Reward Shaping from Hybrid Systems Models in Reinforcement Learning *

Marian Qian[1][0000−0002−4002−5208] and Stefan Mitsch[1][0000−0002−3194−9759]

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
marianq@andrew.cmu.edu, smitsch@cs.cmu.edu

Abstract. Reinforcement learning is increasingly often used as a learning technique to implement control tasks in autonomous systems. To meet stringent safety requirements, formal methods for learning-enabled systems, such as closed-loop neural network verification, shielding, falsification, and online reachability analysis, analyze learned controllers for safety violations. Besides filtering unsafe actions during training, these approaches view verification and training largely as separate tasks. We propose an approach based on logically constrained reinforcement learning to couple formal methods and reinforcement learning more tightly by generating safety-oriented aspects of reward functions from verified hybrid systems models. We demonstrate the approach on a standard reinforcement learning environment for longitudinal vehicle control.

Keywords: Theorem proving · differential dynamic logic · hybrid systems · reinforcement learning · reward shaping

1 Introduction

Complex (autonomous) systems increasingly often employ learning techniques to implement control tasks, which poses serious safety challenges. Formal methods for learning-enabled systems—such as closed-loop neural network verification (e.g., Verisig [16, 15], NNV [27]), falsification [6], shielding [1, 17], Neural Simplex [24], input-output behavior explanations [3], and online reachability analysis and hybrid systems monitoring [21, 9, 10]—address these challenges by analyzing trained controllers for safety violations and explanations of their behavior, or by filtering the actions proposed by controllers with formally verified artifacts. Besides filtering unsafe actions during training, the training setup itself typically is not supported with formal methods. A particularly attractive approach for training controllers is reinforcement learning, for its seemingly straightforward way of specifying desired behavior with a reward function. Designing reward functions, however, is challenging, not least because they need to balance safety-oriented requirements with goal-oriented ones.

* This work was funded by the Federal Railroad Administration Office of Research, Development and Technology under contract number 693JJ620C000025.

In this paper, we base on ideas from logically constrained reinforcement learning [11, 13, 12] to develop a formal approach to generating the safety-oriented aspects of reward functions from verified predictive hybrid systems models. The main intuition behind our approach is that hybrid systems models describe safety envelopes that can be turned into formally verified runtime monitors [22]. These runtime monitors not only distinguish safe from unsafe behavior, but with an appropriate quantitative interpretation can be used to measure the robustness of actions with respect to such safety envelopes. The challenge in deriving a useful (for reinforcement learning) robustness measure from a formal model is that relative importance of safety aspects is not immediately obvious from the formal model alone, and that differences in units makes comparison of the magnitude of robustness values across different aspects of the formal model difficult. For example, an autonomous vehicle model may encode brake force limits and speed limits: as the vehicle approaches a speed limit, it is acceptable to experience decreased robustness in brake limit in order to not violate the posted speed limit. Another challenge is that measure-zero safety aspects can hide progress or regression in other requirements.

We address these challenges by adapting robustness measures from metric-temporal logic [7] and signal-temporal logic [5], and by developing signal rescaling [28] operators to adjust the relative importance of competing safety aspects.

The benefits of this approach are that the safety specification is rigorously checked for correctness and the resulting reward function inherits the predictive nature of the hybrid systems model and its safety guarantees. The contributions of this paper are threefold: based on [22, 7] we develop a quantitative interpretation of hybrid systems models with an account for measure-zero requirements; we develop signal rescaling [28] operators to specify relative importance of (competing) safety aspects in the formal model; and we evaluate our approach on a standard reinforcement learning environment for longitudinal vehicle control [4].

2 Background

In this section, we summarize background theory and introduce notation.

2.1 Differential Dynamic Logic

Differential dynamic logic dL [25] is a formal language for hybrid systems written as hybrid programs. The syntax of *hybrid programs* (HP) is described by the following grammar where α, β are hybrid programs, x is a variable and $e, f(x)$ are arithmetic expressions (terms) in $+, -, \cdot, /$ over the reals, Q is a logical formula:

$$\alpha, \beta \ ::= \ x := e \mid x := * \mid ?Q \mid \{x' = f(x) \ \& \ Q\} \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

Assignment $x := e$ assigns the value of term e to x (e.g., compute acceleration to meet speed limit after T time $a := (v - v_{\text{des}})/T$), and nondeterministic assignment $x := *$ assigns any real value to x. Tests $?Q$ abort execution and discard the run if Q is not true, possibly backtracking to other nondeterministic alternatives. A typical modeling pattern combines nondeterministic assignments

with tests to restrict the chosen values to some set (e.g., choose control within brake and acceleration limits: $a := *; ?-B \leq a \leq A$). Differential equations $\{x' = f(x)\ \&\ Q\}$ are followed along a solution of $x' = f(x)$ for any duration as long as the evolution domain constraint Q is true at every moment along the solution (e.g., speed changes according to acceleration/deceleration, but does not revert when hitting brakes $\{v' = a\ \&\ v \geq 0\}$). Nondeterministic choice $\alpha \cup \beta$ runs either α or β (e.g., either accelerate or brake), sequential composition $\alpha;\beta$ first runs α and then β on the resulting states of α (e.g., first control, then motion), and nondeterministic repetition α^* runs α any natural number of times (e.g., repeated control and environment loop).

The formulas of dL describe properties of hybrid programs and are described by the following grammar where P, Q are formulas, f, g are terms, $\sim \in \{<, \leq, =, \neq, \geq, >\}$, x is a variable and α is a hybrid program:

$$P, Q ::= f \sim g \mid \neg P \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid P \leftrightarrow Q \mid \forall x\, P \mid \exists x\, P \mid [\alpha]\, P \mid \langle\alpha\rangle P$$

The operators of first-order real arithmetic are as usual with quantifiers ranging over the reals. For any hybrid program α and dL formula P, the formula $[\alpha]\, P$ is true in a state iff P is true after all runs of α. Its dual, $\langle\alpha\rangle P$ is true in a state iff P is true after at least one run of α.

