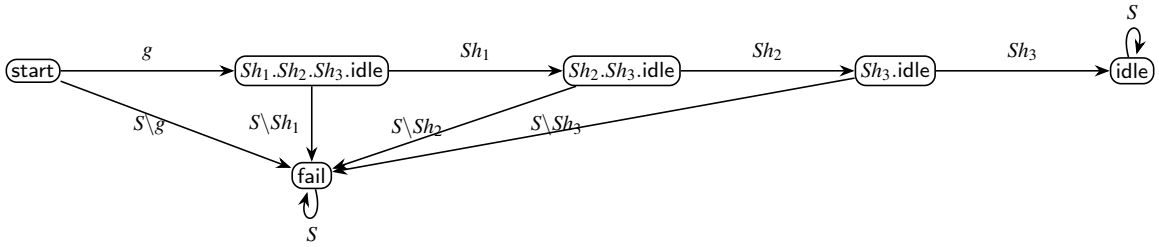


Figure 3: Shield process automaton for the "Blind Agents".

Global Shield Mealy Machine We construct the global shield Mealy machine

$$G = (Q_G, q_0^G, \Sigma_I^G, \Sigma_O^G, \delta_G, \lambda_G),$$

where $Q_G := \{\mathsf{idle}, \mathsf{fail}\} \cup (2^S \times (Q_P \backslash \{\mathsf{idle}, \mathsf{fail}\}))$, $q_0^G := (S_0, \mathsf{start})$, $\Sigma_I^G := S$, and $\Sigma_O^G := \left(\prod_{i=1}^n 2^{A_i}\right) \cup \{\bot\}$. Here, the distinguished output symbol $\bot$ denotes shield failure: no safe action set can be provided for the current situation. Each global Mealy state thus records the current set of reachable Dec-POMDP states together with the current state of the process automaton.

To compute outputs, we fix a deterministic decomposition $\mathsf{Dec} : 2^A \to \prod_{i=1}^n 2^{A_i}$ that turns a joint action set into one safe local action set per agent. For $\mathsf{Dec}(\hat{A}) = (U_1, \ldots, U_n)$, where $\hat{A} \subseteq A$ is a set of joint actions, we require $\prod_{i=1}^n U_i \subseteq \hat{A}$ and $\prod_{i=1}^n U_i = \emptyset \iff \hat{A} = \emptyset$. Thus, if each agent independently chooses an action from its local set, the resulting joint action remains in $\hat{A}$. In this paper, we instantiate Dec by selecting, in a deterministic manner, a maximum-cardinality product set $\prod_{i=1}^n U_i \subseteq \hat{A}$.

For a transition from (S_{cur}, q) to (S_{next}, q'), we compute $\hat{A}$ by intersecting, over all $s \in S_{\mathsf{cur}} \cap g$, with $q \xrightarrow{g} q'$, the joint actions that can move from s into any state in S_{next} using SAS, and output $\mathsf{Dec}(\hat{A})$.

For the "Blind Agents", the resulting global shield Mealy machine is shown in Figure 4.

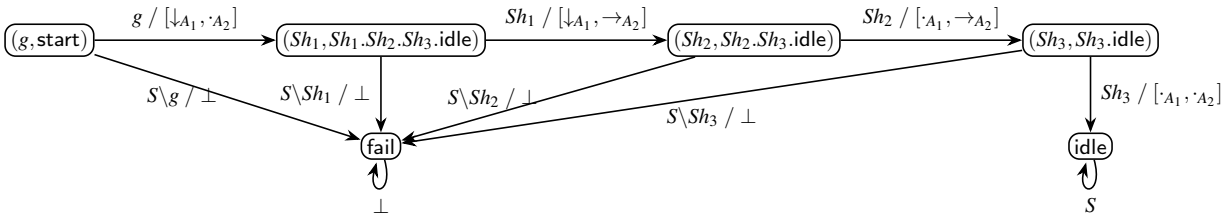


Figure 4: Global shield Mealy machine for the "Blind Agents".

