Machine Learning (2022) 111:2255–2274
https://doi.org/10.1007/s10994-022-06143-6

Planning for potential: efficient safe reinforcement learning

Floris den Hengst[1] · **Vincent François-Lavet**[2] · **Mark Hoogendoorn**[2] · **Frank van Harmelen**[2]

Received: 4 March 2021 / Revised: 28 October 2021 / Accepted: 7 February 2022 /
Published online: 23 March 2022

Abstract

Deep reinforcement learning (DRL) has shown remarkable success in artificial domains and in some real-world applications. However, substantial challenges remain such as learning efficiently under safety constraints. Adherence to safety constraints is a hard requirement in many high-impact application domains such as healthcare and finance. These constraints are preferably represented symbolically to ensure clear semantics at a suitable level of abstraction. Existing approaches to safe DRL assume that being unsafe leads to low rewards. We show that this is a special case of symbolically constrained RL and analyze a generic setting in which total reward and being safe may or may not be correlated. We analyze the impact of symbolic constraints and identify a connection between expected future reward and distance towards a goal in an automaton representation of the constraints. We use this connection in an algorithm for learning complex behaviors safely and efficiently. This algorithm relies on symbolic reasoning over safety constraints to improve the efficiency of a subsymbolic learner with a symbolically obtained measure of progress. We measure sample efficiency on a grid world and a conversational product recommender with real-world constraints. The so-called Planning for Potential algorithm converges quickly and significantly outperforms all baselines. Specifically, we find that symbolic reasoning is necessary for safety during and after learning and can be effectively used to guide a neural learner towards promising areas of the solution space. We conclude that RL can be applied both safely and efficiently when combined with symbolic reasoning.

Keywords Reinforcement learning · Safety constraints · Symbolic planning · Reward shaping

Editor: Nikos Katzouris, Alexander Artikis, Luc De Raedt, Artur d'Avila Garcez, Sebastijan Dumancic, Ute Schmid, Jay Pujara.

✉ Floris den Hengst
Floris.den.Hengst@ing.com

Extended author information available on the last page of the article

1 Introduction

Reinforcement learning (RL) provides an elegant framework for decision making in autonomous agents. In the RL framework, an agent acts in an environment in order to collect rewards (Sutton and Barto 2018). RL driven by neural network-based function approximation, commonly known as Deep Reinforcement Learning (DRL), has recently shown remarkable progress in diverse areas such as personalization (den Hengst et al. 2020), robotics (Gu et al. 2017) and game-playing (Hessel et al. 2018; Silver et al. 2018). The application of DRL in real-world scenarios, however, remains challenging. Despite the significant efforts on making RL agents safe, one of the key challenges remains how to impose "safety constraints that should never [...] be violated" (Dulac-Arnold et al. 2019).

Safety constraints are present in various high-impact domains such as healthcare and finance. Here, regulations and guidelines describe what behaviors are allowed and disallowed. Typically, the behaviors are not listed explicitly, but described by conditions that have to be met at all times. As such, regulations and guidelines form a symbolic and high-level specification of safe behavior in a particular domain. In many thus governed domains, provable compliance to these specifications is an essential prerequisite for the deployment of any system, including DRL-based ones.

Provably safe DRL has recently been approached from the perspective of symbolic reasoning. Symbolic reasoning provides powerful modeling capabilities, unambiguous semantics and well understood computational properties. Fu and Topcu (2014) introduced a framework for checking a bounded safety constraint on a learned model with probabilistic guarantees. Wen et al. (2015) proposed a method for strict adherence, which was extended by Junges et al. (2016) for stochastic settings. Alshiekh et al. (2018) proposed a mechanism that scales to large state-action spaces using a precomputed *shield* which removes actions iff these are unsafe (Bloem et al. 2015).

The above works contain proofs of adherence to the specifications but only target a special case in which being safe is correlated with high expected total rewards. A correlation between high expected total rewards and being safe, however, is not present many important application domains. On the contrary, high rewards can be obtained by engaging in disallowed behaviors in many domains. In such domains, regulations are typically put in place precisely to avoid behaviors that yield high reward but come rare but highly undesirable events, negative long-term consequences and negative externalities. For example, guidelines in healthcare protect organs from damage inflicted during treatment in order to safeguard post-treatment quality of life. Although the problem setting of safe RL in the presence of *antagonistic* constraints has been identified before by e.g. Könighofer et al. (2020), it remains, to the best of our knowledge, largely unexplored how these constraints affect performance and how to mitigate negative effects.

In this work, we identify that safe policies do not outperform unsafe policies in terms of expected reward. We theoretically analyze how symbolic safety constraints impact expected reward and identify a connection between expected future reward and distance to a goal in a symbolic representation of the safety component. We then introduce an algorithm for safe and efficient RL using this distance. By reasoning over the specification and a symbolic goal at an abstract level, an additional reward function is derived *automatically* and supplied to the low-level learner, following the tradition of potential-based reward shaping. This ensures that the optimality of the solution is not at stake if the symbolic plan is incomplete or even incorrect. Our algorithm includes an approach to estimate the *shaping rewards* in an online fashion so that no additional hyperparameters are required.

We evaluate the novel approach, called planning for potential (P4P), on a grid world and on a conversational product recommender. The former was inspired by previous work on safe RL whereas the latter contains real-world regulatory constraints from the banking domain. We compare P4P with a 'vanilla' unsafe baseline and a safe baseline. Additionally, we analyze performance of P4P on increasingly constrained problems. We find that P4P scales well with constraint complexity, is robust with regard to its additional parameter and significantly outperforms the baselines in terms of safety and obtained reward.

This paper is structured as follows: after the preliminaries, we formally introduce the setting of environments with symbolic safety constraints and derive a bound on performance of safe policies. We then show a relation between rewards and reasoning over symbolic constraints using a distance metric. This metric is based on the number of transitions in an automaton representation of the symbolic safety component. We then use this relation in a novel algorithm to improve sample efficiency of RL. We test this algorithm on a simple grid world with a tabular RL algorithm and on a realistic conversational product recommendation benchmark with DRL. The proposed algorithm outperforms all baselines and is the only algorithm capable of solving the realistic task, indicating that symbolic reasoning at an abstract level can be combined with learning via reward shaping as proposed in the P4P algorithm.

2 Related work

In this section, we relate this work to the wider body of work on symbolic safety constraints in RL and the use of symbolic reasoning to improve RL agents. Starting with RL under symbolic safety constraints, we group all the works discussed in the introduction. On top of these, Zhang et al. (2019) encode legality of actions explicitly into rules. Tomic et al. (2020) specifically target learning normative behaviors in a particular normative framework, whereas we focus on high-level, intensional safety constraints. More importantly, most of these target environments where being safe is positively correlated with high total reward which we show to be a special case of safety-constrained RL. The setting of safe RL under constraints that may impact performance negatively was empirically identified in Könighofer et al. (2020). We analyze this problem theoretically and propose an algorithm to learn efficiently in this setting. These works are related in the sense that the contributions presented here are complementary: we argue that symbolic reasoning and reward shaping are important components to making these systems viable in realistic scenarios where rewards and staying safe are not positively correlated.

Closely related are works on restraining bolts by De Giacomo et al. (2019, 2020). These use a similar form of reward shaping based on progress in an automaton representation of LTL$_f$/LDL$_f$ specifications of undesired behaviors. These works, however, target a setting in which an external regulator has no control over the agent except for external transitions to the reward. This is overly restricted for cases where the agent is controlled by actors that want to adhere to safety specifications such as in healthcare. More importantly, this setting eliminates guarantees of safety. This work, on the other hand, targets a setting in which we control the agent fully and where provable guarantees are required. Another recent work by Hasanbeig et al. (2020), presents an approach for safety-constrained RL under the assumptions of knowledge about the transition function, full observability of adjacent state labels and a task fully expressed in LTL. Our work only requires prior knowledge of nonzero transition probabilities at a symbolic level and a goal that expresses which parts of the state-action space are associated with high reward.