The semantics of dL is a Kripke semantics in which the states of the Kripke model are the states of the hybrid system. A state is a map $\omega : \mathcal{V} \to \mathbb{R}$, assigning a real value $\omega(x)$ to each variable $x \in \mathcal{V}$ in the set of variables $\mathcal{V}$. We write $[\![Q]\!]$ to denote the set of states in which formula Q is true, $\omega \in [\![Q]\!]$ if formula Q is true at state ω, $\omega[\![e]\!]$ to denote the real value of term e in state ω, and ω_x^e to denote the state ν that agrees with ω except that $\nu(x) = \omega[\![e]\!]$. We write $\mathrm{FV}(P)$ for the set of free variables in formula P, and $\mathrm{BV}(\alpha)$ to denote the bound variables of program α, see [25]. The semantics of hybrid programs is expressed as a transition relation $[\![\alpha]\!]$ [25]. The differential equations and nondeterministic alternatives in hybrid programs make them an expressive specification language, but require computationally expensive methods similar to online reachability analysis for execution, which is detrimental to their use in reward functions in reinforcement learning. Next, we review ModelPlex [22] as a method to shift much of this computational complexity offline.

2.2 ModelPlex

ModelPlex [22] combines a universal offline safety proof $[\alpha]\, P$ with an existential reachability check whether two concrete states ω, ν are connected by the program α, i.e., whether $(\omega, \nu) \in [\![\alpha]\!]$. The safety proof witnesses that *all* states reachable by model α satisfy P, while passing the reachability check witnesses that the two concrete states ω, ν are connected by the program α, and so state ν inherits the safety proof, i.e., $\nu \in [\![P]\!]$. The reachability check is equivalently phrased in dL as a monitor specification $\langle\alpha\rangle \bigwedge_{x \in \mathrm{BV}(\alpha)} (x = x^+)$ [22]. The dL monitor specification allows ModelPlex, in contrast to online reachability analysis, to shift computation offline by using theorem proving to translate a hybrid systems

model into a propositional ModelPlex formula over arithmetic expressions. Note that the reachability check is inherently a property of the runtime execution and, therefore, the dL monitor specification and the resulting ModelPlex formula are never valid (they introduce fresh variables x^+, which means the existential reachability proof will not succeed offline for all states). Instead, the proof can be finished at runtime for two concrete states (a state ω providing values for x and a state ν providing values for x^+) by plugging in concrete measurements for all variables of the ModelPlex formula.

The set $\mathcal{M}$ of ModelPlex formulas $\phi : \mathcal{S}\times\mathcal{S} \to \mathbb{B}$ is generated by the following grammar ($\sim \in \{\leq, <, =, \neq, >, \geq\}$ and θ, η form the arithmetic expressions of the set $\mathcal{T}$ of ModelPlex terms in $+, -, \cdot, /$ over the reals, i.e., $\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$):

$$\phi, \psi ::= \theta \sim \eta \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi$$

When a ModelPlex formula $\phi \in \mathcal{M}$ is satisfied over states ω, ν, we write $(\omega, \nu) \models \phi$ as shorthand for $\omega_{x^+}^{\nu(x)} \in [\![\phi]\!]$, or in other words, $\phi(\omega, \nu)$ is true.

ModelPlex formulas are quantifier- and program-free, and are therefore easy (and computationally inexpensive) to evaluate from concrete measurements at runtime, which makes them attractive for use in reinforcement learning.

The predictive nature of ModelPlex monitors also makes them useful for safeguarding learned controllers during training and during operation [9, 10], in a shielding-like approach [20] based on hybrid systems models. In this paper, we take a complementary approach to shielding by interpreting ModelPlex monitors quantitatively in rewards.

2.3 Reinforcement Learning and Reward Shaping

Reinforcement learning involves training an agent to reach a goal by allowing the agent to explore an environment while trying to maximize its reward. The agent attempts different actions from the set of actions $\mathcal{A}$. For every action the agent makes, the environment takes a transition from state $s \in \mathcal{S}$ to a new state $s' \in \mathcal{S}$. A reward function then signals to the agent how useful the outcome of action a is: a negative reward signals that taking action a to reach state s' is discouraged, while positive reward encourages action a. Put differently, negative rewards encourage leaving "bad" states, while positive rewards encourage dwelling in "good" states. In Section 4, we develop a principled approach to generate safety-oriented aspects of the reward function from hybrid systems models.

Often, the agent has to learn multiple (competing) aspects of control (e.g., reaching a destination fast while respecting posted speed limits and conserving energy). This process of augmenting the reward function with multiple aspects is referred to as reward shaping. The motivation behind reward shaping is to provide additional reward for accomplishing subtasks that could lead towards the goal in the hopes to improve convergence and efficiency of training.

3 Related Work

Shielding (e.g., [1, 17], see [19] for a survey) prevents reinforcement learning agents during training and operation from taking actions that violate the spec-

ification. Specifications of shields are typically in linear temporal logic (LTL) and focus on discrete environments. Some approaches based on dL [9] target continuous environments, even from visual inputs [14]. Note that shields during training can be detrimental to safety in operation [20] if not specifically trained to return to safe states [9]. Neural Simplex [24] performs shielding with the additional feature of transferring control back to the learned controller when safe. In order to give feedback about compliance with shields to the learning agent during training and as a way of measuring robustness during operation, we follow logically constrained reinforcement learning (see e.g. [11, 13, 12]), but instead of LTL we use differential dynamic logic combined with signal rescaling [24] to shape safety-oriented rewards.

Previous works [18, 2] involving logic-based rewards include using environment-based temporal logic formulas as additional award augmented through potential-based reward shaping. Results have shown faster convergence and optimal policy performance; however, these have been only tested for average-reward learning algorithms rather than discounted-reward [18]. Additionally, reward functions can be augmented with specifications in signal temporal logic for desired agent behavior [2].

4 Rewards from Hybrid Systems Models

We take a complementary approach to shielding by interpreting hybrid systems models quantitatively through their relational abstraction as ModelPlex monitors. To this end, we define a hybrid systems normal form that is designed to align states of the hybrid systems model with states in the training process.

Definition 1 (Time-triggered normal form). *A hybrid systems model α in time-triggered normal form is of the shape $(u := ctrl(x); t := 0; \{x' = f(x,u), t' = 1 \ \& \ t \leq T\})^*$, where $u := ctrl(x)$ is a discrete hybrid systems model not mentioning differential equations.*

A hybrid program in time-triggered normal form models repeated interaction between a discrete controller $u := \text{ctrl}(x)$ that acts with a latency of at most time T and a continuous model $t := 0; \{x' = f(x,u), t' = 1 \ \& \ t \leq T\}$ that responds to the control choice u. Note that the discrete controller $u := \text{ctrl}(x)$ typically focuses on safety-relevant features (such as collision avoidance) while abstracting from goal-oriented features (such as desired cruise speed). The goal of this section is to develop a principled approach to reward shaping to translate the safety-relevant insights of formal verification to reinforcement learning. Fig. 1 illustrates how the states of a formal model correspond to states in reinforcement learning (note that unlike in usual RL notation, where states are responses of the environment and actions are drawn from a separate set, the states of the formal model include the values of actions).