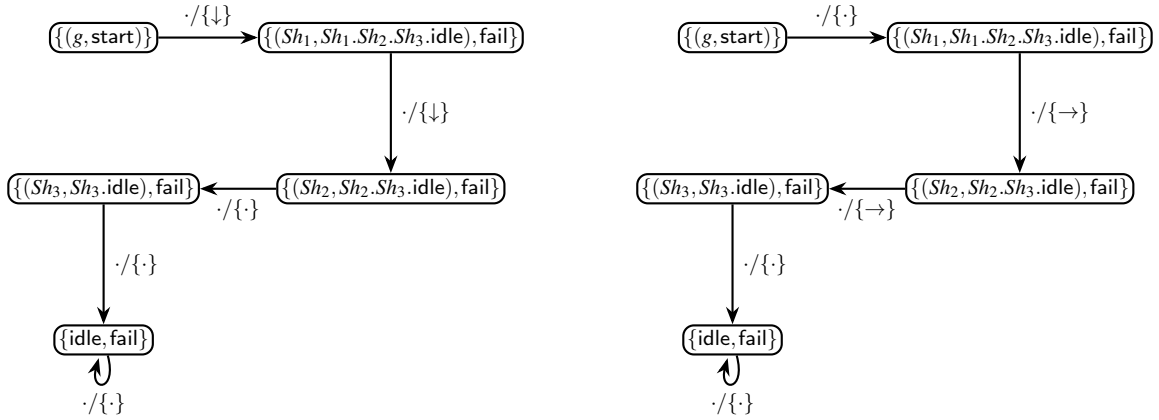
Local Shield Mealy Machine The local shield for agent i is a Mealy machine

$$L_i = (Q_{L_i}, q_0^{L_i}, \Sigma_I^{L_i}, \Sigma_O^{L_i}, \delta_{L_i}, \lambda_{L_i}),$$

where $Q_{L_i} := 2^{Q_G}$, $q_0^{L_i} := \{(S_0, \mathsf{start})\}$, $\Sigma_I^{L_i} := \Omega_i$, and $\Sigma_O^{L_i} := 2^{A_i} \cup \{\bot\}$. Each local Mealy state is a belief subset over global Mealy states consistent with agent i's current observation, as determined by the observation description O_i'. The transition function is defined by $\delta_{L_i}(q^{L_i}, o_i) := \bigcup_{(\hat{S},q)\in q^{L_i}} \{\, \delta_G((\hat{S},q),s) \mid s \in \hat{S},\ o_i \in O_i'(s) \,\}$, where $q^{L_i} \subseteq Q_G$ is the current local Mealy state and $o_i \in \Omega_i$ is the observation.

The local Mealy output is obtained by intersecting the ith components of the global Mealy outputs over the global Mealy states represented by the current local Mealy state. We treat fail as contributing all actions unless every state in the belief is fail, in which case the output is $\bot$, indicating a shield failure: the shield cannot provide any action set that is guaranteed to be safe for the agent to choose from.

In the "Blind Agents", all observations are identical, so the belief state remains maximal and all process branches are treated as possible. The resulting local shields are shown in Figure 5.



(a) Local shield for A_1.

(b) Local shield for A_2.

Figure 5: Local shield Mealy machines in the "Blind Agents".

PRISM analysis We analyse the shielded model using PRISM model checker, treating the agents' policies as nondeterministic. PRISM computes lower and upper bounds on both the probability of shield failure and the probability of reaching unsafe states; in this instance, shield failure has lower and upper bounds of 0.000000, and reaching unsafe states has lower and upper bounds of 0.000000, indicating that our shield ends correctly with no shield failure or reaching unsafe states.

3 Case Study

In this case study, we compare multiple shield process specifications in a more general MAPF setting, using the unshielded case as the baseline.

We evaluate random instances with $n = 2$ and $n = 3$ agents on 3×3, 4×4, and 5×5 grids, using the following shield process specifications:

$$P_1 = \mu X.S_{safe}.X$$
$$P_2 = \mu X.(S_{safe}.X \|_{g_{o_1}} (S_{safe}.X \|_{g_{o_2}} (\cdots (S_{safe}.X \|_{g_{o_{\cdots}}} \mathsf{fail}))))$$

where S_{safe} ranges over safe states, and g_o enumerates all global states consistent with each joint observation $o \in \Omega$.

Process P_1 is a conservative shield that primarily aims to keep the system in safe states; to this end, it often forces each agent to remain in its original position.

Process P_2 is the least conservative shield obtainable under our current pipeline. It minimises the loss of actions due to intersections introduced by the branch structure (i.e., it avoids disallowing multiple partial observations whenever possible) and, conversely, admits the largest possible set of next safe states.