A second line of related work aims to inform a learner of knowledge obtained by symbolic reasoning or planning. Grzes and Kudenko (2008) combine STRIPS-based plans with reward shaping. More recently, several works have proposed using some normal form for representing reward functions (Brafman et al. 2018; Icarte et al. 2018; Camacho et al. 2019; Gaon and Brafman 2020). Of specific interest is the work proposing to specify the reward function as an LTL formula and derive intermediate rewards for reward shaping (Camacho et al. 2017). These works focus on a scenario where the full task can be represented as an LTL goal whereas our approach targets unknown numeric reward function to which a reward is added. We, on the other hand, combine sub-symbolic learning with high-level symbolic reasoning to improve efficiency in a setting with unknown MDP reward and transition functions.

A recent work by Hasanbeig et al. (2021) proposes various interesting innovations in this setting. The most relevant of these in relation to our work is an intrinsic reward based on the observation of novel state labels. This intrinsic reward differs from our shaping reward in four important ways. Firstly, it is defined over the state labeling vocabulary Σ_I whereas our approach is based on automata states and hence captures information over *traces* of *both* state and action labels, i.e. over $(\Sigma_I \times \Sigma_O)^\infty$. Secondly, the intrinsic rewards proposed by Hasanbeig et al. would also reinforce moving further from the goal if there happen to be novel labels there. Our shaping approach only reinforces getting closer to the goal. Thirdly, the intrinsic rewards of Hasanbeig et al. do not rely on potentials and hence may produce optimal policies that are suboptimal to original reward function (Bellemare et al. 2016; Burda et al. 2019. Finally, we contribute an approach for tuning the single novel hyperparameter in our approach automatically and on-the-fly.

Illanes et al. (2020) inform an RL agent with high-level symbolic instructions using the options framework in which low-level learned policies can be reused. The instructions are of a directive nature which is arguably less generic than the constraints used here. Additionally, policies learned in the option framework are sub-optimal whereas our reward shaping approach maintains optimality guarantees of the underlying learner.

3 Preliminaries

3.1 Safety specifications and shield synthesis

We define a finite or infinite sequence of elements from some alphabet as a word and a linear-time (LT) property as a set of finite or infinite words over the alphabet $\Sigma : 2^{AP}$, where AP a set of atomic propositions. We focus on safety properties in terms of system input and output and identify subsets of AP and Σ, relating to these as AP_I, Σ_I for inputs and AP_O, Σ_O for outputs. An invariant is an LT property that has to hold in all reachable states for some system, for example "a product may only be recommended if it matches the customer risk profile". Safety properties generalize invariance properties to include patterns over time, for example "products may only be recommended *after* the customer's objectives are known" (Baier and Katoen 2008).

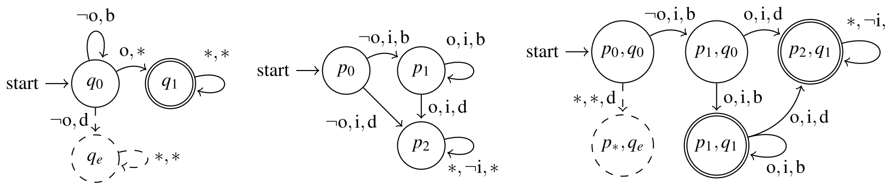
Safety properties can be expressed in a formal language that extends propositional logic with temporal operators. Linear temporal logic (LTL) is such a logic (Pnueli 1977). LTL extends propositional logic with temporal modal operators $\mathbf{X}$ (next) and $\mathbf{U}$ (until). $\mathbf{X}\varphi$

expresses that a formula φ must be true the next time step and $\psi\mathbf{U}\varphi$ expresses that ψ has to hold at least until φ becomes true. From these, the operators $\mathbf{G}\varphi$ (globally) and $\mathbf{F}\varphi$ (finally) can be defined to express that, from a particular step onward, φ has to respectively hold always and at some point in the future respectively. Additionally, the operator $\mathbf{W}$ can be derived, which 'weakens' the $\mathbf{U}$ operators assumptions by allowing that its right-hand-side may or may not be the true in the future.

A LTL safety specification φ_s can automatically be converted into an automaton that represents it. A deterministic finite automaton (DFA) $\varphi_a = \langle \mathbb{Q}, q_0, \Sigma, \delta, \mathbb{F} \rangle$ consists of a set of states $\mathbb{Q}$, an initial state $q_0 \in \mathbb{Q}$, an alphabet $\Sigma = \Sigma_I \times \Sigma_O$, a transition function $\delta : \mathbb{Q} \times \Sigma \rightarrow \mathbb{Q}$ and a set of safe state $\mathbb{F} \subseteq \mathbb{Q}$. A *run* is a finite or infinite sequence of states $\bar{q} = q_0, q_1, \ldots \in \mathbb{Q}^\infty$ induced by a trace $\bar{\sigma} = \sigma_0, \sigma_1, \ldots, \in \Sigma^\infty$ of some system such that $\forall i \in \mathbb{N}, q_{i+1} = \delta(q_i, \sigma_i)$. A trace $\bar{\sigma}$ of some system satisfies specification φ_s and its representation φ_a iff the corresponding run $\bar{q}$ visits safe states only, i.e. $\forall i \in \mathbb{N}, q_i \in \mathbb{F}$. An example LTL specification and its automaton representation can be found in Fig. 1a.

If a model of the environment is available, a reactive system that always produces output in accordance with a specification φ_s can be generated. This challenging task is known as *reactive synthesis* (Pnueli and Rosner 1989). A typical strategy is to formulate the problem as a two-player alternating game between the system and an adversarial environment. Such a safety game can be expressed as a tuple $\mathcal{G} : \langle \mathbb{G}, g_0, \Sigma_I, \Sigma_O, \delta, \mathbb{F} \rangle$ with a finite set of game states $\mathbb{G}$, initial state $g_0 \in \mathbb{G}$, a transition function $\delta : \mathbb{G} \times \Sigma_I \times \Sigma_O \rightarrow \mathbb{G}$ and a set of safe states $\mathbb{F} \subseteq \mathbb{G}$. During the game and for the current state $g \in \mathbb{G}$, the environment first chooses some $\sigma_I \in \Sigma_I$ after which the system chooses $\sigma_O \in \Sigma_O$ and the game transitions to state $g' = \delta(g, \sigma_I, \sigma_O)$. The resulting (infinite) sequence $\bar{g} = g_0, g_1, \ldots$ is called a play and is won by the system iff all visited states are safe: $\forall g_i \in \bar{g}, g_i \in \mathbb{F}$. A winning memoryless strategy is a function $\rho : \mathbb{G} \times \Sigma_I \rightarrow \Sigma_O$ if all plays $\bar{g}$ that can be constructed using it are won by the system. Standard algorithms can compute such a winning strategy if it exists (Mazala 2002).

Shield synthesis is a particular kind of reactive synthesis in which an existing system is assumed and in which an external component to correct the system output is computed. The correction is guaranteed to change the output of the original system so that it satisfies some specification with minimal interference given an abstraction of the system (Bloem et al. 2015).