Let $s_0 \in \mathcal{S}$ be the state before executing $u := \text{ctrl}(x)$, $s_1 \in \mathcal{S}$ be the state after executing $u := \text{ctrl}(x)$ (and before executing $t := 0; \{x' = f(x,u), t' = 1 \ \& \ t \leq T\}$), and $s_2 \in \mathcal{S}$ be the state after executing $t := 0; \{x' = f(x,u), t' = 1 \ \& \ t \leq T\}$ (which becomes s_0 in the next loop iteration). These states relate to the training

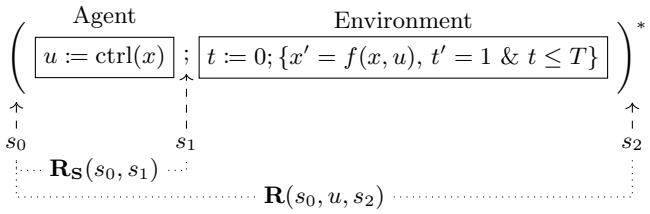


Fig. 1: Overview of aligning states in the formal model and the training process. The reward $\mathbf{R}(s_0, u, s_2)$ is given for the transition (s_0, u, s_2), whereas the states of a formal model include the values of actions, so taking action u traverses to intermediate state s_1 from which the environment responds by producing state s_2. Therefore, we can give separate reward $\mathbf{R_S}(s_0, s_1)$ from the predictive formal model for choosing action u in state s_0.

process in reinforcement learning as illustrated in Fig. 1: s_0 corresponds to the state before the agent picks an action, s_1 is the state after the agent chose an action $u \in \mathcal{A}$ (but before it is actuated in the environment), and s_2 is the result state of executing the action in the environment. In typical reinforcement learning setups, the reward associated with the transition (s_0, u, s_2) is computed by a reward function $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. We provide separate reward $\mathbf{R_S} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ directly for choosing action u in state s_0 from the *predictive* formal hybrid systems model, which requires a quantitative interpretation of ModelPlex formulas, as discussed next.

4.1 Quantitative Interpretation of ModelPlex Formulas

We adapt MTL/STL robustness measures [7, 8, 5] to define a quantitative interpretation of ModelPlex formulas, which describes *how robustly satisfied* a monitor is over two states.

Definition 2 (Quantitative ModelPlex). *The function $\mathbf{Q} : \mathcal{M} \to \mathcal{T}$ interprets a ModelPlex formula $\phi \in \mathcal{M}$ quantitatively as an arithmetic expression $\theta \in \mathcal{T}$ in $+, -, \cdot, /$ over the reals ($\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$):*

$$\begin{aligned}
&\mathbf{Q}(\theta \geq 0)(s_0, s_1) \triangleright \theta(s_0, s_1)\\
&\mathbf{Q}(\theta > 0)(s_0, s_1) \triangleright \theta(s_0, s_1)\\
&\mathbf{Q}(\theta = 0)(s_0, s_1) \triangleright \mathbf{Q}(\theta \geq 0 \wedge -\theta \geq 0)(s_0, s_1)\\
&\mathbf{Q}(\theta \neq 0)(s_0, s_1) \triangleright \mathbf{Q}(\theta > 0 \vee -\theta > 0)(s_0, s_1)
\end{aligned}$$

$$\mathbf{Q}(\phi \wedge \psi)(s_0, s_1) \triangleright \begin{cases} \mathbf{Q}(\psi)(s_0, s_1) & \text{if } \phi \equiv \theta{=}0 \text{ and } (s_0, s_1) \models \theta{=}0 \\ -|\theta(s_0, s_1)| & \text{if } \phi \equiv \theta{=}0 \text{ and } (s_0, s_1) \not\models \theta{=}0 \\ (\mathbf{Q}(\phi) \sqcap \mathbf{Q}(\psi))(s_0, s_1) & \text{otherwise} \end{cases}$$

$$\begin{aligned}
&\mathbf{Q}(\phi \vee \psi)(s_0, s_1) \triangleright (\mathbf{Q}(\phi) \sqcup \mathbf{Q}(\psi))(s_0, s_1)\\
&\mathbf{Q}(\neg\phi)(s_0, s_1) \triangleright -\mathbf{Q}(\psi)(s_0, s_1)
\end{aligned}$$

where $\sqcup$ *is* max, $\sqcap$ *is* min, *the atomic propositions in* ϕ *are normalized to* $\theta = 0, \theta \neq 0, \theta \geq 0, \theta > 0$, *and conjunctions are reordered to list all* $\theta = 0$ *before inequalities.*

Note that equalities $\theta = 0$ result in measure-zero robustness when they are satisfied: in a sense, their robustness is only meaningful when violated, since there is only a single way to satisfy $\theta = 0$. In conjunctions of the shape $\theta = 0 \wedge \phi$, the chosen robustness definition avoids unnecessary measure-zero robustness by evaluating to $\mathbf{Q}(\phi)$ when $\theta = 0$ is satisfied. In contrast, the naive phrasing $\mathbf{Q}(\theta = 0 \wedge \phi) \triangleright \min(\mathbf{Q}(\theta \geq 0), \mathbf{Q}(-\theta \geq 0), \mathbf{Q}(\phi))$ would evaluate to 0 when $\theta = 0$ is satisfied and thus hide changes in robustness in ϕ, which can be valuable reward signals to the agent. Quantitative ModelPlex maintains safety by overapproximating the original monitor verdict, see Lemma 1.

Lemma 1 (Mixed Quantitative ModelPlex Overapproximates Verdict). *The quantitative interpretation maintains the monitor verdict, i.e., the following formulas are valid:* $\mathbf{Q}(\phi) > 0 \rightarrow \phi$ *and* $\mathbf{Q}(\phi) < 0 \rightarrow \neg\phi$ *for all ModelPlex formulas* $\phi \in \mathcal{M}$.