In addition to the number of agents and the grid size, our case study also varies several environmental and evaluation parameters. Specifically, for each configuration we keep the number of obstacles fixed, since obstacle density is associated with the probability of vertex conflicts. We then evaluate each run along three dimensions: collision (vertex conflicts), shield failure ($\perp$ outputs from the local Mealy machines), and reached (all agents reaching their targets).

grid	n	R	obstacles	shield	collision	shield failure	reached
4×4	2	–	9	/	0.727	0.000	0.273
4×4	2	2	9	P_1	0.000	0.000	0.051
4×4	2	2	9	P_2	0.000	0.000	0.929
4×4	2	1	9	P_1	0.000	0.000	0.055
4×4	2	1	9	P_2	0.000	0.000	0.906
4×4	2	0	9	P_1	0.000	0.000	0.046
4×4	2	0	9	P_2	0.000	0.325	0.295
3×3	3	–	3	/	0.986	0.000	0.015
3×3	3	1	3	P_1	0.000	0.000	0.017
3×3	3	1	3	P_2	0.000	0.363	0.292
4×4	3	–	9	/	0.969	0.000	0.031
4×4	3	1	9	P_1	0.000	0.000	0.006
4×4	3	1	9	P_2	0.000	0.414	0.256
4×4	3	2	9	P_1	0.000	0.000	0.009
4×4	3	2	9	P_2	0.000	0.022	0.533

Table 1: MAPF experiments under a random policy.

We first benchmark the different shield process specifications under a random policy across some MAPF configurations (Table 1). Here, a random policy means that each agent samples uniformly from its currently available actions; in the shielded cases, this uniform sampling is taken over the action set returned by the shield.

In Table 1, across all configurations, the unshielded random policy yields high collision, as it does not discriminate between colliding and non-colliding actions. In contrast, both shields eliminate vertex conflicts in all configurations, providing a strict safety improvement over the unshielded baseline.

Among the shielded, P_2 consistently attains a higher reached than both P_1 and the unshielded model. P_2 removes colliding joint actions while still admitting many safe moves, so agents can explore more states and execute more actions under a random policy. By comparison, the conservatism of P_1 over-suppresses many potential safe actions.

Varying the observation radius R shows that a larger observation radius improves the performance of P_2. As R increases, P_2 achieves higher reachability, since richer observations enable more precise belief updates and thus larger safe action sets. shield failure occurs in configurations where (e.g., $n = 3$, $R = 1$ with dense obstacles), limited observability prevents the shield from always proposing a non-empty safe joint action. As R increases, shield failure decreases rapidly.

We next evaluate four policies trained using Q-learning with Q-tables [12] in a 3×3 MAPF configuration with uniformly random obstacle numbers across training and test environments (Figure 6). Q-learning is a reinforcement-learning algorithm that updates action values from sampled transitions, and a Q-table is the finite table storing those state-action values. The policies are: Q_{reach}, trained to maximise the probability of reaching targets only; Q_{safe}, trained to maximise target reach while explicitly penalising vertex conflicts in the reward function; $Q_{\text{reach with } P_1}$, trained to maximise reach, with shield process P_1 enforced during interaction; and $Q_{\text{reach with } P_2}$, trained to maximise reach, with shield process P_2 enforced.