(a) Safety automaton representation of φ **(b)** MDP abstraction **(c)** Safety game

Fig. 1 Example of a car exiting a gated parking lot. The agent can either **d**rive or push a **b**utton next to the gate, i.e. AP_O : {d, b}. State labels indicate whether the gate is **o**pen and whether the car is **i**n the parking lot, i.e. AP_I : {o, i}. **a** An automaton representation of φ : ¬d**W**o to express that the gate should be open before the car may drive, **b** MDP abstraction with all transitions with nonzero probability in the underlying MDP. Initially, the gate is not open and the car is inside. Error transitions and state are not included for legibility. **c** The result of combining (**a**) and (**b**) to form a safety game (excluding abstraction errors). Transitions marked with a solid line are part of the safe strategy. Action 'd' in (p_0, q_0) is not part of the safe strategy since the resulting state is an error state

An abstraction of the system describes how its executions can possibly evolve, and provides the needed information about the environment to allow planning ahead w.r.t. the safety properties of interest. The model required is typically of limited size as result. More so, as it can be expressed in an equivalent lifted representation. It may therefore be easy to construct or learn from data. An illustrative abstraction of an MDP and resulting safety game can be found in Fig. 1. The corresponding shield would replace any unsafe action with a next best safe action.

3.2 Reinforcement learning

RL provides a framework for selecting actions in an environment in order to collect a maximum number of rewards over time (Sutton and Barto 2018; Wiering and Van Otterlo 2012). RL deals with problems formalized as Markov decision problems (MDP). We here define a MDP as a tuple $M : \langle S, A, T, R, \gamma, S_0 \rangle$ where $S \in \{s^{(1)}, \dots, s^{(n)}\}$ is a finite set of environment states, $A \in \{a^{(1)}, \dots, a^{(m)}\}$ a finite set of agent actions, $T : S \times A \times S \to [0, 1]$ a probabilistic transition function, $R : S \times A \times S \to [R_{\min}, R_{\max}]$ a reward function with $R_{\min}, R_{\max} \in \mathbb{R}$, $\gamma \in [0, 1)$ a discount factor to balance current and future rewards and S_0 a distribution of initial states: $s_0 \sim S_0$. The agent observes an environment state s_t at each time step t and performs some action a_t up to some end time $\mathcal{T}$, following some policy $\pi \in \Pi : S \times A \to [0, 1]$ and collects reward $r_t = R(s_t, a_t)$.

If some expectation $\mathbb{E}_\pi$ can be formulated to express the sum of rewards by following some π, then values V and Q can be assigned to a state s and a tuple (s, a) respectively for that π:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=t}^{\mathcal{T}=\infty} \gamma^{k-t} r_k | s_t = s \right] \tag{1}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=t}^{\mathcal{T}=\infty} \gamma^{k-t} r_k | s_t = s, a_t = a \right] \tag{2}$$

A policy π is the optimal policy π^* if it results in the highest obtainable reward: $\forall s \in S, \forall \pi \in \Pi, \forall a \in A : Q_{\pi^*}(s, a) \geq Q_\pi(s, a)$. Finding π^* can be addressed by conditioning the policy on a set of parameters $\pi(s_t|\theta) = a_t$ and finding parameter values θ^* that maximize the corresponding reward by a learning algorithm. For example, θ can be weights of a neural network updated with gradient descent.

A particularly popular parameterized approach of learning an approximation of π^* is known as deep Q-Networks (DQN) (Mnih et al. 2013, 2015). DQN uses a neural network with weights θ to predict $Q_\pi(s, a|\theta)$ and selects actions uniform randomly with some probability $\epsilon \in (0, 1]$ or greedily with respect to $Q_\pi(s, a)$ with some probability $1 - \epsilon$ at each step t. The resulting tuple (s_t, a_t, r_t, s_{t+1}) is added to a buffer or data set D as (s, a, r, s'). Weights are updated in iterations. For every iteration i, the current weights θ_i are updated to minimize the loss function $\mathcal{L}_i(\theta_i) =$

$$\mathbb{E}_{(s,a,r,s') \sim U(D)} \left(r + \gamma \max_{a'} Q(s', a'|\theta_i^-) - Q(s, a|\theta_i) \right)^2 \tag{3}$$

where $U(D)$ is a uniform random sample of D and θ_i^- the parameters used in action selection during iteration i. These parameters θ_i^- are only replaced with θ_i every C iterations and held fixed otherwise as this increases stability of the learned Q-network over time, hence improving performance.

4 Safe reinforcement learning

RL has proven capable of learning complex behaviors from interactions with an environment in a trial-and-error fashion alone. Symbolic reasoning, on the other hand, is well suited when safety guarantees on behavior are necessary. Safe RL combines these in order to learn complex behaviors under strong guarantees of safety, both during and after learning. In this section, we introduce safety-constrained environments following Alshiekh et al. (2018), identify that safe policies are not expected to outperform unsafe policies and then analyze how safety constraints impact the expected future reward.

Definition 1 (*Safety-constrained environments*) A *safety-constrained environment* is a tuple $E : \langle M, AP, L_I, L_O, \varphi\rangle$ where $M : \langle S, A, T, R, \gamma\rangle$ is an MDP, φ is a safety specification with propositions $AP : AP_I \cup AP_O$ and labelling functions $L_I : S \rightarrow 2^{AP_I}$, $L_O : A \rightarrow 2^{AP_O}$.

Definition 2 (*Safe policies*) A policy $\pi \in \Pi : S \times A \rightarrow [0, 1]$ is *safe* in E if for any sequence $s_t, a_t, s_{t+1}, a_{t+1}, \ldots$ it generates with nonzero probability, the corresponding sequence of labels $(L_I(s_t) \cup L_O(a_t), L_I(s_{t+1}) \cup L_O(a_{t+1}), \ldots)$ satisfies φ. The set of safe policies in E is denoted Π_E.

For the purposes of this paper, the MDP transition function T is unknown. If results of actions are unknown, safety of a given policy cannot be verified up-front without further assumptions. In order to ensure safety, however, we need only know which sequences of labels for a given policy in an environment have a nonzero probability. These can be modeled with an automaton abstraction of the MDP.

Definition 3 (*MDP abstractions*) Given an environment E, the automaton $\varphi_M : \langle \mathbb{Q}, q_0, \Sigma_I \times \Sigma_O, \delta, \mathbb{F}\rangle$ is an abstraction of M if for every trace $s_0, s_1, \ldots \in S^\infty$ and corresponding action sequence $a_0, a_1, \ldots \in A^\infty$ with nonzero probability in E, for every run $\bar{q} : q_0, q_1, \ldots \in \mathbb{Q}^\infty$ with $q_{i+1} = \delta(q_i, L_I(s_i), L_O(a_i))$, this run $\bar{q}$ visits only states in $\mathbb{F}$.

Remark 1 It can be verified whether an automaton is an abstraction of M. If a run is generated in which some state $q_i \notin \mathbb{F}$ then it is not an abstraction of M. This property can be used to refine the abstraction when it is tested or to hand over control to a human operator or fallback policy.

An abstraction can be used to synthesize safe policies. The cross product of the abstraction and an automaton representation of the specification forms a safety game from which a safe strategy can be computed as described in the previous section. Policies following this strategy are provably safe in E (Alshiekh et al. 2018). We now introduce a performance bound for safe policies.