Proof. By structural induction on ModelPlex formula and term operators. □

Mixed inequalities in Lemma 1 require for the quantitative interpretation to be conservative in the sense of causing false alarms on weak inequalities (a robustness measure of 0 is inconclusive). When comparisons are restricted to only weak inequalities or only strict inequalities, we maintain equivalence between the quantitative and the Boolean interpretation of ModelPlex, see Corollary 1.

Corollary 1 (Weak/Strict Quantitative ModelPlex Maintains Verdict). *When comparisons are restricted to weak/strict inequalities, the quantitative interpretation maintains the monitor verdict, i.e., the following formula is valid for weak inequalities* $\mathbf{Q}(\phi) \geq 0 \leftrightarrow \phi$ *(for strict inequalities* $\mathbf{Q}(\phi) > 0 \leftrightarrow \phi$*) for all ModelPlex formulas* $\phi \in \mathcal{M}$ *restricted to weak inequalities* $\geq, =$ *in atomic propositions (strict inequalities* $>, \neq$*, respectively) and Boolean connectives* $\wedge, \vee$.

Proof. By structural induction on ModelPlex formula and term operators. □

With a quantitative interpretation of how robustly the choices of a reinforcement learning agent satisfy the formal model, we next discuss several ways of combining the safety reward with other goal-oriented reward elements.

4.2 Logical Constraint Reward

In order to include feedback about the constraints of a formal model into the reward function, the agent receives goal-oriented reward when operating safely according to the model (safety monitor is satisfied), but receives the goal-oriented reward adjusted with a penalty from the monitor when operating unsafely.

Definition 3 (Logical Constraint Reward). *Let $P \in \mathcal{M}$ be a ModelPlex formula for $\langle u := ctrl(x)\rangle \bigwedge_{x \in BV(u:=ctrl(x))}(x = x^+)$, let $\mathbf{R_G}$ be a goal-oriented reward function, and $\mathbf{R_S} = \mathbf{Q}(P)$ be the safety reward function from the quantitative interpretation of P. Let s_0, s_1, s_2 be the states before the agent chooses an action, after it chooses an action, and after the action is executed in the environment, respectively. The logical constraint reward is defined as:*

$$\mathbf{R}(s_0, s_1, s_2) = \begin{cases} \mathbf{R_G}(s_0, s_2) & \text{if } (s_0, s_1) \models P \\ \mathbf{R_G}(s_0, s_2) + \mathbf{R_S}(s_0, s_1) & \text{otherwise} \end{cases}$$

Def. 3 discourages the agent to violate the assumptions of the formal model, while ignoring the safety reward when satisfied. The intuition behind this is that positive reward from the formal model is largest when robustly inside the boundary of the formal model's safety envelope, which may encourage the agent to make overly cautious (robust) action choices instead of making progress. Whether Def. 3 prioritizes goal-oriented reward or safety-oriented reward is entirely determined by the relative magnitude of $\mathbf{R_G}(s_0, s_2)$ vs. $\mathbf{R_S}(s_0, s_1)$. This can sometimes be undesirable since it does not necessarily encourage the agent to avoid safety violations. In order to emphasize safety and prioritize some aspects of the formal model over other aspects (e.g., satisfying a speed limit vs. satisfying deceleration assumptions) we introduce *reward scaling* to change the magnitude of rewards while preserving the safety verdict.

4.3 Reward Scaling

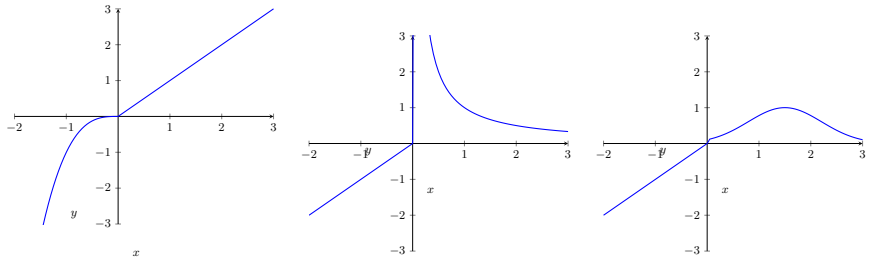
The logical constraint reward of Section 4.2 does not prioritize goal- vs. safety-oriented reward. Moreover, ModelPlex monitors do not distinguish between quantities of different units and sort in the formal model (e.g., a monitor conflates acceleration verdict, speed verdict, and position verdict into a single robustness measure). When used in a reward function, however, we may want to prioritize some safety aspects. For example, "violating" the brake assumptions of the formal model by having better brakes is acceptable and should not be penalized as hard as violating a speed limit. Here, we develop signal rescaling functions [28] to emphasize the verdict of the entire monitor or certain aspects of it.

Definition 4 (Scaling Function). *A scaling function $C : \mathbb{R} \to \mathbb{R}$ scales the result of a logical constraint reward function $\mathbf{R_S}$ such that the sign of its verdict remains unchanged, i.e., $C(0) = 0$ and $\forall r{\neq}0.\ C(r) \cdot r > 0$.*

Note that in the examples below we use $C(\theta)$ as a notation to indicate that the scaling function applies to a specific term θ.

Example 1 (Strong penalty). In order to emphasize that violating the formal model is highly undesirable, the following scaling function penalizes monitor violations while maintaining rewards for safety, see Fig. 2a:

$$C(\mathbf{R_S}(s_0, s_1)) = \begin{cases} \mathbf{R_S}(s_0, s_1) & \text{if } \mathbf{R_S}(s_0, s_1) > 0 \\ \mathbf{R_S}(s_0, s_1)^3 & \text{otherwise} \end{cases}$$



(a) Strong penalty (b) Boundary preference (c) Distance preference

Fig. 2: Reward scaling function illustrations

Example 2 (Boundary preference). In order to encourage behavior that follows close to a safety boundary (i.e. drive close to but not past a speed limit), the following scaling function gives more reward when the state's boundary distance is very small while giving less reward otherwise, see Fig. 2b:

$$C(\mathbf{R_S}(s_0, s_1)) = \begin{cases} \frac{1}{\mathbf{R_S}(s_0,s_1)} & \text{if } \mathbf{R_S}(s_0, s_1) > 0 \\ \mathbf{R_S}(s_0, s_1) & \text{otherwise} \end{cases}$$

Example 3 (Distance preference). In order to encourage behavior that follows a certain distance from a safety boundary (i.e. drive 10 km/h below but not past a speed limit), the following scaling function gives more reward when the state's boundary distance is close to the desired distance while giving less reward otherwise, see Fig. 2c:

$$C(\mathbf{R_S}(s_0, s_1)) = \begin{cases} e^{(-(\mathbf{R_S}(s_0,s_1)-1.5)^2)} & \text{if } \mathbf{R_S}(s_0, s_1) > 0 \\ \mathbf{R_S}(s_0, s_1) & \text{otherwise} \end{cases}$$

Uniform scaling maintains the sign of the monitor verdict, but emphasizes its importance relative to the goal-oriented components of a reward function. If the goal-oriented reward is designed to encourage reaching a goal fast (i.e., $\mathbf{R_G}(s_0, s_2) \leq 0$ for all states $s_0, s_2 \in \mathcal{S}$), the logical constraint reward should be restricted to safety violations and offset to "exceed" the goal-oriented reward: $\min(\mathbf{R_S}(s_0, s_1), 0) - |\delta|$, where δ is the minimum attainable goal-oriented reward.