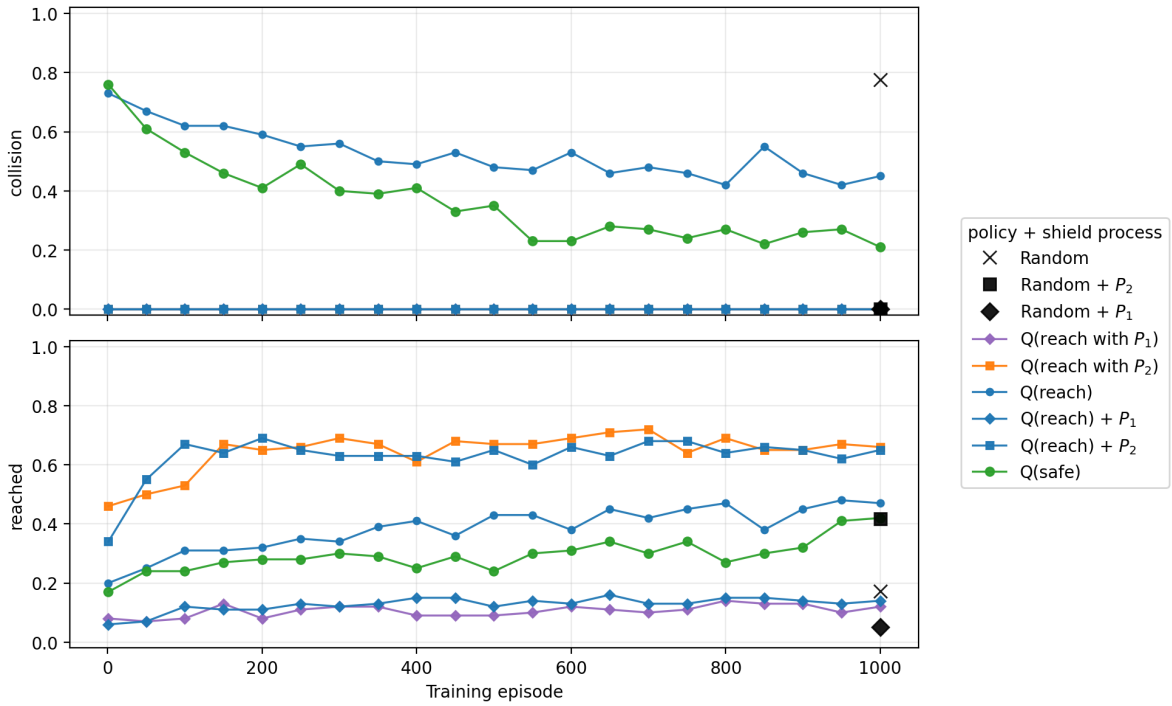


Figure 6: Evaluation curves under training for learned policies with shields under 3×3, $n = 2$, $R = 1$ configuration

For comparison, the figure also includes random policies (unshielded, shielded by P_1, and shielded by P_2) as baselines, as well as the post-hoc shielding of Q_{reach} by P_1 and by P_2. shield failure does not occur in this configuration and is therefore omitted from the plots. Black dots indicate the random baselines.

The learning curves show that both Q_{reach} and Q_{safe} exhibit non-zero collision, but these rates decrease over training. collision declines faster for Q_{safe}, as expected, due to the explicit vertex conflicts penalty. All shielded variants maintain zero collision throughout, demonstrating that the shields enforce safety regardless of the underlying policy. In terms of reached, Q_{safe} attains slightly lower success rates than Q_{reach}, reflecting the trade-off introduced by the vertex conflicts penalty in learning.

The best performance in reached is obtained by $Q_{\text{reach with } P_2}$ and by shielding Q_{reach} with P_2. These combinations achieve high reached while maintaining zero collision, showing that P_2 provides strong safety guarantees without unduly restricting actions. In contrast, policies trained with P_1 or evaluated under P_1 perform only slightly better than the random policy with P_1: the over-conservatism of P_1 severely limits movement, so even a trained policy cannot substantially exceed the baseline, though it may still

discover narrow paths to some targets. Overall, these results reinforce the earlier findings: P_2 delivers a more favourable safety-performance trade-off than P_1, and all shielding can strictly improve safety relative to the unshielded while preserving, or even enhancing, task performance across environments.

4 Related Work

Bloem et al. [2] establish reactive-synthesis and temporal-logic foundations that underpin many runtime-enforcement and shield-style artefacts. Alshiekh et al. [1] propose temporal-logic shields for safe reinforcement learning by synthesising an online action filter that blocks unsafe actions during learning and execution. Jansen et al. [4] propose probabilistic shields that use probabilistic model checking to restrict actions so that safety objectives hold with explicit probability guarantees. Elsayed-Aly et al. [3] extend shielding to multi-agent reinforcement learning by enforcing safety properties in settings where multiple agents jointly determine outcomes. Melcer et al. [6] introduce shield decentralisation by decomposing a centralised safe-action filter into per-agent filters suitable for decentralised execution. Melcer et al. [7] study decentralisation under general partial observability via a SAT-based formulation that decides whether local action sets can be chosen so that their product remains within the centralised safe joint-action set. Melcer et al. [8] explore learned shields for multi-agent reinforcement learning by training shield policies when explicit models or specifications are unavailable or costly to engineer. El Mqirmi et al. [9] propose abstraction-based post-hoc checking for deep multi-agent reinforcement learning, verifying learned behaviours against desired properties rather than filtering actions online.