Theorem 1 (Performance bound) *For any environment E with MDP M and safe policies Π_E, let $\pi^*_E \in \Pi_E$ be the optimal safe policy and $\pi^*_M \in \Pi$ be the optimal (possibly unsafe) policy. Then $\pi^*_E \leq \pi^*_M$ where $\pi_1 \leq \pi_2$ iff $\forall s \in S, \forall a \in A, Q_{\pi_1}(s, a) \leq Q_{\pi_2}(s, a)$.*

Proof $\Pi_E \subseteq \Pi$, hence $\pi^*_E \in \Pi$ and $\pi^*_E \leq \pi^*_M$. □

Theorem 1 shows that safe policies are generally not expected to outperform their unsafe counterparts in terms of reward: the environments targeted in previous work where being unsafe leads to low rewards are a special case of safety constrained environments. We continue to investigate when constraints negatively impact expected reward. In order to do so, we first introduce the notion of a goal. Problems with pre-specified goals are approached within the framework of goal-based MDPs in RL. Such MDPs terminate if a goal state $s_g \in S$ is reached and have a reward function of the form $R(s,a,s') = 1$ if $s' = s_g$ and 0 otherwise. Here, we consider goal-based problems within the framework of safety-constrained environments and define them in terms of AP.

Definition 4 (*Goals*) For a safety-constrained environment $E : \langle M, AP, L_I, L_O, \varphi \rangle$, a goal $\sigma_g \in 2^{AP}$ is reached if an action $a \in A$ with labeling $\sigma_a : L_O(a)$ is selected in a state $s \in S$ with labeling $\sigma_s : L_I(s)$ such that $\sigma_g \subseteq \sigma_s \cup \sigma_a$.

Definition 5 (*Goal-based environments*) An environment $E : \langle M, AP, L_I, L_O, \varphi, \sigma_g \rangle$ with goal σ_g is goal-based if it has a reward function of the following restricted form: $R(s,a,s') = 1$ if the goal is reached by performing a in s and $R(s,a,s') = 0$ otherwise.

How a constraint specifically impacts expected reward depends on the particular goal, the constraint and the transition function T, which is unknown in the case of interest here. Even in this case, however, the impact of a constraint can be derived in some illustrative cases which serve as an inspiration to the algorithm presented later. First, we focus on the case where the goal can be reached immediately.

Theorem 2 (Q values and reaching goals) *For any goal-based environment E with safe optimal policy π^*_E and for any $s, s' \in S$ and any $a, a' \in A$ with $\sigma_s : L_I(s), \sigma_{s'} : L_I(s'), \sigma_a : L_O(a)$ and $\sigma_{a'} : L'_O(a')$:*

$$\begin{aligned} Q_{\pi^*_E}(s,a) > Q_{\pi^*_E}(s',a') \qquad & \text{if} \\ \sigma_g \subseteq \sigma_s \cup \sigma_a \qquad & \text{and} \\ \sigma_g \nsubseteq \sigma_{s'} \cup \sigma_{a'} \qquad & \end{aligned}$$

Proof If $\sigma_g \subseteq \sigma_s \cup \sigma_a$ then performing a in s ends the episode in E and yields the maximum obtainable reward of $R(s,a,\cdot) = 1$. Since $\sigma_g \nsubseteq \sigma_{s'} \cup \sigma_{a'}$, $R(s',a',\cdot) = 0$ and therefore $Q_{\pi^*_E}(s,a) > Q_{\pi^*_E}(s',a')$. □

Corollary 1 *For an environment E in some state $s \in S$ for any two actions $a, a' \in A$ such that $\sigma_g \subseteq \sigma_s \cup \sigma_a$ and $\sigma_g \nsubseteq \sigma_s \cup \sigma_{a'}$:*

$$Q_{\pi^*_E}(s,a) > Q_{\pi^*_E}(s,a')$$

Proof By substituting $s' = s$ in Theorem 2. □

When the goal cannot be reached immediately, reaching the goal requires multiple transitions in both the underlying MDP and the safety game $\mathcal{G}$. The minimum number of transitions required in $\mathcal{G}$ for any of its safe states $f \in \mathbb{F}$ is denoted $\Delta_{\mathcal{G}}(f_i, \sigma_g)$. It can be derived using planning and interpreted as the distance from f to the goal while staying safe.

Definition 6 (*Distance in safety games*) For a goal-based environment E with safety game $\mathcal{G} : \langle \mathbb{Q}, q_0, \Sigma_I, \Sigma_O, \delta, \mathbb{F} \rangle$ with a winning strategy ρ, a distance map $\Delta_\mathcal{G} : \mathbb{F} \to \mathbb{N}_1$ is defined as the length of the shortest play $\bar{f} = f_i, \dots, f_k, f_l$ starting in any $f_i \in \mathbb{F}$ with $\delta(f_k, \sigma_i, \sigma_o) = f_l$ such that the play $\bar{f}$ can be constructed with ρ and $\sigma_g \subseteq \sigma_i \cup \sigma_o$.

Remark 2 $\Delta_\mathcal{G}(f, \sigma_g) = 1$ iff $\sigma_g \subseteq \sigma_s \cup \sigma_a$ for some $a \in A$ in a given $s \in S, f \in \mathbb{F}$

Reaching the goal may also require multiple transitions in the MDP. Since T is unknown in the setting of interest, the expected number of transitions is unknown as well. Therefore, we look into a second illustrative case where safety constraints and goals are defined fully on the action space, i.e. $AP_I = \emptyset$.

Theorem 3 (Q values and distances) *For any goal-based environment E with safety game* $\mathcal{G}$ *and for any* $s, s' \in S$, $a, a' \in A$, $f, f' \in \mathbb{F}$:

$$\begin{aligned} Q_{\pi_E^*}(s,a) > Q_{\pi_E^*}(s',a') \qquad & \text{if} \\ \Delta_\mathcal{G}(f', \sigma_g) > 1 \qquad & \text{and} \\ \Delta_\mathcal{G}(\delta(f, \emptyset, \sigma_a), \sigma_g) < \Delta_\mathcal{G}(\delta(f', \emptyset, \sigma_{a'}), \sigma_g) & \end{aligned}$$

(sketch) Suppose that the consequent of the equation holds. Let n denote the distance towards a goal by taking a, $n = \Delta_\mathcal{G}(\delta(f, \emptyset, \sigma_a), \sigma_g)$, and n' denote the distance towards a goal by taking action a', $n' = \Delta_\mathcal{G}(\delta(f', \emptyset, \sigma_{a'}), \sigma_g)$. An optimal safe policy in E takes n time steps to reach σ_g after performing a and similarly so for n' and a'. Now by substituting $\mathcal{T} = n$ and $\mathcal{T} = n'$ in Eq. 2 and since $\gamma < 1$ and $n < n'$ we find $Q_{\pi_E^*}(s,a) > Q_{\pi_E^*}(s',a')$. □

Remark 3 An inequality for a single $s \in S$ and single $f \in \mathbb{F}$ can be derived analogously to Corollary 1.

The presented analysis shows how Q values relate to goals and distances to goals in safety games. Although the analysis is targeted at goal-based environments, their implications are also applicable to other settings, such as those in which the symbolic goal serves as a proxy for a high reward area of the state-action space. Our analysis indicates that there are two classes of safety constrained environments and it shows how to identify them. In the first class, safe policies are expected to perform equally to unsafe policies in terms of obtained rewards. The distance from initial state to a goal in the associated safety game is not impacted by the safety constraints in these environments: they would be equal to the distance of a safety game resulting from vacuous constraints that always hold. In the second class of safety constrained environments, unsafe policies are expected to outperform safe ones. The safety constraints add transitions to the shortest path towards a goal. As constraints are added, the distance from the goal grows. Every transition that a safety constraint contributes, leads to at least one additional transition for the learner to incorporate and as such makes the learning problem more complex. The next section introduces an algorithm to improve the scalability of safe RL as problems become more constrained.

5 Planning for potential

In this section, we propose a scalable and efficient RL algorithm adhering to a safety specification. The algorithm uses symbolic knowledge available in a safety-constrained environments to speed up the learning. Optimality is preserved in cases of incomplete or even incorrect prior knowledge. In the proposed approach, the learner is informed of progress with respect to a symbolic goal by transforming the reward function *automatically*. The approach follows the tradition of potential-based reward shaping which we introduce first.