The states s checked with a monitor $\mathbf{R_S}$ are composed of different aspects of the environment and agent behavior that can be scaled component-wise within $\mathbf{R_S}$ so certain aspects are given more weight in the monitor verdict.

If desired, reward scaling can decrease the importance of safety through decreasing the penalty for unsafe behavior, implying that violating safety for brief moments in time is allowed. This behavior has shown up briefly in our experiments when penalization for violating safety constraints was too low.

Definition 5 (Component-wise Scaling). *Scaling $C_{\downarrow_V} : \mathcal{T} \to \mathcal{T}$ applies scaling function C to components in variables V of a safety-reward function $\mathbf{R_S}$:*

$$C_{\downarrow_V}(\mathbf{R_S}) = \begin{cases} \min(f,g) \triangleright \min(C_{\downarrow_V}(f), C_{\downarrow_V}(g)) & \\ \max(f,g) \triangleright \max(C_{\downarrow_V}(f), C_{\downarrow_V}(g)) & \\ f \triangleright C(f) & \textit{if } FV(f) \subseteq V \\ f \triangleright f & \textit{otherwise} \end{cases}$$

Component-wise scaling maintains the sign of safety-reward function $\mathbf{R_S}$:

$$\begin{aligned} \forall s_0, s_1 \in \mathcal{S}\big(\quad & \mathbf{R_S}(s_0,s_1) = 0 \land C_{\downarrow_V}(\mathbf{R_S})(s_0,s_1) = 0 \\ \lor\ & \mathbf{R_S}(s_0,s_1) \neq 0 \land C_{\downarrow_V}(\mathbf{R_S})(s_0,s_1) \cdot \mathbf{R_S}(s_0,s_1) > 0\big) \ . \end{aligned}$$

Example 4 (Emphasizing speed). In order to emphasize that violating a speed limit is unsafe, while being close to it is desirable, we scale the difference between current speed v and speed limit v_{des} as follows.

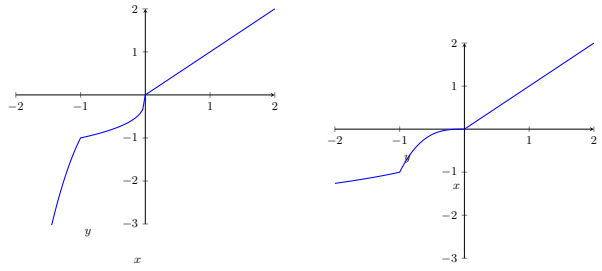
$$C_1(v_{\text{des}} - v) = \begin{cases} (v_{\text{des}} - v) & \text{if } v_{\text{des}} - v > 0 \\ (v_{\text{des}} - v)^{1/3} & \text{if } 0 \geq v_{\text{des}} - v > -1 \\ (v_{\text{des}} - v)^3 & \text{otherwise} \end{cases}$$

$$C_2(v - v_{\text{des}}) = \begin{cases} (v - v_{\text{des}}) & \text{if } v - v_{\text{des}} > 0 \\ (v - v_{\text{des}})^3 & \text{if } 0 \geq v - v_{\text{des}} > -1 \\ (v - v_{\text{des}})^{1/3} & \text{otherwise} \end{cases}$$

We apply the scaling to reward components in $\{v, v_{\text{des}}\}$: $C_1(v - v_{\text{des}})_{\downarrow_{\{v,v_{\text{des}}\}}}$ and $C_2(v_{\text{des}} - v)_{\downarrow_{\{v,v_{\text{des}}\}}}$. The above functions scale two different speed verdicts, $v - v_{\text{des}}$ and $v_{\text{des}} - v$, which in a formal model and thus a monitor may arise for different control choices (e.g., requiring to slow down when current velocity, v, exceeds the speed limit, v_{des} vs. allowing to speed up when $v < v_{\text{des}}$). When $v_{\text{des}} - v \leq 0$, safety is violated and therefore C_1 scales the negative verdict more aggressively by applying an exponential function. When $v - v_{\text{des}} \leq 0$, safety is satisfied, but going much slower than v_{des} is undesirable; therefore, C_2 scales the negative verdict using a fractional exponential function, see Fig. 3.

4.4 Potential-based Logical Constraint Reward

Reward shaping may cause unexpected behavior from the trained agent, as the reward directly influences what actions the agent takes and may cause the agent to learn a suboptimal policy [23]. To prevent unforeseen and unwanted behavior, potential-based reward shaping [23] provides additional reward to the agent while guaranteeing the agent will learn the optimal policy of the original reward function. The additional reward is specified using the transition from the current to the future state, and this transition is formalized as the difference in "potential" of the two states. Let $\Phi(s)$ characterize certain features of a state $s \in \mathcal{S}$



(a) $C_1(v_{\text{des}} - v)$ (b) $C_2(v - v_{\text{des}})$

Fig. 3: Component-wise scaling to emphasize speed rewards

(e.g., safety). Potential-based reward is then $\mathbf{R_P}(s, a, s') = \gamma\Phi(s') - \Phi(s)$, where s is the current state, a is the action taken to reach a future state s', and γ is some discount factor, which gets added to the original reward: $\mathbf{R}(s, a, s') + \mathbf{R_P}(s, a, s')$. The main intuition [23] why the policy is still optimal under this modified reward function is that the potential function itself does not prefer one policy over the other. Therefore, the original optimal policy is still preferred when the potential difference is added to the original reward function.