5 Conclusion and Future Work

This paper presents a shield process algebra for specifying safe global behaviour under Dec-POMDP, together with a compilation pipeline that produces local Mealy shields executable from local observations. The resulting local Mealy shields enforce safety without communication or shared global states and are implemented in Rust with PRISM integration for model analysis. Our case study illustrates how the algebra and pipeline can eliminate collisions while preserving task performance.

Future work includes optimising the decentralised joint-action decomposition (Dec), for example by integrating SAT-based decentralisation methods [7]; combining our approach with learned shields [8] to reduce manual specification effort; and improving robustness through probabilistic shield design [4]. Another direction is to extend the algebra with compositional constructs, such as combinations of shield processes, to improve both semantic clarity and pipeline efficiency.

Acknowledgements

The first author received travel support to attend PLACES 2026 from Balliol College, University of Oxford, through the Donald Michie Scholarship Fund.

The second author is partially supported by EPSRC grants EP/T006544/2, EP/T014709/2, EP/Y005244/1, EP/V000462/1, EP/X015955/1, EP/Z0005801/1; Horizon EU TaRDIS 101093006 (UKRI No. 10066667); and ARIA.

References

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum & Ufuk Topcu (2018): *Safe Reinforcement Learning via Shielding*. *Proceedings of the AAAI Conference on Artificial Intelligence* 32(1), doi:10.1609/aaai.v32i1.11797. Available at `https://ojs.aaai.org/index.php/AAAI/article/view/11797`.

[2] Roderick Bloem, Bettina Koenighofer, Robert Koenighofer & Chao Wang (2015): *Shield Synthesis: Runtime Enforcement for Reactive Systems*, doi:10.48550/arXiv.1501.02573. arXiv:1501.02573.

[3] Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu & Lu Feng (2021): *Safe Multi-Agent Reinforcement Learning via Shielding*, doi:10.48550/arXiv.2101.11196. arXiv:2101.11196.

[4] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban & Roderick Bloem (2020): *Safe Reinforcement Learning Using Probabilistic Shields*. In Igor Konnov & Laura Kovacs, editors: *31st International Conference on Concurrency Theory, CONCUR 2020*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, pp. 31–316, doi:10.4230/LIPIcs.CONCUR.2020.3. 31st International Conference on Concurrency Theory, CONCUR 2020 ; Conference date: 01-09-2020 Through 04-09-2020.

[5] George H. Mealy (1955): *A method for synthesizing sequential circuits*. *The Bell System Technical Journal* 34(5), pp. 1045–1079, doi:10.1002/j.1538-7305.1955.tb03788.x.

[6] Daniel Melcer, Christopher Amato & Stavros Tripakis (2022): *Shield Decentralization for Safe Multi-Agent Reinforcement Learning*. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho & A. Oh, editors: *Advances in Neural Information Processing Systems*, 35, Curran Associates, Inc., pp. 13367–13379. Available at `https://proceedings.neurips.cc/paper_files/paper/2022/file/57444e14ecd9e2c8f603b4f012ce3811-Paper-Conference.pdf`.

[7] Daniel Melcer, Christopher Amato & Stavros Tripakis (2024): *Shield Decentralization for Safe Reinforcement Learning in General Partially Observable Multi-Agent Environments*. pp. 2384–2386, doi:10.65109/VAAR1124.

[8] Daniel Melcer, Stavros Tripakis & Christopher Amato (2025): *Learned Shields for Multi-Agent Reinforcement Learning*. In: *The Seventeenth Workshop on Adaptive and Learning Agents*. Available at `https://openreview.net/forum?id=DXHxmyq5kO`.

[9] Pierre El Mqirmi, Francesco Belardinelli & Borja G. León (2021): *An Abstraction-based Method to Verify Multi-Agent Deep Reinforcement-Learning Behaviours*. *CoRR* abs/2102.01434, doi:10.48550/arXiv.2102.01434. arXiv:2102.01434.

[10] Frans Oliehoek & Christopher Amato (2016): *A Concise Introduction to Decentralized POMDPs*. doi:10.1007/978-3-319-28929-8.

[11] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski & Roman Barták (2019): *Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks*. *CoRR* abs/1906.08291, doi:10.48550/arXiv.1906.08291. arXiv:1906.08291.

[12] Christopher J. C. H. Watkins & Peter Dayan (1992): *Technical Note: Q -Learning*. *Mach. Learn.* 8(3–4), p. 279–292, doi:10.1007/BF00992698.