5.1 Reward shaping

Shaping is a technique within RL in which the original MDP $M : \langle S, A, T, R, \gamma \rangle$ is replaced with a surrogate $M' : \langle S, A, T, R', \gamma \rangle$ in order to guide the learner. It is desirable that R' is easier to learn but yields only optimal policies that are also optimal under the original R. Ng et al. (1999) showed that $R' = R + \mathcal{S}$ with shaping function $\mathcal{S} : S \times A \times S \rightarrow \mathbb{R}$ such that $\mathcal{S}(s, a, s') = \gamma\Phi(s') - \Phi(s)$ with so-called potential $\Phi : S \rightarrow \mathbb{R}$ are the only R' that guarantee that any policy optimal in M' is also optimal in M if no further information on transition and reward functions is known. The challenge now consists of defining a potential function Φ that informs the learner.

In safety-constrained RL, knowledge about the task is available in symbolic form in the safety component. To use this knowledge, a description of state-action tuples associated with high reward are described in symbolic form. The distance to this symbolic goal is established with planning. By comparing distances prior to and after taking an action, we know whether that action contributed to reaching the symbolic goal. The agent is informed of this with potential-based reward shaping, hence we call this approach planning for potential (abbreviated P4P).

5.2 Algorithm

P4P is listed in Algorithm 1. For a given safety constraint and MDP abstraction, a safety game and shield are computed following Alshiekh et al. (2018). Next, a map of potentials Φ is computed for all safe states of the safety game, after which the learning loop begins. This is a traditional RL learning loop with two modifications: actions are selected from the set of safe actions (line 11) and the reward is augmented with the difference between potentials (line 15).

Algorithm 1: Efficient RL in a safety-constrained environment with P4P.

```
   Inputs : specification $\varphi$, abstraction $\varphi_M$, goal $\sigma_g$, cost $c > 0$
   Output: policy $\pi$
 1 $\mathcal{G} \leftarrow \varphi \times \varphi_M$
 2 shield $\leftarrow$ computeShield($\mathcal{G}$)
 3 $\Phi \leftarrow$ potentials($\mathcal{G}, \sigma_g, c$) ;                    // see Algorithm 2
 4 Initialize $\pi$ arbitrarily
 5 foreach episode do
 6     $g \leftarrow g_0$ from $\mathcal{G}$
 7     foreach time step do
 8         $s \leftarrow$ get from environment
 9         $\sigma_s \leftarrow L_I(s)$
10         $a \leftarrow$ select safe action from shield, $\pi$
11         Take action $a$
12         $r, s' \leftarrow$ get from environment
13         $\sigma_a \leftarrow L_O(a)$
14         $g' \leftarrow \delta(g, \sigma_s, \sigma_a)$
15         $r' \leftarrow r + \gamma\Phi(g') - \Phi(g)$
16         Update $\pi$ using $(s, a, s', r')$
17         $s \leftarrow s', g \leftarrow g'$
18     end
19 end
```

Algorithm 2 lists how the map of potentials Φ can be computed. The minimum number of transitions, or *shortest distance*, from each state to the goal state are determined using symbolic planning. These distances can be derived prior to interacting with the environment and with minimal knowledge of the MDP transition function. This makes them well suited as a signal of progress for an exploring agent in a safety-constrained environment. The algorithm presented here computes potentials for all states upfront. Although limited in terms of scalability with respect to the safety game state space, this simple approach will suffice for many settings for two reasons. Firstly, the safety game only includes aspects of safety and this 'abstraction' over the full MDP yields relatively small safety games. Secondly, the calculation of these values is a one-time operation that is easily dwarfed by the iterative training approach of many RL algorithms used in practice. If necessary, however, more elaborate methods can be applied. Any solution to the unweighted single start shortest path algorithm tailored to the desired performance characteristics can be used. More so, the distances and potentials can be calculated in an online fashion, i.e. deferring the calculation of these values for all states to the moment of first visiting them to increase scalability for settings with a large number of safety game states that are not visited in practice.

Algorithm 2: Planning-based potentials Φ.

```
   Input  : safety game $\mathcal{G}$, goal $\sigma_g$, cost $c > 0$
   Output: potentials $\Phi$
1  Initialize $\Phi$ for all states in $\mathbb{F}$
2  Initialize $dist$ for all states in $\mathbb{F}$
3  foreach $state \in \mathbb{F}$ do
4      $dist(state) \leftarrow$ CalcDist($state$)
5  end
6  $initDist \leftarrow dist(g_0$ from $\mathcal{G})$
7  foreach $state \in \mathbb{F}$ do
8      $\Phi(state) \leftarrow c \cdot (initDist - dist(state))$
9  end
10 return $\Phi$
11 Procedure CalcDist($state$)
12     $minDist \leftarrow \infty$
13     foreach $\sigma_s, \sigma_a \in \Sigma_I, \Sigma_O$ do
14         if $\sigma_g \subseteq \sigma_s \cup \sigma_a$ then return 1
15         $next \leftarrow \delta(state, \sigma_s, \sigma_a)$
16         if $next \in \mathbb{F}$ then
17             $distance \leftarrow$ CalcDist($next$) $+ 1$
18             $minDist \leftarrow \min(dist, minDist)$
19         end
20     end
21     return $minDist$
```

In order to convert distances to progress, the difference in distances to a goal between the initial state and any other state are calculated. This difference represents the progress for any given state. It is multiplied with cost parameter c to offset the 'costs' already incurred while moving closer to the goal. A reward bonus is given to state-action pairs closer to the symbolic goal in accordance to the inequalities of Theorems 2 and 3. As P4P uses potentials to augment the obtained rewards, there are no formal requirements on c except for being > 0 by Theorems 2 and 3. In the case of stochasticity in updates of estimates for V and Q, however, the value of c may affect convergence. In this case, c can be set using knowledge of the domain, tuned as a hyperparameter or estimated during learning in an online fashion, see Sect. 5.4.

5.3 Example

For the safety-constrained environment in Fig. 1, the goal is to not be in the parking lot $\sigma_g : \neg$i. This goal can be reached safely by taking any action in (p_2, q_1). We can only visit this state by first visiting (p_1, q_0) which has potential:

$$\begin{aligned}\Phi((p_1 q_0)) &= c * (\Delta_{\mathcal{G}}((p_0, q_0)) - \Delta_{\mathcal{G}}((p_1, q_0)) \\ &= c * (3 - 2) = c\end{aligned}$$

Transitioning from $(p_0, q_0) \rightarrow (p_1, q_0)$ therefore yields an additional reward of $\gamma \cdot c$. On the other hand, the additional reward for transitioning from $(p_1, q_0) \rightarrow (p_1, q_1) = \gamma \cdot c - c$ and effectively reflects the fact that visiting (p_1, q_1) was not necessary in order to reach the goal.

The additional rewards supplied to the learning algorithm guide the learner to prefer certain states and actions. Doing so involves balancing the provided bias to increase learning without limiting the learner. Two mechanisms in P4P ensure that the amount of bias is suitable. Firstly, the use of potential-based shaping ensures that the optimal policy under transformed reward function is optimal under the original reward as well (see Sect. 5.1). Secondly, progress in the algorithm is based on the shortest safe path toward the goal in the safety game. This may be overly optimistic when the trajectories in the MDP corresponding with this path have low probability. If this is the case, the learner will observe these trajectories infrequently and the effects of P4P will be limited. P4P thus successfully leverages all information available at the symbolic level without overly biasing the learner.