Example 5 (Potential-based Logical Constraint Reward). The original reward function $\mathbf{R}(s, a, s')$ is augmented with additional reward that is calculated using the predictive formal model. We align the loop iterations of the model in time-triggered normal form with the learning states s and s' of the reward function: let $s_0, s_1, s_2{=}s$ be the states from the formal model leading up to learning state s, and let $s'_0{=}s_2{=}s, s'_1, s'_2{=}s'$ be the states of the formal model corresponding to the transition (s, a, s'). Since monitors are evaluated over two model states, the safety potential associated with learning state s is a function of the previous action $\Phi(s) = \mathbf{R_S}(s_0, s_1)$, whereas the safety potential associated with learning state s' is according to the safety of the current action $\Phi(s') = \mathbf{R_S}(s'_0, s'_1)$. Then, the additional reward is $\mathbf{R_P}(s, a, s') = \gamma\Phi(s') - \Phi(s)$ with $\gamma = 1$, and the final reward function is: $\mathbf{R'}(s, a, s') = \mathbf{R}(s, a, s') + \mathbf{R_P}(s, a, s')$.

5 Evaluation

Formal model. In our evaluation, we adapt an existing formal model of a train protection system [26] and apply it to a standard reinforcement learning environment for longitudinal vehicle control [4]. The goal of the agent in longitudinal vehicle control is to drive forward as fast as possible while respecting posted speed limits. The speed limit control model (Model 1) comes with 3 components: a speed controller spd in lines 1–2 chooses acceleration to a desired cruising speed v_{des}, the automatic train protection atp in line 3 may override the choice of spd if the remaining distance to the speed limit d is unsafe, and the motion model drive in lines 4–5 describes how the vehicle position p and speed v change in response to the agent's acceleration choice.

Model 1 Speed limit control, adapted from [26, Fig. 5]

$$
\begin{array}{rll}
\text{spd} & 1 & \big(\quad (?v \le v_{\text{des}}; a := *; ? - B \le a \le A) \\
& 2 & \quad \cup (?v \ge v_{\text{des}}; a := *; ? - B \le a \le 0)\big); \\
\text{atp} & 3 & \text{if } \left(e - p \le \frac{v^2 - d^2}{2B} + \left(\frac{A}{B} + 1\right)\left(\frac{A}{2}\varepsilon^2 + v\varepsilon\right)\right) \text{ then } a := -B \text{ fi} \\
\text{drive} & 4 & t := 0; \\
& 5 & \{p' = v,\, v' = a,\, t' = 1 \;\&\; v \ge 0 \land t \le \varepsilon\}
\end{array}
$$

From Model 1, we use [22] to obtain a ModelPlex formula, whose main components reflect the model structure:

$$
\begin{aligned}
\text{slc} \equiv \quad & 0 \le v \le v_{\text{des}} \land -B \le a^+ \le A \land e - p > S \\
& \lor\, 0 \le v_{\text{des}} \le v \land -B \le a^+ \le 0 \land e - p > S \\
& \lor\, 0 \le v \land a^+ = -B \land e - p \le S \\
& \qquad \text{where } S \equiv \frac{v^2 - d^2}{2B} + \left(\frac{A}{B} + 1\right)\left(\frac{A}{2}\varepsilon^2 + v\varepsilon\right)
\end{aligned} \tag{1}
$$

Environment. LongiControl [4] provides a longitudinal vehicle control environment with state space $[x(t), v(t), a(t), a_{\text{prev}}(t), v_{\text{lim}}(t), \vec{v}_{\text{lim,fut}}(t), \vec{d}_{\text{lim,fut}}(t)]$ of vehicle position $x(t)$, speed $v(t)$, acceleration $a(t)$, previous acceleration $a_{\text{prev}}(t)$, current speed limit $v_{\text{lim}}(t)$, upcoming two speed limits $\vec{v}_{\text{lim,fut}}(t)$, and distances to the upcoming two speed limits $\vec{d}_{\text{lim,fut}}(t)$. The action space is continuous acceleration in the interval $[-3, 3]\,\text{m/s}^2$, and the sampling time is 0.1 s. The environment has posted speed limits of 50 km/h at $[0, 250)$ m, 80 km/h at $[250, 500)$ m, 40 km/h at $[500, 750)$ m, and 50 km/h after 750 m.

We map the environment to the ModelPlex formula (1) as follows: $x(t) \mapsto p, v(t) \mapsto v, a(t) \mapsto a^+, v_{\text{lim}}(t) \mapsto v_{\text{des}}, 3 \mapsto A, 3 \mapsto B, 0.1 \mapsto \varepsilon$ with two separate configurations in order to demonstrate how the formal model can influence the behavior of the trained reinforcement learning agent:

Configuration 1 encourages behavior similar to Model 1, satisfying a speed limit before the speed limit begins: $\vec{v}_{\text{lim,fut}}(t)_1 \mapsto d, x(t) + \vec{d}_{\text{lim,fut}}(t)_1 \mapsto e$, i.e., the first elements of the speed limit and speed limit position vectors are handed to the monitor

Configuration 2 to illustrate the effectiveness of the monitor in influencing learned behavior, this configuration encourages "unusual" behavior opposing r_{forward} (2) by favoring meeting the posted speed limit by the end (rather than the beginning) of the speed limit: $v_{\text{lim}}(t) \mapsto d, x(t) + \vec{d}_{\text{lim,fut}}(t)_1 \mapsto e$

Reward function. For evaluation, we use the reward function from [4, p. 1034] as a baseline, with weighted penalties for slow driving (r_{forward}), energy consumption

(r_{energy}), jerk (r_{jerk}), and speeding (r_{safe}):

$$\underbrace{-\xi_{\text{forward}} r_{\text{forward}} - \xi_{\text{energy}} r_{\text{energy}} - \xi_{\text{jerk}} r_{\text{jerk}}}_{\mathbf{R_G}} \underbrace{- \xi_{\text{safe}} r_{\text{safe}}}_{\mathbf{R_S}} \tag{2}$$

We obtain a safety robustness measure $\mathbf{Q}(\text{slc})$ from formula (1), and then replace the speeding penalty with the (scaled) safety reward function as follows:

Logical constraint reward (LCR) with $\mathbf{R_S} = \min(0, \mathbf{Q}(\text{slc})(s_0, s_1)) - 1$ to ignore the safety robustness measure when satisfied and offset safety violations to exceed the largest magnitude of the $\mathbf{R_G}$ components in (2).

Logical constraint reward scaling (LCRS) applies the component-wise scaling of Example 4 to the safety robustness measure $\mathbf{Q}(\text{slc})$.