In P4P, distances to a goal are derived by symbolic reasoning and subsequently used to inform a learner via reward shaping. The shaped reward function is more dense than the original reward function and guides the agent towards promising regions of the state-action space in exploration phases. Furthermore, rewards are obtained as the agent progresses towards its goal. Thus, a part of the value assignment problem of RL is already solved for the learner. Finally, the usage of potential-based shaping guarantees that optimality guarantees for the underlying learning algorithm also apply to P4P, even if the provided goal is incomplete or incorrect.

5.4 Estimation of *c*

The parameter c in Algorithms 1 and 2 controls the additional rewards given to the agent for each transition towards a goal in the safety game. As previously noted, there are no formal requirements on setting parameter c, except for $c > 0$ according to Theorems 2 and 3. Although convergence towards the optimal policy is not affected by the value of c in the long run, a suitable value can impact speedups obtained when using function approximation such as in DRL. In this section, we describe two ways to find a suitable value for c. They can be used depending on the available upfront knowledge. The first approach is based on the size of the safety game and existing knowledge of the maximum obtainable reward. If this knowledge is available, then parameter c can be set using the following heuristic:

$$c := \frac{\hat{R}_{\max}}{dist(g_0)} \tag{4}$$

where $\hat{R}_{\max}$ denotes an estimate of $R_{\max}$, the maximum of R, and $dist(g_0)$ denotes the distance of the initial state in the safety game to its closest goal as calculated in Algorithm 2.

Alternatively, c can be tuned in an online fashion from interactions alone. In order to do so, rewards are to be associated with transitions in the safety game. Figure 2 shows how state transitions in the MDP are associated with state transitions in the safety game. Although the agent is successful in the end, a reward of -3 is incurred by transitioning $q_0 \rightarrow q_1$. These 'costs' for safety game transitions are stored in a buffer, averaged and multiplied with -1 in order to establish c in an online fashion. In some environments, reward is 0 most of the time. An example environment is the goal-based environment in Definition 5. In such environments, no rewards are obtained by transitioning in the safety game. An elegant solution for this problem is to linearly transform all observed rewards to $\mathbb{R}_{\leq 0}$ as a first step of estimating c dynamically.

6 Experimental setup

The experimental setup was designed to answer four specific research questions: how can RL agents learn safely and efficiently (Q1)? How do different constraints impact efficiency (Q2)? How sensitive is the proposed approach to its hyperparameters (Q3)? In order to answer these questions, the proposed P4P approach was evaluated in two environments. In these environments, agents were trained in different conditions: a baseline not adhering to the specification ('unsafe'), a 'shielded' baseline from Alshiekh et al. (2018) and three P4P variants. In the first of these variants, the P4P hyperparameter c is estimated online as proposed in Sect. 5.4. The other two variants use a fixed value for this parameter in order to answer Q3. The values are set to overestimate ('P4P-o') and underestimate ('P4P-u') the true cost.

6.1 Grid world environment

A grid world environment from (Alshiekh et al. 2018) with an 'exact' abstraction was used. Grid world environments are often used in RL as they are relatively simple to grasp but include many of the characteristics of more challenging learning problems and are useful to e.g. test intuitions. In the grid world used here (Fig. 3a), being safe does not correlate with high reward. A reward of 1 is obtained when all regions have been visited in order and 0 otherwise. States in gray can be visited but are to be avoided according to the safety requirements. The goal was formulated as $area1 \wedge area2 \wedge area3 \wedge area4$. Parameter c was estimated in an online fashion by storing rewards obtained for each transition in the safety game (see Sect. 5.4). Additionally, we ran experiments with over- and underestimates for c to answer Q3. These were established as follows. In this environment, the average cost of a safety game transition can be calculated. Based on the number of actions necessary to

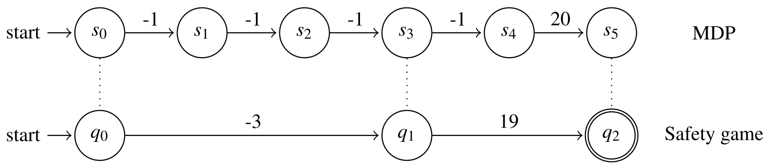


Fig. 2 Traces for a successful episode in a hypothetical MDP (top) and safety game (bottom). Transition labels indicate rewards associated with that transition

visit all regions safely and the reward that is obtained, the average cost was established at 8e−3. The cost parameter was set to $c = 1e{-}5$ as an underestimate (P4P-u) and $c = 2$ as an overestimate (P4P-o) in order to test robustness to this parameter. All agents were trained using ϵ-greedy tabular Q-learning with $\alpha = 0.2$ and $\gamma = .95$ (Watkins and Dayan 1992). Exploration parameter ϵ was cooled down linearly from − 0.2 to 0.01 over the total number of 1e4 episodes. Episode length and number of violations of the safety specification were recorded across ten random seeds.

6.2 Conversational recommendation environment

A realistic and high-dimensional environment from the banking domain was included due to the availability of real-world constraints (den Hengst et al. 2019). In this conversational recommendation environment, the agent interacts with a simulated user. The task is to recommend a product and provide the desired information in a minimal number of turns. Reward is specified as follows: each conversation turn yields a reward of −1 and at the end of each conversation, an additional reward of 20 is obtained if the provided information meets the information need and 0 otherwise. The unconstrained action space consists of 38 actions. States are each represented by a boolean vector with length 136 that describes beliefs over the customers preferences and the dialogue history. The dimensionality of the state-action space is 38×2^{136} and makes this a challenging problem for which function approximation is necessary.

Realistic constraints were constructed from a real-world regulatory document. All statements pertaining to the interaction between a bank and customers were extracted from this document and formalized in consultation with two domain experts. The vocabulary used in the specification is listed in Table 1 and the specifications are listed in Table 2. Separate specifications were used to gauge the impact of different constraints (Q2). Accuracy was recorded for random rollouts for each separate constraint as a proxy for the 'difficulty' of the constraint. Constraints are presented in decreasing difficulty in Table 2. Increasingly difficult specifications were created by combining separate specifications: $\varphi_{1,2} = \varphi_1 \wedge \varphi_2, \varphi_{1,2,3} = \varphi_{1,2} \wedge \varphi_3, \dots, \varphi_{1-6} = \bigwedge \varphi_{1,\dots 6}$. For all constraints, the goal *rec* was used.

Agents were trained in eight conditions: a baseline, P4P with online estimated c (P4P), P4P with an overestimate of c (P4P-o) and P4P with an underestimate of c (P4P-u), all for both the unsafe and shielded case. The under- and over-estimates of c were determined as follows: the reward of a dialogue turn is − 1. However, only some turns result in a transition in the safety game. Therefore, 1 is a reasonable underestimate for c. The overestimate was based on the average number of turns used by an unconstrained agent. After training, it requires on average seven turns to complete the task. Therefore, we used $c = 8$ as an overestimate for each *single* transition towards the goal.

For each condition, five agents with different random seeds were trained on 30K dialogues. After every 1K dialogues, performance was measured on 500 test dialogues. Rewards, accuracy, number of dialogue turns and safety specification violations were recorded. All agents use ϵ-greedy DQN where ϵ is linearly cooled down from 0.3 to $\epsilon_f = 0.05$ and with a learning rate $\alpha = 1e{-}4$. These hyperparameters were selected after a grid search on $\epsilon_s \in \{0.3, 0.5, 0.9\}$, $\epsilon_f \in \{0.05, 0.3\}$ and $\alpha \in \{1e{-}3, 1e{-}4\}$.

Table 1 Atomic propositions in the recommender environment

	Propositions	Explanation
AP_O	*rec*	The agent makes a recommendation
	e	The agent explains the expected result of a recommended product
	ena	The agent explains the need for analysis of the customer profile
	dsp	The agent discloses the customer profile
	dvp	The agent recommends a product that deviates from the risk profile
AP_I	*ok*	The objective of the customer is known
	cdvp	The customer confirms they want to deviate from their risk profile
	vp	The customer verifies the risk profile disclosed by the agent
	ue	The customer indicates to understand the explanation of the result

Table 2 Formalization of regulatory safety statements into LTL specifications

$\varphi_{\#}$	Regulatory statement	Specification
1	Explain the expected result, check whether it is understood	$\mathbf{G}(rec \rightarrow ((e \vee rec)\mathbf{W}ue))$
2	No recommendation if the profile has been disclosed but not verified	$\mathbf{G}(dsp \rightarrow (\neg rec\mathbf{W}vp))$
3	No deviation from risk profile until customer confirms deviation	$\neg dvp\mathbf{W}cdvp$
4	No recommendation until customer objective known	$\neg rec\mathbf{W}ok$
5	No recommendation until the customer profile has been disclosed	$\neg rec\mathbf{W}dsp$
6	No recommendation until the need for analysis has been explained	$\neg rec\mathbf{W}ena$

7 Experimental results

7.1 Grid world environment

Figure 3b shows episode lengths for all included agents in the grid world environment. P4P converges toward an optimal policy quickly (Q1). As a result of the 'exact' abstraction, the potentials reflect progress made for every time step. P4P significantly outperforms both safe and unsafe baselines, which both achieve optimal behavior eventually. More so, we see that both P4P-u and P4P-o converge faster than P4P (Q3). This is explained by the fact that P4P requires a short phase where an appropriate estimation of c is to be learned. Additionally, we see comparable results for P4P-o and P4P-u, which is explained by the relatively small effect of shaping reward scale on the probability of selecting a particular action in the case of ϵ-greedy tabular Q-learning.

7.2 Conversational recommendation environment

Performance metrics for all agents in the conversational recommender environment are listed in Table 3. The shielded baseline does not learn to solve the task at hand, i.e. average accuracy is 0.00. In contrast, accuracies for P4P are comparable to the unsafe baseline (Q1). Finally, all unsafe agents violate the specification: rewarding safe behavior is not sufficient for a safe agent. Figure 4 shows the accuracy of the tested approaches on varying constraints (Q2). P4P performs comparable to the unsafe baseline and comparable to or better than the safe baseline. Benefits of P4P grow as problems become more constrained (Q2).

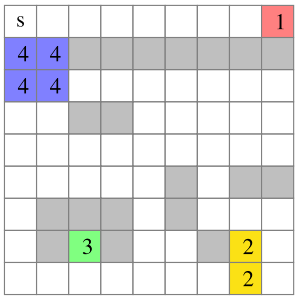
We continue to investigate sensitivity to the c parameter by comparing the results between P4P variants (Q3). We first revisit Table 3. Both P4P variants converge to high reward policies without a significant difference in reward, accuracy or number of turns between the two variants. However, Fig. 4 shows differences in data efficiency. Specifically, P4P with $c = 8$ converges to high rewards faster than the $c = 1$ variant. The signal for progress with respect to the safety constraints is more prominent with $c = 8$ without harming overall performance.

8 Discussion

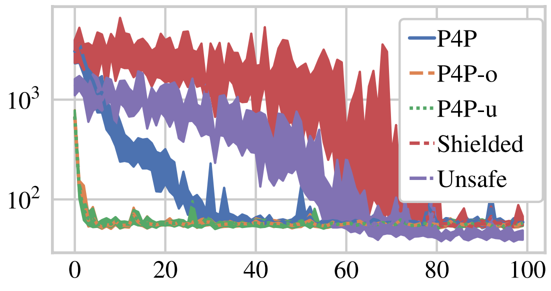
This work set out to address the problem of efficient and provably safe RL in settings where being safe need not be associated with high rewards. We formally introduced environments with symbolic safety constraints and showed that the performance of safe policies are only expected to perform equally to unsafe policies in a special case. We analyzed how constraints impact expected future rewards and showed a relation between expected rewards and the progress toward a goal in an automaton representation of the available symbolic knowledge.

We then proposed an algorithm to scale safe RL with constraint complexity based on symbolic reasoning. Reasoning is used to infer progress towards a symbolic goal. A reinforcement learner is then infused with this progress signal using additional rewards, following the convention of potential-based reward shaping. We evaluated the so-called P4P algorithm on two existing environments, one of which with real-world constraints. We found that it significantly outperforms baselines and scales well as problems become more constrained. Additionally, we introduced an approach for tuning its single additional hyperparameter in an online fashion and showed that the algorithm is robust against various values of this parameter.

In P4P, safety constraints are expressed at a symbolic, intensional level and need not align with large total rewards. Such problems are abundant in e.g. regulated domains such as healthcare and finance. Here, regulatory constraints prevent rare yet undesirable events,



(a) Environment



(b) Episode length per episode in the grid world environment

Fig. 3 Grid world with start position 's'. Positions marked in gray are to be avoided

Table 3 Recommendation environment test set results (mean $\pm$ 95% confidence interval)

	Unsafe		Safe	
	Baseline	P4P	Shielded	P4P
Reward	5.43 ± 1.94	6.43 ± 1.94	-11.93 ± 1.73	**5.82 ± 1.96**
Accuracy (%)	65.84 ± 2.75	**79.60 ± 2.74**	0.00 ± 0.00	**79.16 ± 2.74**
Violations (%)	80.16 ± 11.36	74.00 ± 7.49	0.00 ± 0.000	0.00 ± 0.000
Turns	7.73 ± 1.539	9.49 ± 1.591	11.93 ± 1.730	10.01 ± 1.561

Bold denotes significant improvements w.r.t. baseline/shielded

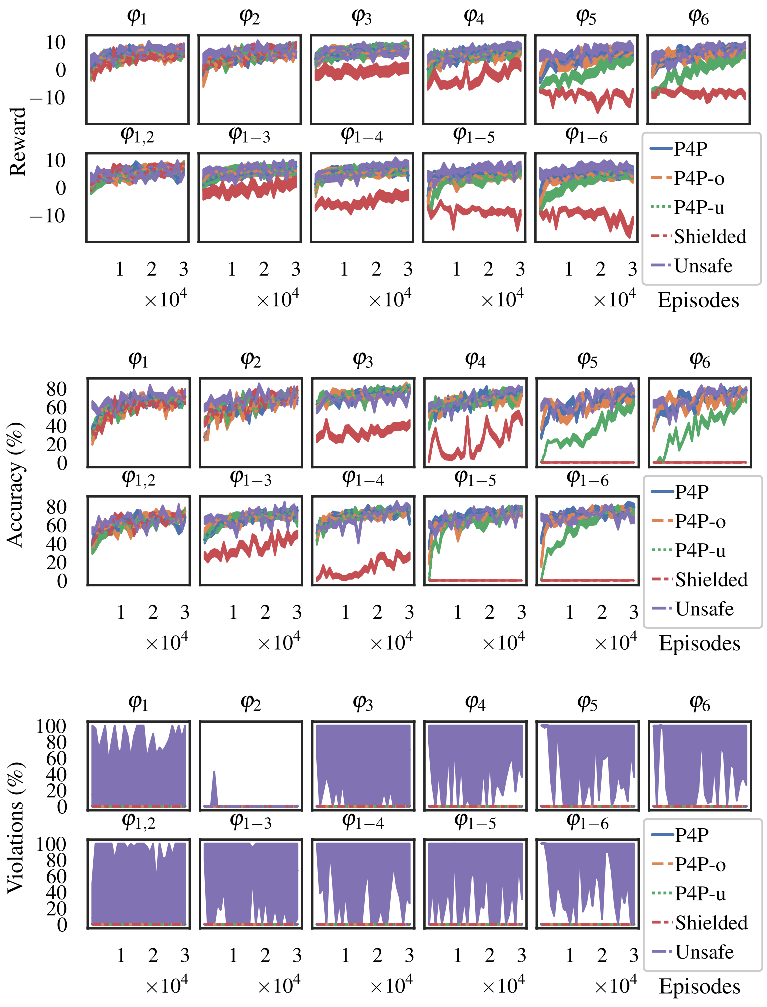


Fig. 4 Test set results in an increasingly constrained recommendation environment (mean and 95% confidence intervals)

unwelcome long-term effects and negative externalities. P4P exemplifies that safety in RL can be achieved at negligible performance penalty if learning and reasoning are combined.