Potential-based reward shaping (PBRS) applies the potential-based reward shaping of Example 5 to the safety robustness measure $\mathbf{Q}(\text{slc})$.

We used reward function **LCR** with Configuration1 and reward functions **LCR**, **LCRS**, **PBRS** with Configuration 2.

Model training. Using these different reward functions, we train several agents with the Soft Actor-Critic (SAC) method [1] and the hyperparameters of LongiControl [4, Table 4] with a learning rate of 1e-5. We trained the baseline agent with a learning rate of 1e-4.

Evaluation metrics. We evaluate the training process in terms of number of epochs until convergence, and the safety of the resulting agents in terms of the number and magnitude of speed limit violations. We also quantify how successful the agents are in reaching the goal, which is to drive as fast and as close to the speed limit as possible, by measuring the accumulated reward per (2) and the difference between agent speed v and the posted speed limits v_{des} during an evaluation period. Note that training and evaluation episodes do not terminate early under unsafe behavior of the agent but instead invoke a negative reward as a penalty. Below are the results of our experiments.

Results. Fig. 4 displays the average accumulated reward across several evaluation periods at every tenth epoch during training. Note, that during training we use different reward functions, which means their magnitude is not directly comparable. For the baseline reward function, we see that the reward converges to around -200 at 3000 epochs. The logical constraint reward using Configuration 1 converges to -300 at 1250 epochs. The following experiments using Configuration 2 converge to -325 at 1750 epochs for the logical constraint reward, -310 at 3000 epochs for logical constraint reward scaling, and -300 at 2250 epochs for potential-based reward shaping (faster or at the same rate as the baseline model). This helps with efficiency regarding how many epochs are required to finish training a reinforcement learning agent.

[1] GitHub of environment: `https://github.com/dynamik1703/gym_longicontrol`

Fig. 5a plots the accumulated reward according to the baseline reward function (2), so can be compared relative to each other. The baseline agent has the highest accumulated reward, indicating that it operates more aggressively (less robustly) than other agents. The robustness nature of the other agents can also be seen in Fig. 5b, which plots the $v_{\text{des}} - v$ to compare how well the agents achieve the goal of driving close to the speed limit.

When evaluating the agents for Configuration 1, we compare **LCR**(**1**), logical constraint reward, model against the baseline agent as their desired behavior is to drive as close to the speed limit as possible while satisfying the speed limit. After modifying the safety-oriented reward with the safety robustness measure from ModelPlex **Q**(slc), the resulting **LCR**(**1**) experiments resulted in agents that successfully satisfied the safety requirement by driving below the speed limit, while there were also some agents that violated safety when the upcoming speed limit was less than the current speed limit; using reward scaling to penalize unsafe behavior more can address this issue. The safe logical constraint reward agents had more robust behavior regarding maintaining safety as they left a larger gap to the desired speed limit as seen in Fig. 5b.

For the agents using Configuration 2, we also see that they are generally more robust compared to the baseline agent by the increased distance to desired speed in Fig. 5b. Note that for Configuration 2, the purpose was to demonstrate how the formal model can influence the behavior of the learned agent, which is most prominent when emphasizing speed with **LCRS**(**2**), logical constraint reward scaling: the agent's behavior shows a preference for reaching the 80 km/h speed limit at the end of the speed limit at 500 m in Fig. 5b. This behavior extends into violating the monitor when passing into the subsequent 40 km/h speed limit.

The other Configuration 2 models, **LCR**(**2**) and **PBRS**(**2**) did not violate safety defined by the monitor.

In summary, the experiments suggest that training from modified reward functions **LCR**(**1**) converges faster, and **LCR**(**2**), **PBRS**(**2**) generally satisfies safety. In addition, modifying aspects that correspond to the formal model can

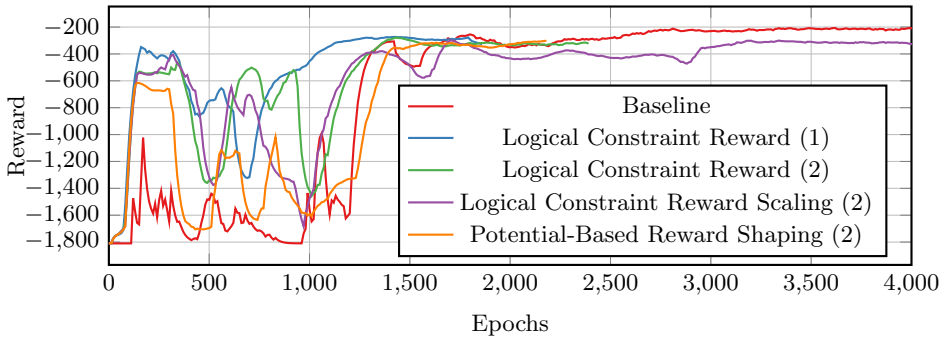
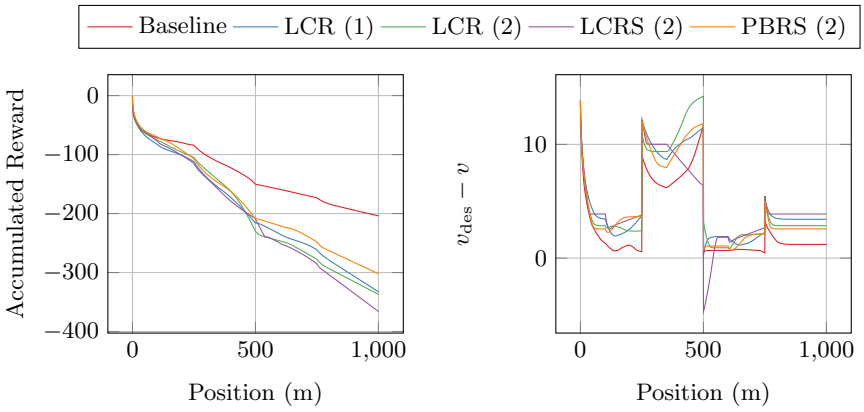


Fig. 4: Accumulated reward across evaluations during training; smoothed with exponential moving average



(a) Accumulated reward (b) Distance from speed limit

Fig. 5: Averaged reward and performance across five evaluation runs at end of training

effectively change the agent's behavior **LCRS**(**2**), which can help with designing the reward function for different goals of reinforcement learning problems.