The findings presented here inspire various directions for future work. Firstly, we have seen that constraints have a varying effect on learning efficiency. It would be useful if these could be estimated analytically and up-front by building on the framework of safety-constrained environments as first introduced by Alshiekh et al. (2018). Secondly, there is an interesting direction in altering the approach to be applicable to settings that do not necessarily involve safety constraints, but where knowledge about suitable policies is available at a symbolic level. Of particular interest here is the decomposition of the full RL task in subtasks, as has been proposed recently by Andreas et al. (2017) and Illanes et al. (2020). The use of constraints may be an interesting alternative if more fine-grained symbolic information is not available. Thirdly, the approach presented here can be combined with methods that learn a set of symbolic labels for the state and action spaces or that learn an automaton representation of the problem (Hasanbeig et al. 2021). This is of particular interest if the safety constraints are not *strict*, because learning these requires violation of the constraints during early stages.

Funding This study was funded by ING Bank N.V.

Data availibility Data and code will be made available upon acceptance of this paper. FdH contributed the code, performed experiments and drafted the manuscript. FdH, VFL, MH and FvH contributed the conception and design of study, analysed and interpreted data and critically revised the manuscript.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Research involving human or animal subjects No ethical approval was sought for this study on the grounds of no involvement of any human or animal subjects.

References

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., & Topcu, U. (2018). Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Andreas, J., Klein, D., & Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 36th international conference on machine learning vonference* (pp. 166–175). PMLR.

Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. MIT.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems, 29*, 1471–1479.

Bloem, R., Könighofer, B., Könighofer, R., & Wang, C. (2015). Shield synthesis. In *International conference on tools and algorithms for the construction and analysis of systems* (pp. 533–548). Springer.

Brafman, R. I., De Giacomo, G., & Patrizi, F. (2018). LTLf/LDLf non-Markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. A. (2019). Large-scale study of curiosity-driven learning. In *International conference on learning representations*.

Camacho, A., Chen, O., Sanner, S., & McIlraith, S. A. (2017). Non-markovian rewards expressed in LTL: Guiding search via reward shaping. In *Tenth annual symposium on combinatorial search*.

Camacho, A., Icarte, R. T., Klassen, T. Q., Valenzano, R. A., & McIlraith, S. A. (2019). Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the 28th joint conference on artificial intelligence* (Vol. 19, pp. 6065–6073).

De Giacomo, G., Iocchi, L., Favorito, M., & Patrizi, F. (2019). Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling* (Vol. 29, pp. 128–136).

De Giacomo, G., Favorito, M., Iocchi, L., & Patrizi, F. (2020). Imitation learning over heterogeneous agents with restraining bolts. In *Proceedings of the international conference on automated planning and scheduling* (Vol. 30, pp. 517–521).

den Hengst, F., Hoogendoorn, M., Van Harmelen, F., & Bosman, J. (2019). Reinforcement learning for personalized dialogue management. In *International conference on web intelligence* (pp. 59–67). IEEE/WIC/ACM.

den Hengst, F., Grua, E. M., el Hassouni, A., & Hoogendoorn, M. (2020). Reinforcement learning for personalization: A systematic literature review. *Data Science, 3*(1), 107–147.

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019). Challenges of real-world reinforcement learning. In *ICML workshop on real-life reinforcement learning*.

Fu, J., & Topcu, U. (2014). Probably approximately correct mdp learning and control with temporal logic constraints. In *Proceedings of robotics: Science and systems* (Vol. 10).

Gaon, M., & Brafman, R. (2020). Reinforcement learning with non-markovian rewards. In P*roceedings of the AAAI conference on artificial intelligence,* (Vol. 34, pp. 3980–3987).

Grzes, M., & Kudenko, D. (2008). Plan-based reward shaping for reinforcement learning. In *International IEEE conference intelligent systems* (Vol. 2, pp. 10–22). IEEE.

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)* (pp. 3389–3396). IEEE.

Hasanbeig, M., Abate, A., & Kroening, D. (2020). Cautious reinforcement learning with logical constraints. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems* (pp. 483–491).

Hasanbeig, M., Jeppu, N. Y., Abate, A., Melham, T., & Kroening, D. (2021).Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *The 35th AAAI conference on artificial intelligence, AAAI* (Vol. 2, p. 36).

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Icarte, R. T., Klassen, T., Valenzano, R., & McIlraith, S. (2018). Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 37th international conference on machine learning conference* (pp. 2107–2116).

Illanes, L., Yan, X., Icarte, R. T., & McIlraith, S. A. (2020). Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the international conference on automated planning and scheduling* (Vol. 30, pp. 540–550).

Junges, S., Jansen, N., Dehnert, C., Topcu, U., & Katoen, J.-P.. (2016). Safety-constrained reinforcement learning for mdps. In *International conference on tools and algorithms for the construction and analysis of systems* (pp. 130–146). Springer.

Könighofer, B., Lorber, F., Jansen, N., & Bloem, R. (2020). Shield synthesis for reinforcement learning. In *International symposium on leveraging applications of formal methods* (pp. 290–306). Springer.

Mazala, R. (2002). *Infinite games* (pp. 23–38). Springer. ISBN 978-3-540-36387-3.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS deep learning workshop*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th international conference on machine learning* (pp. 278–287).

Pnueli, A. (1977). The temporal logic of programs. In *18th Annual symposium on foundations of computer science* (pp. 46–57). IEEE.

Pnueli, A., & Rosner, R. (1989). On the synthesis of a reactive module. In *ACM SIGPLAN-SIGACT* (pp. 179–190).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT.

Tomic, S., Pecora, F., & Saffiotti, A. (2020). Learning normative behaviors through abstraction. In *Proceedings of the 24th European conference on artificial intelligence*.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3–4), 279–292.

Wen, M., Ehlers, R., & Topcu, U. (2015). Correct-by-synthesis reinforcement learning with temporal logic constraints. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4983–4990). RSJ/IEEE.

Wiering, M., & Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, Learning, and Optimization, 12*, 3.

Zhang, H., Gao, Z., Zhou, Y., Zhang, H., Wu, K., & Lin, F. (2019). Faster and safer training by embedding high-level knowledge into deep reinforcement learning. arXiv preprint. arXiv:1910.09986

Authors and Affiliations

Floris den Hengst[1] · **Vincent François-Lavet**[2] · **Mark Hoogendoorn**[2] · **Frank van Harmelen**[2]

Vincent François-Lavet
vincent.francoislavet@vu.nl

Mark Hoogendoorn
m.hoogendoorn@vu.nl

Frank van Harmelen
frank.van.harmelen@vu.nl

[1] ING Bank N.V., Amsterdam, Netherlands

[2] Present Address: Vrije Universiteit Amsterdam, Amsterdam, Netherlands