6 Conclusion

We explored using predictive hybrid models with formalized safety specifications as an alternative to manually defining safety-oriented aspects of reward functions in reinforcement learning. [2] Based on logically constrained reinforcement learning, the agents we trained were implemented using dL-formalized safety rewards. The logical constraints of the safety-oriented reward are combined with the goal-oriented reward of the agent through reward scaling and potential-based reward shaping. We found that partly auto-generated reward functions produce agents that generally maintain the level of safety of hand-tuned reward functions and that reward scaling can be used to emphasize certain aspects of the generated reward functions. There were still agents that violated safety, specifically within logical constraint reward functions, and including dL-based shielding [9] can address these safety concerns. In addition, we observed faster convergence during training when using augmented reward functions, specifically for the logically constrained reward and the potential-based reward functions.

Future work includes generating goal-oriented rewards from liveness proofs, and extending the formal modeling language to specify scaling functions directly in the formal model, e.g., as design annotations. We also plan to use the predictive nature of the formal model for additional forms of reward shaping, e.g., to interpret the continuous dynamics of the formal model as a reward predictor for some reward aspects that are sustained over an extended time period, because they cannot be influenced instantaneously but only through affecting motion.

[2] GitHub for experiment code: `https://github.com/marianqian/gym_longicontrol_formal_methods`

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 2669–2678 (2018)
2. Balakrishnan, A., Deshmukh, J.V.: Structured reward shaping using signal temporal logic specifications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China, November 4 - 8, 2019. pp. 3481–3486 (2019). https://doi.org/10.1109/IROS40897.2019.8968254
3. Bayani, D., Mitsch, S.: Fanoos: Multi-resolution, multi-strength, interactive explanations for learned systems. In: Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings. pp. 43–68 (2022). https://doi.org/10.1007/978-3-030-94583-1_3
4. Dohmen, J., Liessner, R., Friebel, C., Bäker, B.: LongiControl: A reinforcement learning environment for longitudinal vehicle control. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,. pp. 1030–1037. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010305210301037
5. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings. pp. 264–279 (2013). https://doi.org/10.1007/978-3-642-39799-8_19
6. Dreossi, T., Donzé, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. J. Autom. Reason. **63**(4), 1031–1053 (2019). https://doi.org/10.1007/s10817-018-09509-5
7. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications. In: Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers. pp. 178–192 (2006). https://doi.org/10.1007/11940197_12
8. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/j.tcs.2009.06.021
9. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 6485–6492 (2018)
10. Fulton, N., Platzer, A.: Verifiably safe off-model reinforcement learning. In: Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I. pp. 413–430 (2019). https://doi.org/10.1007/978-3-030-17462-0_28

11. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives. In: Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings. pp. 108–124 (2020). https://doi.org/10.1007/978-3-030-59152-6_6
12. Hammond, L., Abate, A., Gutierrez, J., Wooldridge, M.J.: Multi-agent reinforcement learning with temporal logic specifications. In: AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 583–592 (2021). https://doi.org/10.5555/3463952.3464024
13. Hasanbeig, M., Abate, A., Kroening, D.: Cautious reinforcement learning with logical constraints. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020. pp. 483–491 (2020). https://doi.org/10.5555/3398761.3398821
14. Hunt, N., Fulton, N., Magliacane, S., Hoang, T.N., Das, S., Solar-Lezama, A.: Verifiably safe exploration for end-to-end reinforcement learning. In: HSCC '21: 24th ACM International Conference on Hybrid Systems: Computation and Control, Nashville, Tennessee, May 19-21, 2021. pp. 14:1–14:11 (2021). https://doi.org/10.1145/3447928.3456653
15. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verifying the safety of autonomous systems with neural network controllers. ACM Trans. Embed. Comput. Syst. **20**(1), 7:1–7:26 (2021). https://doi.org/10.1145/3419742
16. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In: Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part I. pp. 249–262 (2021). https://doi.org/10.1007/978-3-030-81685-8_11
17. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields (invited paper). In: 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference). pp. 3:1–3:16 (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.3
18. Jiang, Y., Bharadwaj, S., Wu, B., Shah, R., Topcu, U., Stone, P.: Temporal-logic-based reward shaping for continuing reinforcement learning tasks. In: Association for the Advancement of Artificial Intelligence (2021). https://doi.org/2007.01498
19. Könighofer, B., Bloem, R., Ehlers, R., Pek, C.: Correct-by-construction runtime enforcement in AI - A survey. CoRR **abs/2208.14426** (2022). https://doi.org/10.48550/arXiv.2208.14426
20. Könighofer, B., Lorber, F., Jansen, N., Bloem, R.: Shield synthesis for reinforcement learning. In: Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles - 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20-30, 2020, Proceedings, Part I. pp. 290–306 (2020). https://doi.org/10.1007/978-3-030-61362-4_16
21. Lin, Q., Mitsch, S., Platzer, A., Dolan, J.M.: Safe and resilient practical waypoint-following for autonomous vehicles. IEEE Control. Syst. Lett. **6**, 1574–1579 (2022). https://doi.org/10.1109/LCSYS.2021.3125717
22. Mitsch, S., Platzer, A.: ModelPlex: verified runtime validation of verified cyber-physical system models. Formal Methods Syst. Des. **49**(1-2), 33–74 (2016). https://doi.org/10.1007/s10703-016-0241-z

23. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999. pp. 278–287 (1999)
24. Phan, D.T., Grosu, R., Jansen, N., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural Simplex architecture. In: NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings. pp. 97–114 (2020). https://doi.org/10.1007/978-3-030-55754-6_6
25. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reason. **59**(2), 219–265 (2017). https://doi.org/10.1007/s10817-016-9385-1
26. Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings. pp. 246–265 (2009). https://doi.org/10.1007/978-3-642-10373-5_13
27. Tran, H., Cai, F., Lopez, D.M., Musau, P., Johnson, T.T., Koutsoukos, X.D.: Safety verification of cyber-physical systems with reinforcement learning control. ACM Trans. Embed. Comput. Syst. **18**(5s), 105:1–105:22 (2019). https://doi.org/10.1145/3358230
28. Zhang, Z., Lyu, D., Arcaini, P., Ma, L., Hasuo, I., Zhao, J.: On the effectiveness of signal rescaling in hybrid system falsification. In: NASA Formal Methods - 13th International Symposium, NFM 2021, Virtual Event, May 24-28, 2021, Proceedings. pp. 392–399 (2021). https://doi.org/10.1007/978-3-030-76384-8_24